Surveillance: From Solutions to New Problems

Paul Vines

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2017

Reading Committee:

Tadayoshi Kohno, Chair

Franziska Roesner, Chair

Zachary Tatlock

Program Authorized to Offer Degree:
Paul G. Allen School of Computer Science & Engineering

University of Washington

Abstract

Surveillance: From Solutions to New Problems

Paul Vines

Co-Chairs of the Supervisory Committee:
Professor Tadayoshi Kohno
Paul G. Allen School of Computer Science & Engineering

Assistant Professor Franziska Roesner
Paul G. Allen School of Computer Science & Engineering

The use of digital communications networks like the Internet has led to the creation of digital surveillance operations. The digital nature of these networks allows surveillance to be conducted in a broader and more automated fashion than traditional surveillance. In addition, the types of adversaries with surveillance motives and capabilities changes in the digital realm: while governments are the traditional operators of surveillance, this dissertation identifies the extensive online advertising ecosystem as a similarly capable surveillance framework. As the Internet continues to facilitate more and more everyday actions of individuals, there is increased risk that more and more of these everyday actions can become subject to surveillance. Understanding what surveillance is occurring, who is capable of accessing this data, and how individuals can protect themselves is critical to future privacy and security of individual users.

This dissertation addresses parts of these problems on all three fronts. First, it designs and assesses a covert communication system designed to facilitate two-way real-time text conversations over a steganographic covert channel in online games. Second, this dissertation evaluates user defenses in the webtracking context to rigorously evaluate the efficacy of these defenses using both traditional tracking mechanisms approaches and newer targeted

ad measurements. We find that that many existing recommended defenses are not effective, particularly that the type of ad-friendly tracking defenses that would be essential to protecting user privacy without disrupting ad-revenue based business models are significantly less effective than more aggressive ad-blocking defenses. Finally, it addresses who is capable of obtaining the data gathered by surveillance entities, specifically the advertising ecosystem, by exploring the ability for individuals to use this ecosystem for targeted surveillance. We refer to this as ADINT — for advertising intelligence — and find that the modern advertising ecosystem allows individuals or small groups with modest budgets ($1,000) to obtain surveillance capabilities reminiscent of state-level capabilities, such as fine-grained location tracking. These three components demonstrate the perniciousness of modern digital surveillance and the difficulty that users face in trying to defend against it.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

This dissertation represents several years of work that would not have been possible without the support and collaboration of many people.

Firstly, I would like to thank my advisors, Tadayoshi Kohno and Franziska Roesner, for their help and guidance about matters academic, professional, and personal. The work that this dissertation contains would not have been possible without their support: Yoshi's boundless enthusiasm and off-the-wall ideas and Franzi's practical consideration and insightful analysis of our research were both equally critical to this venture. In particular, any degree to which the ideas or writing in this dissertation are exceptional can be considered due to the diligent and exceptional help both of them have kindly provided to me.

I would also like to thank the professors I had the fortune to work with earlier in my graduate career on my way to finding the Security and Privacy Lab: Georg Seelig for taking an interest in me when I initially entered this program in Synthetic Biology. I thank Mike Ernst for allowing me to come and get my first taste of academic security work in the software engineering context and facilitating my transition to working directly with Yoshi. I would also like to thank Zach Tatlock for being on my reading committee and always faithfully interpreting my jabs at verification in the spirit they were created. Finally I thank Emma Spiro for feedback and interesting insights from outside the paranoid world of security and privacy.

I am thankful to the community of CSE as a whole for keeping my years here interesting and engaging. In particular I appreciate the members of the Security and Privacy Lab: Alexei Czeskis, Miro Enev, Karl Koscher, Temitope Oluwafemi, Ian

Smith, Ada Lerner, Alex Takakuwa, Peter Ney, Camille Cobb, Kiron Lebeck, Anna Kornfeld Simpson, Lucy Simko, Eric Zeng, and Kimberly Ruth for their comradery, friendship, support, and perhaps most of all their unique perspectives and openness to fascinating discussions on all manner of topics. I can only hope to find another such group of friends in my future. I also want to thank some other members of CSE that I have particularly benefited from knowing and working with: Will Scott, Raymond Cheng, Trevor Perrier, Eric Mullen, PJ Valez, and Justin Bare. I would like to thank Emily McReynolds and the members of the Tech Policy Lab for their lively debates and fascinating knowledge of technology policy and law. Finally, I thank the UW CTF team, Batman's Kitchen, for giving me the first taste of CTFs and catalyzing my decision to transition into security.

I also want to thank those that supported my career and academic pursuits since before I found UW. At Roanoke College I would like to thank Darwin Jorgensen, Leonard Pysh, Rama Balasubramanian, Durell Bouchard, and Anil Shende for the opportunities to see research first-hand and supporting me in my academic pursuits.

Looking beyond academia, I would like to thank a variety of sources for the inspiration and extracurricular education they have provided: GURPS Biotech for making me want to become a researcher, the Security Now! podcast for securing me my first internship, and Protest The Hero for providing the background to which most of this research was performed.

Finally, I want to thank my entire family for their support and encouragement — in various ways — of my academic career. I want to thank my brother, Mason Vines, for letting me hang around his computers and telling me to take my first CS class. I thank my sister, Maury Vines Wrightson, for always being enthusiastic about science and setting an example of academic pursuit for me to follow. I want to thank my parents, Roger and Sharon Vines, for making sure I learned just enough to know

anything I needed to know, but never so much that I thought I knew everything I would ever need, or anything I did not need; most of all, for teaching me how to learn and teach myself. I need to thank my wife, Ruthie, for encouraging me to move into security and for supporting both my work and my life through these years. Finally, I want to thank my family for their opinion that games come before schoolwork.

This dissertation and my graduate career spent creating it were supported by a National Science Foundation Award — CNS-1463968 and CNS-0846065 — by the Defense Advanced Research Projects Agency (DARPA) under contract FA8750-12-2-0107, the Short-Dooley Endowed Career Development Professorship., and by the Tech Policy Lab.

DEDICATION

To my parents, Roger and Sharon Vines, for encouraging my curiosity and supporting me in always learning new things. And my dear wife, Ruthie, for helping me discover what I wanted and staying by my side for these years in Seattle.

Chapter 1

INTRODUCTION

Surveillance has always been a concern for humans, but increasing reliance on networked technologies has increased the pervasiveness and decreased the cost for performing surveillance. Additionally, the opacity of modern software, networks, and data use practices make it difficult for targets of surveillance to know how, when, or by whom they are being surveilled.

Governments are the traditionally dominant operators of surveillance. In the modern context the governments of advanced nations appear to maintain this role; they are able to use their sovereign authority to insert surveillance technologies into various points in the underlying infrastructure used for networked devices and services. The Snowden disclosures have revealed that the privileged network positions advanced governments can obtain enable previously impossible forms of mass surveillance [39]. This low cost of surveillance by governments means more individuals than ever have reason to be concerned that they are being monitored if engaged in activities a government or agent within that government would take any interest in. A wide variety of defenses to aspects of this surveillance have been developed: secure messaging apps that ensure confidentiality and integrity of message content have proliferated in the years following the Snowden disclosures [49]. However, security of message *content* still allows metadata surveillance to take place, which can often be just as useful [21]. Securing communications against metadata-level surveillance — or hiding the existence of the communication from government surveillance altogether — is a fundamentally different type of problem from securing message content.

Difficult as protecting users from government surveillance is, the government is also not the only entity to have constructed a mass surveillance framework around the modern Internet. The latter parts of this dissertation turn towards another source of mass surveillance:

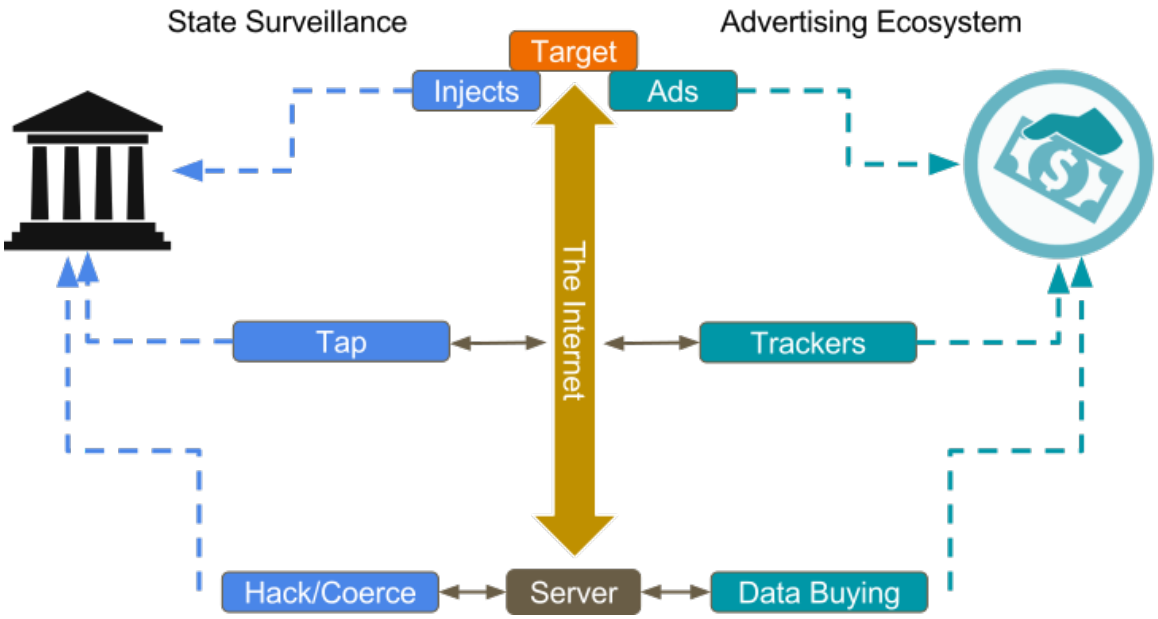

Figure 1.1: Abstract comparison of the surveillance capabilities of state intelligence and the data collection conducted by the advertising ecosystem. Traditional surveillance can inject malicious spyware onto a target's device to extract information; the advertising ecosystem is built to facilitate targeted injection of content into target devices in the form of ads. The Internet can be passively monitored at various points to collect target network traffic; tracking via redirects and analytics scripts are pervasively included on websites to collect target actions at the application level. States can use authority to coerce or simply hack the databases of many services to extract user data; advertisers purchase user data from services.

the commercial Internet. Much of the Internet economy is built on advertising revenues: this economic incentive has led to the creation of a massive tracking and advertising ecosystem deeply embedded into many parts of the Internet. This dissertation argues that the advertising ecosystem enables similar forms of surveillance normally associated with privileged network positions only available to government actors (see Fig. 1.1 for visualization of this).

Recent polls suggest users are uncomfortable with this level of data collection, storage, trading, and analysis by these corporations [71, 101, 115]. As a consequence of this desire to not be tracked online, there have been a number of privacy defenses created. These defenses vary widely in what they try to achieve and how they function technically. These defenses

are typically focused on preventing the web-tracking portion of the advertising ecosystem - the practice of observing which sites and pages a user visits. However, the number and variety of these defenses makes it difficult for users to understand which defenses actually protect their privacy and which do not, since casual users generally do not directly observe the advertising ecosystem's surveillance operations.

No matter how effective some of these defenses are, they remain focused on a single source of information in the advertising ecosystem: web-tracking. Other details, like user account information, purchases, service or app usage, and social media data are all other major components of information the advertising ecosystem collects and uses. The ability to collect and exchange this information among corporations has been mostly uninhibited because the motives and actions of advertising corporations are considered generally benign: they are generally assumed to try to more effectively sell products or services to users. While some consider this objective harmful, legislatures have generally disagreed, resulting in little legislation to protect user privacy from these practices.

However, the assumption that profit-motivated corporations are the only potential users of advertising ecosystem data is wrong. The capabilities of the advertising ecosystem are distributed among many distinct entities, and there is a focus on making it easier and cheaper for every business to use advertising. This means that now individuals or small groups can buy ads targeted with equal disturbing accuracy as global corporate brands. Furthermore, it means individuals can effectively extract the vast quantities of information the advertising ecosystem has collected about users and use this information for arbitrary purposes. This allows practically any individual to turn the data collection and analysis framework of the advertising ecosystem into their own surveillance service with capabilities reminiscent of government intelligence agencies.

1.1 Contributions

This dissertation focuses on research in several discrete aspects of modern surveillance, specifically it contributes:

1.1.1 Rook: Surveillance-Resistant Real-Time Communication.

In Chapter 2 this dissertation focuses on providing a defense for one aspect of mass government surveillance: communication metadata analysis. In response, it develops a novel system for covert communication between users. This system provides several important features that related censorship and surveillance circumvention systems lack, namely real-time two-way communication and the ability to operate with all system components inside the adversary's sphere of influence.

1.1.2 Bifocals: Examining Web Tracking Defenses Under Two Lenses of Inquiry

In Chapter 3 this dissertation focuses on a particular aspect of advertising ecosystem surveillance: web-tracking. It develops a new methodology for measuring the efficacy of tracking defenses by combining traditional analysis of tracking based on observing tracking *mechanisms* — HTTP requests and cookies — with newer *outcomes*-based techniques — measuring targeted ads. The results of this combined approach strengthens previous conclusions, provides contrasting data about the efficacy of some defenses, and rigorously evaluates some defenses not previously tested (clearing client side state).

1.1.3 ADINT: Exploiting Ad Targeting for Intelligence Gathering

In Chapter 4 this dissertation takes a broader view of the surveillance properties of the advertising ecosystem as a whole. Many users are concerned about the extent of data collection and use by large corporations for advertising purposes. However, this dissertation argues — and demonstrates — that this advertising ecosystem is available to individuals, not just the large corporations normally associated with it. I demonstrate that individuals can turn this ecosystem into a personal surveillance framework. This work explores the practical feasibility and potential capabilities of targeted advertising for intelligence gathering (ADINT) in two ways: (1) a case study using a single ad-network to establish the practicality of ADINT; (2) a survey of 21 ad-networks to determine the potential for different types of surveillance.

Chapter 2

ROOK: SURVEILLANCE-RESISTANT REAL-TIME COMMUNICATION

This chapter explores the challenges to securing users against surveillance attacks on the metadata of their communications, in addition to the actual content. In particular, I describe work my collaborators and I performed to design a surveillance defense in the form of a covert communication system to enable users to hold real-time conversations with one another without a powerful network-level adversary — like a government — being able to detect who is communicating or that there is any communication happening at all. This work was originally published in the Workshop on Privacy in the Electronic Society in 2015 [120].

2.1 Motivation and Overview

There are increasing concerns about the privacy of online communications throughout the world: there are several countries which have historically acted to both censor and surveil private communications within their borders [19, 123]. However, both surveillance and censorship of the Internet has continued to increase in scale, sophistication, and prevalence [47, 62, 129]. Researchers have proposed and built a wide variety of system that attempt to evade or circumvent this surveillance and censorship in the recent past — we refer to these as simply "circumvention systems" in the following text. Circumvention systems fall broadly into three categories with somewhat overlapping technical requirements.

1. *Content security* systems, like Signal [114], seek to preserve confidentiality, integrity, and authenticity of communications. In the wake of the Snowden disclosures these systems have received widespread publicity and a fairly high rate of adoption among varied user-bases [1]. Their continued existence is currently being called into question

by various governments [20, 29, 77], but from a technical perspective the problem of communication content security has largely been solved.

2. Censorship circumvention: in this case, a user is seeking access to some content that their ISP or governing authority has blocked access to. Classically this takes the form of a repressive regime blocking access to outside journalism that reveals government failures [129]. In general these circumvention systems are aimed at getting the user some form of tunnel to outside the sphere-of-control of the censoring network. From there they can then retrieve whatever censored content they want.
3. Anonymity services, or metadata-surveillance circumvention, is the last category of circumvention systems. Here the objective is to allow users to communicate without a surveillor being able to determine who is communicating with whom. Tor [28] is an example of a deployed anonymization system. Recently new systems with more stringent privacy guarantees have been developed [118]. However, many of these systems are very noisy and easy to detect. Thus, if the adversary is also a willing censor (see above), then they are likely very willing to detect and block these anonymity services.

Of these three categories, Rook aims to provide a metadata-surveillance defense, but do so using techniques more similar to a traditional censorship circumvention system. Specifically, Rook provides anonymity by hiding the existence of any communication at all, which precludes the adversary detecting who is communication. However, unlike in classic censorship scenarios, Rook aims to act entirely within the adversary's sphere-of-control. Many circumvention systems across all three categories rely on at least one component of the system being free to operate without interference from the adversary. In contrast, Rook only relies on the adversary not blocking its cover-traffic (online game network traffic).

In this chapter we present Rook as a low bandwidth low latency (approx. 30 bits/second) censorship resistant communication platform. While Rook can be used to secretly communicate any kind of data, the structure of its network and limitations of its bandwidth make it best suited for chat applications. We intend Rook to be used to facilitate secret IRC-like services among users. These services can run the Off-The-Record (OTR) protocol [12] between

clients to ensure end-to-end security even from the Rook server. Rook was partially inspired by recent works on pushing circumvention systems further into the application layer [27, 55], but also represents a new direction for circumvention systems in several ways: first, it utilizes the normal network traffic of online games to hide its data; only one other system, published concurrently [48], has explored this area of applications. Games have a number of features that make them ideal host applications for hiding data which are further explained below. Second, Rook alters the host game traffic in full compliance with the application's protocol; this means an adversary must first commit the resources to develop a program to correctly parse the application protocol, and then must also commit computational resources to each suspected Rook connection since the adversary cannot use a stateless single-packet inspection model to detect a Rook connection. As is the case with all of these circumvention systems, Rook is not necessarily undetectable, and we outline possible new research directions for attacks below (see Section 2.5 and 2.6). However, Rook does represent a significant increase in the cost of trying to detect and block its use.

Many different types of applications have been used as cover or host applications for previous circumvention systems. Skype in particular has proven popular because its calls operate on a peer-to-peer connection and are assumed to be a reasonably legitimate activity for a user to be engaged in [55, 89]. Multiple TCP/IP-header based covert channels have also been developed. Online games provide a similar opportunity but with greater deniability on the part of users. Traditionally, many online games have allowed individuals to host their own private servers to reduce the resource burden on the companies making the games. This is particularly the case for the genre of game Rook is primarily focused on: the First Person Shooter (FPS). The existence of privately-hosted servers creates opportunities for communities to arise and causes many regular players of these games to play almost exclusively on one or a handful of servers. This provides a completely legitimate cover for a Rook user to repeatedly connect to the same game server over and over to communicate with the other Rook users also connecting to these game servers. Furthermore, legitimate players will often play for hours at a time, day after day. This could be significantly more suspicious in the

case of another application, such as Skype, repeatedly being used to contact the same IP for hours at a time every day. Finally, like VoIP services, games are a widespread and popular form of network use [116]; we believe a censor would face similar dissent to a decision to block all Internet-gaming as they would to blocking all VoIP. The general design of Rook is not specific to a game, and so if a censor attempted to block Rook by blocking a single game, it could be adapted to another.

Another advantage of games is that they do not imply actual communication: a Skype connection inherently implies the two parties are sharing information, while a game client connecting to a server does not imply any more communication than the data packets being used to play the game. Since the connection is not encrypted, an adversary could easily detect if there is other information being sent via chat or voice in the game. This creates a plausible deniability that Rook users are connected in any way. Where a Skype connection implies two IPs are exchanging information, a game connection only implies two IPs happen to be playing the same game, and probably are not even aware of who the other IP is. Finally, games provide a way to exchange information while not interrupting the legitimate activity. In most cases the host application, if it is even being run, is not performing its normal operation: e.g., Skype when being used to proxy web traffic is not also providing a true VoIP conversation [55, 89]. Instead, the game is actually being played normally while information is exchanged; the adversary could plant a user in the same game server and not observe any significant difference in how a Rook user played versus a normal legitimate player (see Section 2.4).

Rook was developed with the FPS as the primary type of game to be used, because these types of games provide network traffic that is both low-latency and based on unpredictable user-actions. FPS games generally feature between 8 and 128 players on a server at a time. Each player controls one avatar inside a 3-dimensional game world in which they attempt to maneuver and kill the other avatars. As an example implementation, Rook uses the FPS Team Fortress 2 [117], based on the Source Engine from Valve Software. The Rook design can also be used for other types of games, although it may require modification particularly

in the case of TCP-based games.

The rest of this chapter is laid out as follows: Section 2 describes the design and architecture of the Rook system. Section 3 describes our example implementation of Rook for Team Fortress 2. Section 4 contains our evaluation of Rook against a a variety of current and future proposed attacks including a game-specific trigram value analysis. Section 5 describes related works and how Rook's approach fits into the context of current circumvention system research. Section 6 discusses potential future works in this space building upon Rook. Section 7 contains a summary of Rook's contributions and analysis of our implementation.

2.2 System Design

Rook is a system to facilitate secretly passing data between two Rook clients and a Rook server, operating on machines running game clients and a game server, respectively. In this case, secret is defined as the act of sending the data being unobservable from to an outside observer, as well as the contents of the data itself being encrypted. The data channel between a Rook client and Rook server is composed of two one-way channels to create an overall bidirectional data channel. The creation of the channel requires a shared secret key between the Rook client and Rook server.

2.2.1 Intended Use Model

While Rook is fundamentally capable of secretly transporting any kind of data, specific features and limitations of both Rook and online games make it particularly suited towards functioning as a secret chat server. Rook provides low bandwidth but also real-time communication, this mostly precludes it from being used for higher-bandwidth tasks such as normal web-browsing or viewing censored videos and pictures. Furthermore, because an online game is being used as the cover application, it would be odd for a user to connect to a server that is in a distant country, since such a server would typically have high latency compared to a more local server.

The intended model for Rook use is a local Rook server running on a machine that is

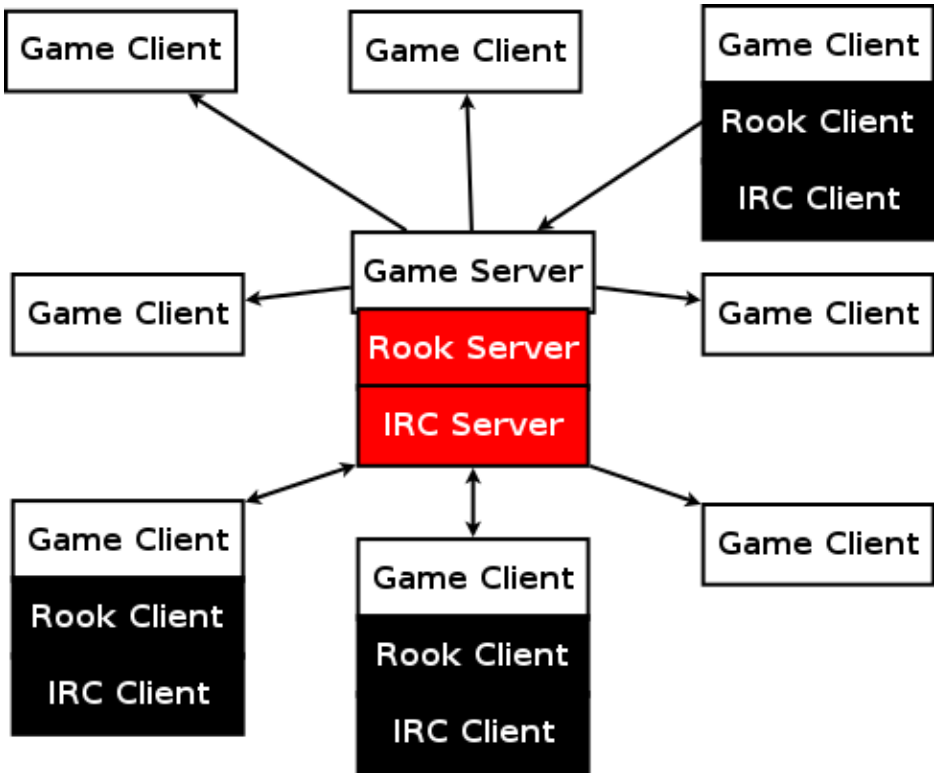

Figure 2.1: Example formation of a Rook network: one Rook server with multiple Rook clients and normal game clients connected to it.

running one or more private game servers used by both Rook users and normal game players. This Rook server would then also run a chat server such as a standard IRC server, that is tunneled through Rook. Rook users who possess a shared key with the Rook server can then group-chat with other Rook users on the same server as they would on a normal IRC server, but without having to fear censorship. For private one-on-one chats Rook users could easily use Off-The-Record messaging over Rook to ensure privacy even from the Rook server. We discuss our prototype implementation which includes these features in Section 2.4.

2.2.2 Threat Model

For this work we assume the following threat model for our monitor adversary, similar to the standard warden and prisoners threat model used in steganography [3]. The monitor is

attempting to detect and/or block the exchange of any information besides legitimate game information. The monitor is considered to have the following capabilities:

- All network traffic between all client and the server are observed
- The normal application traffic is unencrypted
- The monitor can store all data observed over a gaming session and run statistical analyses on it
- The monitor has no internal access to the devices running the game clients and server
- Users can conduct a 1-time secure rendezvous to exchange a shared secret key
- The monitor can join the server and observe the actual gameplay (this is not possible if the Rook server is password protected, as some private servers are)
- The monitor does not wish to disrupt legitimate gameplay of innocent users
- The monitor can conduct some active probing of game client and servers, but cannot disrupt legitimate traffic for extended periods of time
- The monitor seeks evidence that information, aside from normal game data, is being exchanged

2.2.3 Criteria for Success

Rook is a successful censorship resistant platform if the monitor cannot either:

- Positively identify use of Rook more often than falsely identifying legitimate game traffic as use of Rook
- Successfully disrupt Rook communication without impacting legitimate gameplay.

We assume that even if Rook becomes popular, it will still represent a minority of game traffic. Therefore, the first criterion is subject to this difference, so that even a small false positive rate in a detection scheme can yield larger absolute numbers of false positives than true positives. Additionally, the efficiency and practicality of deploying a given detection scheme should be taken into account. General-purpose Deep-Packet Inspection (DPI) techniques, for example, are easier for an adversary to use on a wide-scale than specialized detectors

that require storing and analyzing entire traffic captures.

2.2.4 System Overview

The essence of Rook's scheme for secretly sending data is to identify portions of game packets that can contain many different values and then infrequently select a packet and replace some of that mutable data with other legitimate game values representing the secret data it wishes to communicate. On the receiving end, the receiver does the inverse: it also identifies these mutable portions of the packet and decodes the values inserted by the sender back into the secret data. This concept is fundamentally simple but grows in complexity in the context of defeating the various kinds of attacks that can be launched against it. The following sections explain in detail the components of the Rook sending and receiving scheme and why they are designed the way they are.

2.2.5 Mutable Fields

Rook relies on finding *mutable fields* within game packets. In theory, we could replace any arbitrary bits in the payload of the packet with the secret data, because the packet will never be sent to the actual game process on the other end. However, many bit values are immutable with respect to the protocol of the particular game being used.

For example, many games have application-level sequence numbers in the payload, and overwriting these with arbitrary data would cause the application to issue an error and/or crash.

A relatively easy attack against Rook would be to passively duplicate game packets as they traverse the network and try to parse them using the game's protocol; if they frequently fail to parse then it is unlikely they are merely being mangled in-transmission and so reasonable to suspect use of Rook. Therefore, the implementation of Rook must correctly implement at least part of the game network protocol in order to be able to parse packets and correctly identify which bits are part of *mutable fields*. These are the only bits which are modified when Rook sends data.

In addition to this requirement, the values of some fields are reflected by subsequent packets sent by the receiver of the value. For example, the value of the weapon switch command sent from the client to the server is reflected by subsequent server-to-client messages confirming which weapon is now selected. A passive attacker could relatively easily observe the original field being sent, and then observe whether or not its value was reflected in subsequent return packets. If the packet containing the original field was dropped this field value would not be reflected and the attacker could suspect the use of Rook. Fortunately there are relatively few fields like this, and so the packet parser module for Rook must simply be set to never alter packets containing these fields.

2.2.6 Symbol Tables

Unfortunately even if the *mutable fields* of a game packet can be correctly determined, not all combinations of these bits are necessarily valid or reasonable to exist in normal game traffic.

For example, imagine a *mutable field* sent from the server to the client representing the velocity vector of another avatar in the game. That value might be encoded as a 16-bit float. However, it might be that the velocity of an avatar is never actually greater than 100.0. If Rook replaced all 16-bits with arbitrary data it could easily be spotted by an adversary parsing packets and looking for velocity vectors with values greater than 100.0.

To prevent this type of attack, Rook does not insert the raw bits of the data it is sending. Instead, Rook keeps a *symbol table* for each *mutable field*. This *symbol table* is constructed by observing normal gameplay at the start of a game connection. For each *mutable field* encountered in the observed data a count of what values it had is kept. This count is then translated into a frequency of a certain value appearing in the *mutable field*. The *symbol table* for that *mutable field* is generated by first pruning values whose frequencies are more than two orders of magnitude less than median frequency. Less frequent elements of the pruned list are then removed until the length of the list is a power of two; that list is the *symbol table* for that *mutable field*. To our knowledge, this is a unique approach for disguising

data in a circumvention system; we believe similar approaches could be adopted by other circumvention systems to improve their undetectability.

When Rook attempts to send data by altering a *mutable field*, rather than writing k-bits of its secret data to the field, it instead converts n-bits of secret data into a symbol using the *symbol table* for that *mutable field*, where $n = log_2(length\ of\ symbol\ table) <= k$. By using a *symbol table* generated from normal game traffic Rook avoids sending values that are never or very infrequently seen and so prevents an adversary from filtering traffic based on these.

When the altered packet is received by the other Rook user, the process is reversed to translate from symbols in *mutable fields* to secret data bits, using the same *symbol table* in reverse. Because of this, it is required that both sides possess a copy of the same *symbol table*.

In earlier iterations of Rook a static symbol table was used, created from a long gameplay packet capture. However, the n-gram analysis described in Section 2.4 showed a flaw in this scheme: the values of some *mutable fields* are dependent upon client configurations (such as graphics settings) and thus created anomalous values in Rook datasets when Rook was used under different settings because the values in the *symbol table* never appeared naturally. Instead, Rook now generates a *symbol table* dynamically when the client first connects to the server. This way, all values sent using Rook represent values not just possible for the game, but already naturally occurring between client and server in a given session. Althogh not shown to be necessary by our analyses, we may add dynamically updating symbol tables to future versions of Rook, as discussed in Section 2.6.

2.2.7 Scheduling Altered Packets

Even though these network protocols are designed to tolerate poor network conditions, gameplay does degrade if too many packets are dropped in a row. Therefore, Rook is designed to let most packets through; by default it selects roughly one-in-ten packets to be altered. Since the values inserted by Rook are only legitimate game values, we cannot send a short flag

indicating which packet is the one that has been altered, because that flag might naturally occur without the sender having written it. Any flag long enough to be satisfyingly unlikely to occur randomly is too much overhead for the bandwidth of Rook or would be easily detectable by an attacker. Therefore the Rook sender and Rook receiver must pre-schedule which packets will be altered.

Before this schedule is arranged, the sender and receiver initially synchronize using a flag value. This flag value is derived from their shared key to prevent an attacker from either easily enumerating all Rook servers by sending them flags, or passively detecting Rook channels by looking for the initial flag value being sent. The sender picks an arbitrary packet with enough *mutable fields* to store the flag, default of 40-bits in length. The sender then synchronizes their sending schedule based on the sequence number of the packet they put the initial flag in. The receiver scans received packets until they see this flag; when they receive the flag the receiver also synchronizes its receiving schedule based on that packet's sequence number. In this way, both the sender and receiver have a synchronized schedule for which packets will be altered by Rook. In our implementation, the server performs this receiver-side scanning for all clients for the first 5 minutes of their connection and then assumes they are not potential Rook clients until they reconnect again.

2.2.8 Shared Deterministic Cryptographic Random Number Generator (DCRNG)

This solution by itself is not necessarily sufficient because of the regularity of packets being altered. If the interval between altered packets is a constant value, an attacker could launch statistical attacks to compare the values of sets of packets and would likely easily be able to determine the difference in a sequence of packets that have all been altered versus a sequence of completely unaltered packets.

To avoid this, the sender and receiver use a shared deterministic cryptographic random number generator (DCRNG) with a seed derived from their shared secret key. This DCRNG is used to produce a set of sequence numbers to use as the shared schedule for which packets to alter.

2.2.9 Shared Keys

As the previous two sections show, the Rook client and server need to share a secret key from which they derive the expected initial flags and DCRNG seeds. This key exchange must be performed out-of-band once before the first Rook connection is made. After the first connection is established, however, the client and server can easily negotiate a new shared key. Furthermore, to ensure forward secrecy, they can share this key via a standard Diffie-Hellman key exchange. The method used for the initial out-of-band exchange is out of scope of this work. To further help prevent key compromise from affecting multiple clients, the Rook server is expected to keep a list of shared keys, and distribute a different key to each Rook client.

2.2.10 Message-Present Bits

A disadvantage of the scheduled sending scheme is that the receiver must assume the scheduled packet has been altered, and so must assume any values within its *mutable fields* represent secret data. To prevent the receiver from erroneously interpreting these values when the sender actually simply had nothing to send, the Rook system uses a message-present bit in each altered packet. To help avoid detection the *mutable field* this message-present bit is inserted into is randomly determined using the same synchronized DCRNG described above.

When a sender inserts data into a packet, they insert a 1 for the value of the message-present bit, translated into a value using the *symbol table* in the same way all other data is, indicating data is present. If the sender has no data to insert, they insert a 0 for the message-present bit, and no additional data.

When the receiver attempts to extract hidden data from the packet, they first check the *mutable field* containing the message-present bit. If it is a 1 then they receive the rest normally, if it is a 0 then the receiver stops there and does not attempt to retrieve any hidden data.

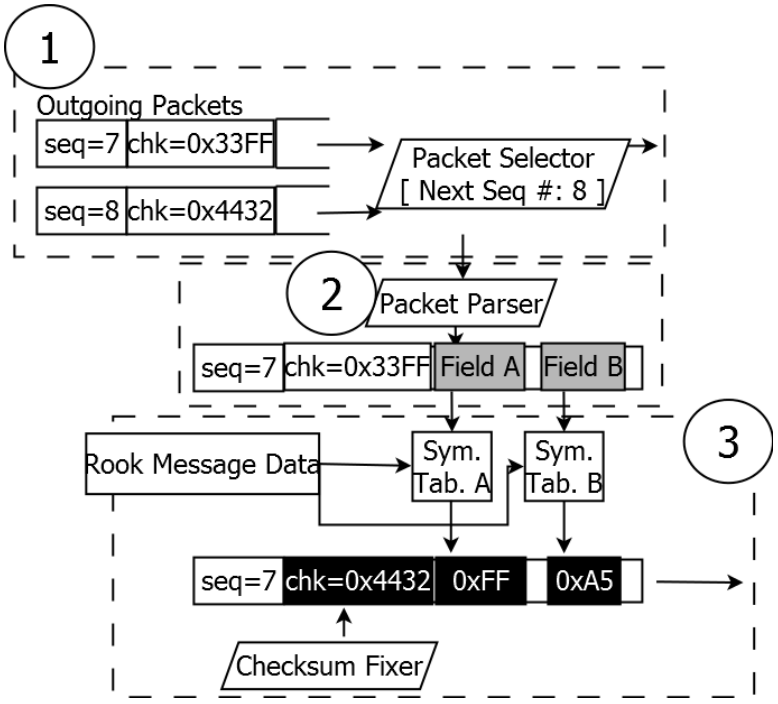

Figure 2.2: Rook Sending Process: (1) Rook selects a packet to alter based on sequence #. (2) Rook parses the selected packet to find *mutable fields*. (3) Rook overwrites *mutable field* values based on the data to be sent and the *symbol table* for that *field*. Then the checksum is recomputed and the packet sent.

2.2.11 Packet Loss Resilience

The intention of Rook is to serve as a reliable communication channel. However, it is using a UDP network protocol built to be robust to missing data rather than retransmitting lost packets. Therefore, Rook must include its own layer of reliable delivery to prevent either naturally poor network conditions or an active adversary from easily disrupting the communications.

Rook uses a simple sequence-number/sequence-number-ack header prepended to the message data. Because all the game protocols so-far observed contain an incrementing sequence-number in the payload of the packet, the Rook receiver can easily determine if a game packet containing Rook data was dropped. In this case, the receiver simply does not increment its

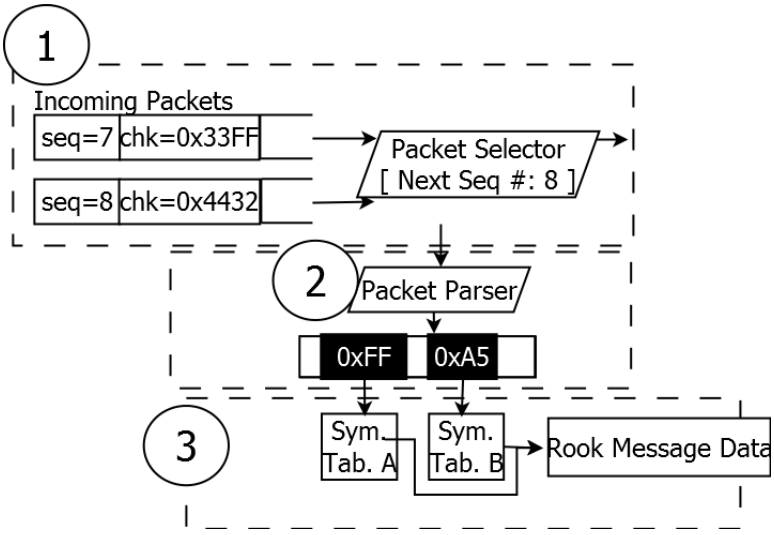

Figure 2.3: Rook Receiving Process: (1) Rook selects the altered packet based on sequence #. (2) Rook parses the selected packet to find which bits are in *mutable fields*. (3) Rook uses the *symbol tables* corresponding to the *fields* to convert their values back into bits of data.

sequence-number-ack value and the original sender will understand it needs to retransmit. This is fundamentally the same mechanism used for reliable delivery in TCP, but Rook takes advantage of the game already providing sequence numbers to avoid that overhead.

Keeping the DCRNG synchronized through a packet loss event is an additional complexity. The solution is for the Rook system to always generate a fixed number of random numbers per-packet to use for packet processing (such as determining which *mutable field* is the message-present). When the receiver observes a packet drop based on the missing game packet sequence number, it simply generates the fixed number of random numbers just as if it had received the dropped packet. While the packet data is lost, the DCRNGs of the sender and receiver stay synchronized so the data can be retransmitted. In this way, Rook can be robust against packet-dropping attacks that are not large enought to cause the game connection to be broken (typically 30 seconds or less).

2.2.12 Client-Server Connection Setup

A Rook connection consists of two active channels connecting a Rook client and Rook server to allow data to covertly flow in either direction. The Rook system does not assume these channels are established as soon as the game connection is established, a Rook server also does not have any ability to recognize a Rook client except by data sent over the Rook channel. Therefore, a special connection handshake is needed to establish the two Rook channels to connect the Rook client and Rook server. The Rook server contains a list of secret keys, S, that are used as the shared keys to facilitate the covert channel. A given Rook client only uses one secret key, S_i to connect to a given server. For simplicity, the following description of the connection handshake is described as if the Rook server also only had a single key, S_i.

Initialization. Both the Rook server and client use their shared secret key S_i to seed their deterministic random number generators ($DCRNG_S$). The $DCRNG_S$ is then used to generate a pair of flag values, (F_C, F_S), and two lists of indices, $(I_{C,0}, ..., I_{C,n}, I_{S,0}, ..., I_{S,n})$. The client and server also each generate an AES key from output of the $DCRNG_S$ and store the IVs to transmit later.

The server inspects packets from each game client that is not already a Rook client. It performs essentially the same procedure as a receiver, except there are *message-present bits*, and it only checks specific *mutable fields* based on pseudorandom indices derived from the shared key (see above).

Client Sends Flag. When the Rook client wishes to connect, they begin filtering outgoing packets to find one with enough free bits to fit the flag value F_C. They then embed F_C into the payload using a slightly modified version of the sending algorithm above: there are no *message-present bits*. The client then uses this packet's application-level sequence number to compute its *next_packet* value to know when to send next, and begins sending its IV.

The client also begins listening for F_S in incoming packets as described above.

Server Finds Flag. When the flag value F_C is detected the Rook server immediately

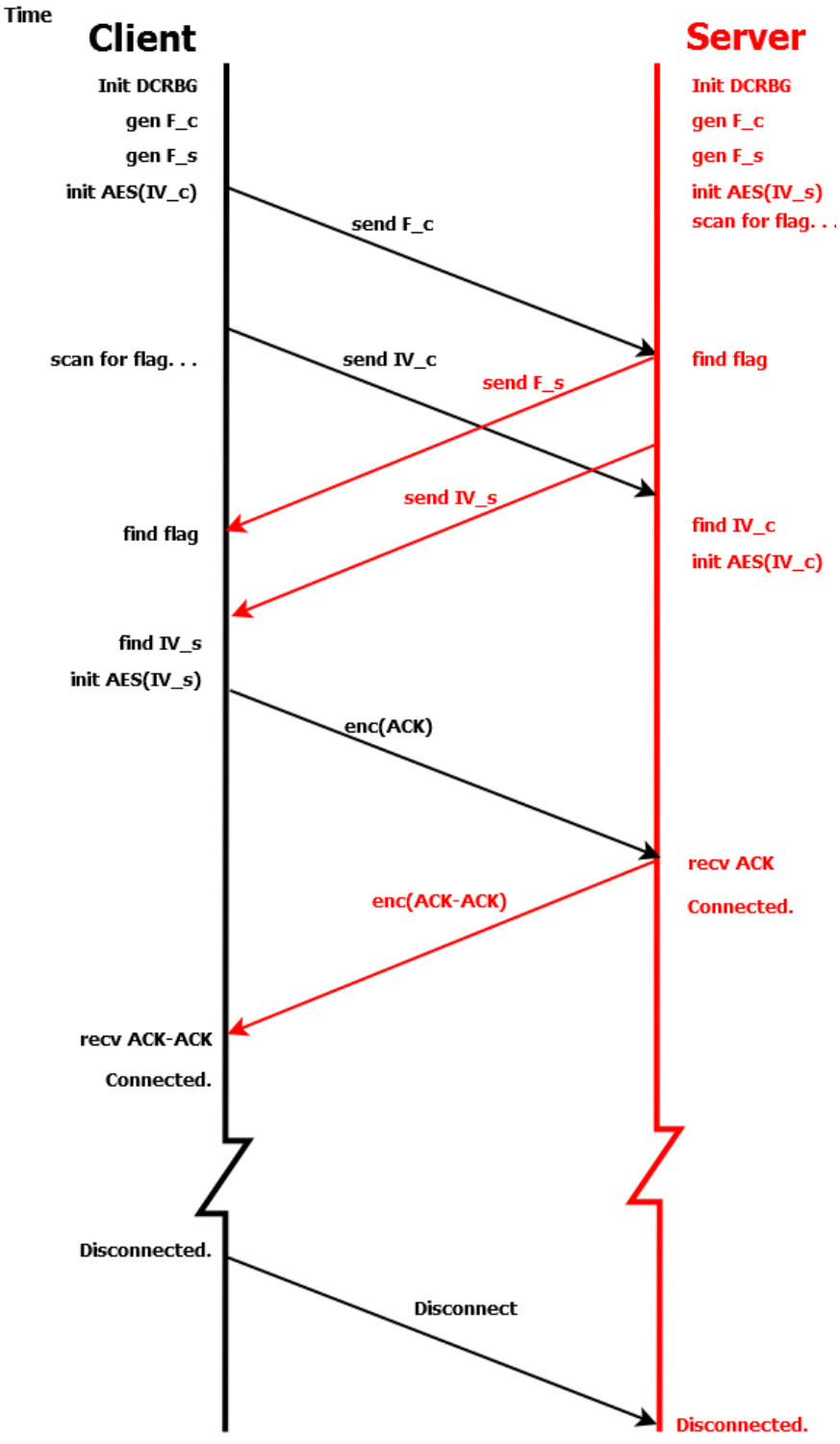

Figure 2.4: Rook Connection Handshake

calculates the *next_packet* of its receiver based on this packet's application-level sequence number, just as the Rook client did when it embedded the flag. The client sender and server receiver are now synchronized, because their *next_packet* values are the same and their $DCRNG_S$ are synchronized to output the same random values. The Rook server now begins receiving the client IV.

Additionally, when the flag is received, the Rook server begins attempting to embed F_S into outgoing packets in the same manner the client embedded F_C into its outgoing packets (described above), followed by the server's IV.

Server Receives IV. When the server receives the client's IV, the server initializes its decryptor with the shared key and client's IV. The server then waits to receive an encrypted "ACK" message from the client. When this is received, it considers both Rook channels to be working and the connection established, and sends its own encrypted "ACK-ACK" message to the client

Client Receives Flag and IV. When the client detects F_S it synchronizes its receiver by generating the *next_packet* value. As before when the server received F_C, upon receiving F_S the client receiver and server sender are now synchronized.

The client now initializes its decryptor with the server's IV and shared secret key from subsequent messages. When it has received the entire IV, it generates the a decryptor for the server's messages. Furthermore, the client also encrypts an "ACK" message and sends it to the server.

Client Receives ACK. After sending its encrypted "ACK", the client waits for the server to send back an encrypted "ACK-ACK". When this is received the client considers both Rook channels to be working and the connection to be established.

2.2.13 Client-Client Connection Setup

At this point, the client is connected to the server, but is not communicating with other clients. Fundamentally, these connections simply consist of the Rook server forwarding data

between the two clients. Depending on the preferences of the server, it could list all clients connected to it and allow them to establish conversations with each other (as with IRC), or it could hide this information and force clients to already know something like a username or a secret key to try to message a particular client. When two clients establish connections to one another they have the option to establish an Off-The-Record conversation to ensure confidentiality from the Rook server.

2.3 Implementation

The preceding section described the design of the Rook system, which could be applied to any games with certain network characteristics. To study and evaluate the effectiveness of this system we implemented it for the game Team Fortress 2. Because Team Fortress 2 is built on the Source Engine, it shares the same network stack as other popular Source Engine games including Counter Strike: Global Offensive, and Day of Defeat: Source. Therefore, our implementation can also function for these, although the symbol tables would need to be regenerated to assure they still contain reasonable values for the specific game. All the testing and evaluation was done using Team Fortress 2.

The main challenge to implementing Rook for a given game is reconstructing the packet format for the game. Both sides of the Rook connection must be able to parse the packet enough to correctly identify bits that can be altered without causing the packet to provoke errors from the real game packet parser. In practice this means the Rook code needs to be able to successfully parse the most common types of packets and correctly identify any others to be safely ignored. A description of what we determined about the Team Fortress 2 packet structure can be found in Appendix A.

2.3.1 Basic Packet Structure

We determined the packet structure of a Source Engine packet sent from both the server and client to be as follows:

- 32-bit Sequence Number - increments on each packet sent, used to determine ordering and duplicate packets
- 32-bit Sequence Ack Number - the sequence number of the last packet received
- 8-bits of Flags
- 16-bits of checksum computed over the rest of the packet - uses CRC32 truncated to 16-bits
- 8-bits of reliable state informaton about 8 subchannels
- If one of the flags is set, then 8-bits are read containing information about choked packets
- Series of net-messages, a 6-bit message type followed by n-bits of message data

The difference in packets between the client and server are the types of net-messages they send, described below.

2.3.2 Client Messages

The client mainly sends two types of messages: *tick* and *usercmd*.

Tick

The *tick* message is simply 48 bits of data. It appears to be necessary for keeping the client and server synched correctly and uses an incrementing value so it is not a good *mutable field* to use.

Usercmd

The *usercmd* message is a message of potentially 14 fields containing information about user commands. Each field is preceded by a 1-bit flag indicating whether that field has been provided or not. These fields represent all the in-game actions a player's avatar can take such as switching weapons, turning, and moving in different directions. In all, the messsage can be anywhere from 13 to 359 bits. Multiple of these messages can be found in a single

packet, if the user has issued multiple commands since the last packet was sent. The majority of the fields are based on the user actions, and so are good *mutable fields* for Rook to use. Some of these fields end up being poor candidates due to low entropy: the weapon switch field will typically only have three different values possible at any given time.

2.3.3 Server Messages

The server also mainly sends two types of messages: *tick* (as above) and *PacketEntity*.

PacketEntity

The *PacketEntity* message contains updates for the entities, the other players and moveable-objects, on the server. Each entity has a number of properties depending on its class. Each of these properties may or may not be updated when the entity is updated. Furthermore, each property is parsed differently depending on its type. Rook primarily modifies character facing and position fields as these appear to be the most frequently changing.

2.4 Evaluation

2.4.1 Bandwidth and Usability

Our implementation of Rook for Team Fortress 2 currently operates at 34 bits/second from game client to game server, and 26 bits/second from game server to game client. This is relatively low but still useful for the real-time chat messaging that is the target of this system. As part of our evaluation, we also incorporated an open-source implementation of the Off-The-Record [12] chat protocol to communicate with other Rook clients connected to the same server. The main overhead of the OTR protocol is in the initial key exchange, which can take several minutes in the current Rook implementation. However, after the initial connection is made the secure messages between clients are not significantly slower than unencrypted Rook messages.

Rook use also did not significantly impact the gameplay of the user or trigger any warnings

from the built-in Valve Anti-Cheat system. We believe a server of Rook users and non-users could play together without any obvious differences noticeable between the two. However, Rook is not designed to protect against a human attacker individually targeting a specific Rook server. All FPS games studied also allowed password-protected servers, which could prevent an attacker from being able to join and observe a Rook server.

2.4.2 Surveillance Resistance

To evaluate the surveillance resistance of Rook we classify the types of attacks we consider into six types:

1. Anti-Mimicry
2. Single-Packet Deep-Packet-Inspection
3. Traffic Shape Analysis
4. Statistical Multipacket Deep-Packet-Inspection
5. Game-Specific n-gram Analysis]
6. Machine-Learned Classifier

Anti-Mimicry

Anti-mimicry attacks are defined as attempts to probe and identify Rook servers or clients based on comparing their response to probes to the response of a normal game server or client. It has been previously shown [38, 54] that many anti-surveillance and anti-censorship systems are vulnerable to these types of attacks. Rook is not vulnerable to these types of attack because it is not mimicking the game client and server but actually running them on both ends.

Altering packets could be used to create a denial of service attack against Rook. The attacker could replace mutable field values with other values to corrupt the data Rook receives. The attacker would need to do this to all game packets since they do not know which packets are scheduled to be read by Rook. Thus the attacker would still impact

legitimate players by randomly changing their commands and server updates. Randomly dropping packets could also be used as a denial of service attack. The attacker would need to drop few enough to not degrade play experience for legitimate players while also dropping enough to consistently drop a Rook-altered packet by chance and disrupt the messages. Rook could potentially respond to this type of attack by sending redundant messages although this would further lower its bandwidth.

Stateless Deep-Packet-Inspection

Stateless Deep-Packet-Inspection (DPI) is what many censors appear to currently use if they do anything more advanced than IP or port blocking [127]. Stateless DPI is not effective against Rook because all packets altered by Rook adhere to the packet specification for the game and only use previously observed game values in the alterations. Therefore, if a stateless DPI system detected a Rook altered packet as malicious, it would necessarily have to also falsely detect many game packets of legitimate players as malicious.

2.4.3 Statistics-Based Attacks

While more costly to deploy, and hence less likely, we now consider attempting to detect Rook using multipacket statistical analyses to compare normal game traffic to game traffic altered by the use of Rook.

Data Gathering Methodology

To evaluate the traffic shape analysis and the statistical DPI attacks discussed below, we created datasets as follows: a TF2 server with 20 bots is run on one machine on the LAN. This machine also runs the Rook server in the datasets using Rook. A second machine on the LAN runs a TF2 client, connects to the server, and runs the Rook client.

There are no universal standards for evaluating circumvention systems, owing in part to the diversity of their methods. We showed above how Rook is secure against past methods of

attack on some circumvention systems. However, there is a large space of possible statistical attacks to try. To show the efficacy of our attacks, we also gathered samples a high-bandwidth (HB) configuration of Rook. It functions in the exact same manner as Rook, except it replaces every 1-in-2 packets, while normal Rook replaces approximately every 1-in-10. This allows us to demonstrate a particular statistical attack is capable of detecting systems like Rook, even if it cannot detect Rook being run in its typical configuration.

For datasets using Rook the user connects to the TF2 server, both the Rook server and Rook clients observe 600 packets (approximately 60 seconds of gameplay) in each direction and then create their symbol tables. The Rook client then connects to the Rook server and each side sends pseudo-random data at the maximum data-rate. The packets sent and received after the Rook connection is made are captured using Wireshark for analysis as described below. Each capture is approximately 7,000 client-to-server packets, and 6,000 server-to-client packets (about 5 minutes of gameplay/actual Rook use).

We gathered 61 datasets: 20 using Rook in HB configuration; 20 in the typical configuration; 20 of normal gameplay; and one of normal gameplay that was used as a baseline for investigating the three datasets listed above. The baseline is used to represent a known-normal dataset an attacker could use to try to differentiate normal and Rook traffic by comparing them to the baseline.

During the course of experimental setup, we observed that many unexpected factors can affect the values sent in the game packet payloads; these appear to include: operating system, graphics hardware, and game window resolution. To give our attacks the best reasonable environment, we attempted to hold as many variables in the experiment stable as possible. In addition to using the same number of bots in all cases, the game player was also the same and already had experience in the game, the class played was the same, the game level played on was the same, the level was restarted before each sample, and data was continually sent at the maximum bandwidth. These essentially reflect the best possible circumstances an adversary could be expected to capture traffic under, as there should be as little variance as possible between samples and the covert channel's bandwidth use is maximized.

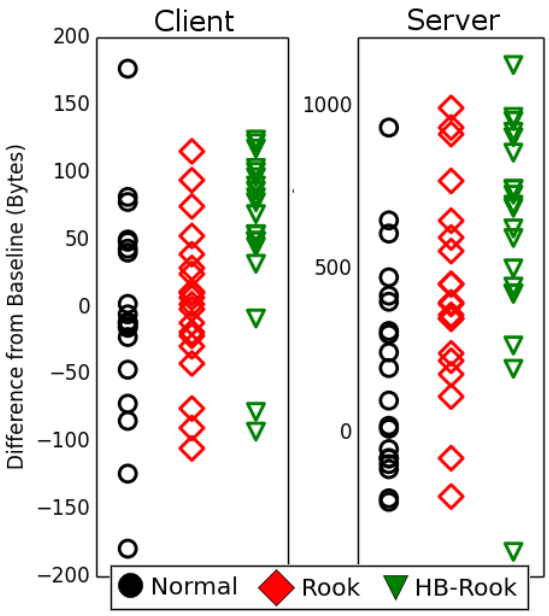

Figure 2.5: The difference in bandwidth consumed from the baseline traffic. HB-Rook appears to average slightly higher than normal.

The above efforts to control the environment for more consistent analysis contributed to the dataset size, since the generation could not be automated or parallelized. However, this size appears reasonable for evaluation since most of the samples show strong similarities with only a few outliers.

Traffic Shape Analysis

Traffic shape analysis is conducting statistical analyses on the size and timing features of the traffic between the client and server. Some previous covert channels have used timing changes to send secret information, and some have simply injected extra bytes into application packets, e.g., [40, 78]. These approaches have the potential to be detected by comparing statistics between known normal traffic and suspicious traffic, e.g., average size of packets or median timing between packets [61].

Rook should not be vulnerable to these approaches because it does not alter the timing or

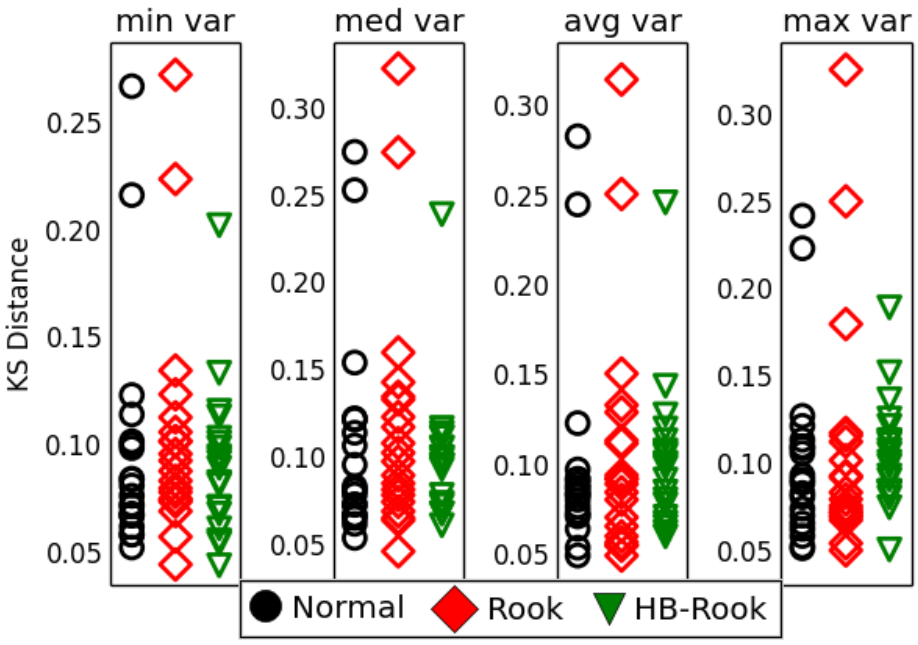

Figure 2.6: Results of the KS test performed on the distributions of variance across byte positions in client packets of the same size. There are a few outliers in all three categories, but no clearly distinguishing features.

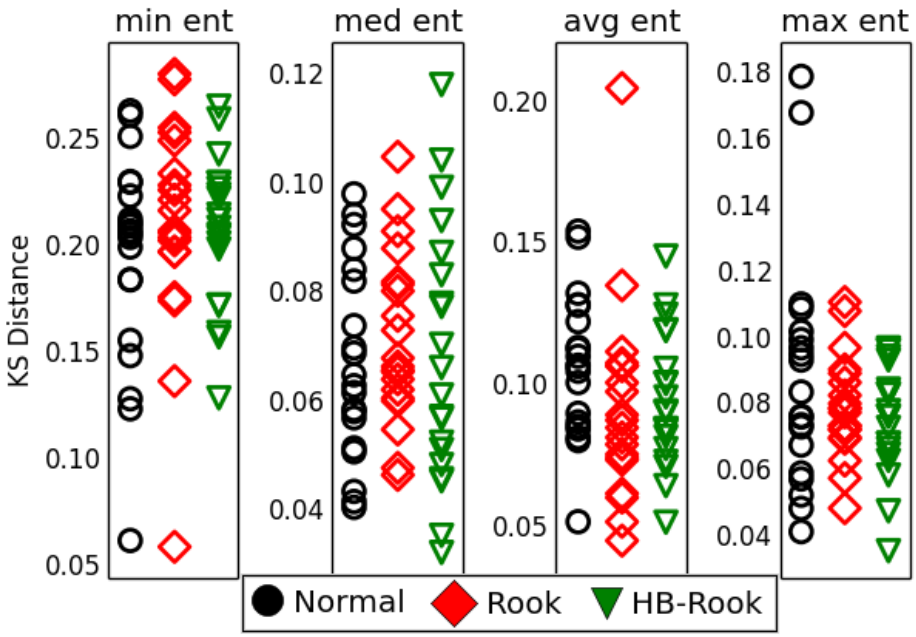

Figure 2.7: Results of the KS test performed on the distributions of entropy computed across each packet for client packets of the same size. No feature where Rook or HB-Rook are clearly distinguishable from all normal samples.

length of game packets to embed its information, it only alters individual data-fields within the packet. Furthermore, using Rook does not cause any additional packets to be sent, or packets to be changed in size, by the game server or client, so the traffic shape should be unaffected.

To evaluate this attack we extracted the overall bandwidth and spacings of packets and compared normal traffic to Rook-altered traffic using the 2-sample Kolmogorov-Smirnov (KS-2S) test. We chose this test for its simplicity and use in previous system evaluations [53, 55, 61, 72]. The results show (see Fig. 2.5) that both typical Rook and HB-Rook use is difficult to distinguish from normal game traffic. We are not certain of the cause of the slightly higher average bandwidth for HB-Rook: it may be due to Rook causing more packets with significant updates to be sent, reducing the effectiveness of the delta compression the game uses.

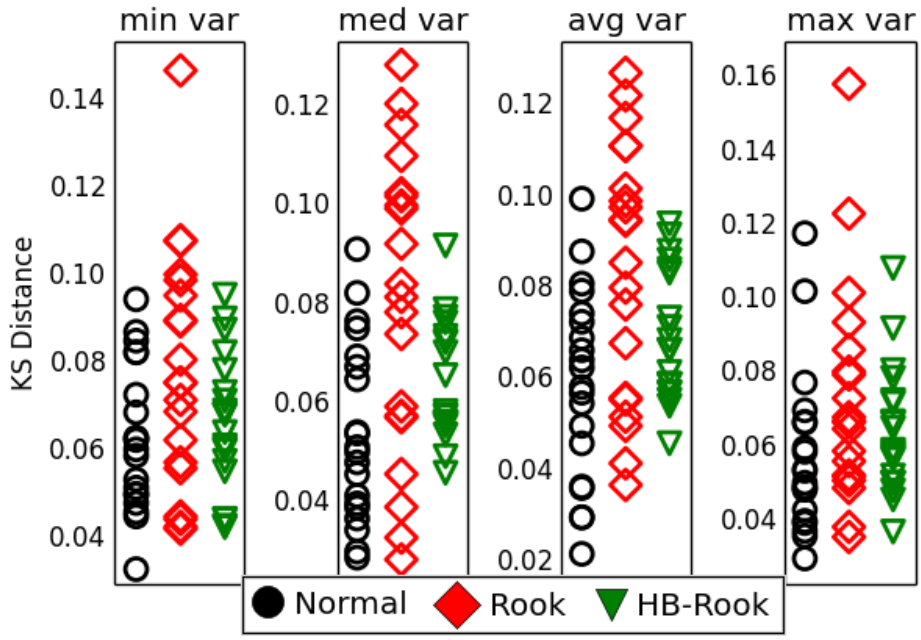

Figure 2.8: Results of the KS test performed on the distributions of variance across byte positions in server packets of the same size. Slightly lower average distances for the Rook traffic.

Statistical Multipacket Deep-Packet-Inspection

Statistical multipacket DPI is the monitor taking a packet capture of all traffic between the client and server and running statistical tests across the payloads to compare these results to those obtained from doing the same process to traffic from a known normal game. There are many possible tests one could do, and no standard for using this kind of approach to evaluate a channel of Rook's kind. As a starting point, we adapt methods used in the related area of covert timing channels to test Rook's detectability. In these systems the timing of packets is the information channel and therefore statistical tests are run on the distributions of packet timings (similarly to the traffic shape analysis above) [53]. The analogous channel in Rook is the packet payloads, so we run the same tests on statistics computed over the payloads to detect anomalies.

Since only previously observed values are used by Rook, the only potentially detectable

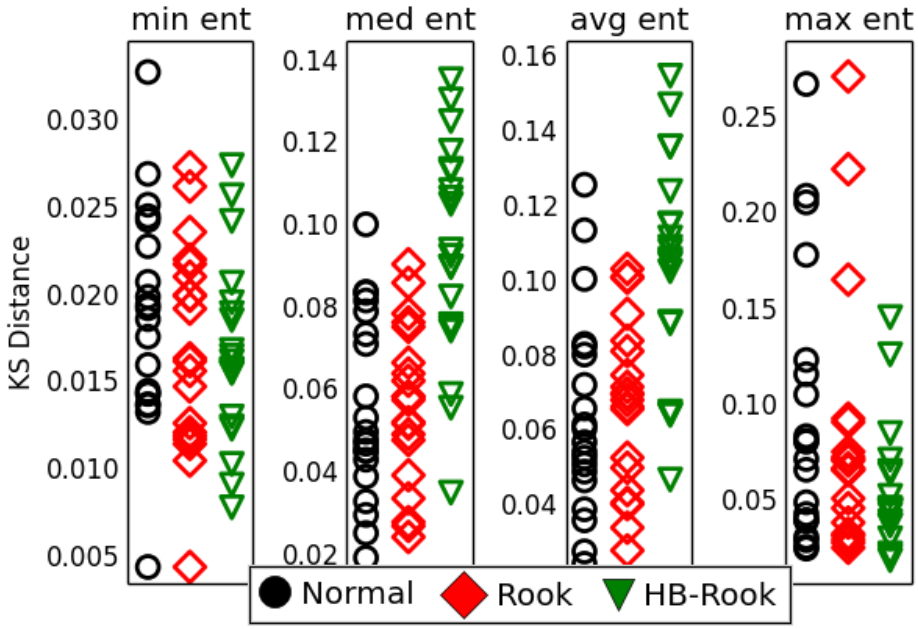

Figure 2.9: Results of the KS test performed on the distributions of entropy computed across each packet for server packets of the same size.

difference between Rook and normal payloads is differences in the distributions of these values. Therefore, we measure the variance and entropy across each data-field in Rook versus normal traffic. We measure the variance by first grouping packets by size and then computing the variance across all these packets at the same byte position. This process is repeated for each byte position, and for each size of packet captured. Entropy was measured by computing the entropy of each packet for all packets of the same size. The minimum, median, average, and maximum of each of these two statistics for each packet size was extracted to form eight distributions to test. The distributions of all of these results were compared to the same distributions derived from the baseline game traffic, and compared using the KS-2S test.

The resulting graphs can be seen in Figs. 2.6, 2.7, 2.8, and 2.9. These are each the stated statistic gathered from the sample traffic compared to the same statistic gathered from a baseline normal traffic sample using the KS-2S test. These show relatively little difference

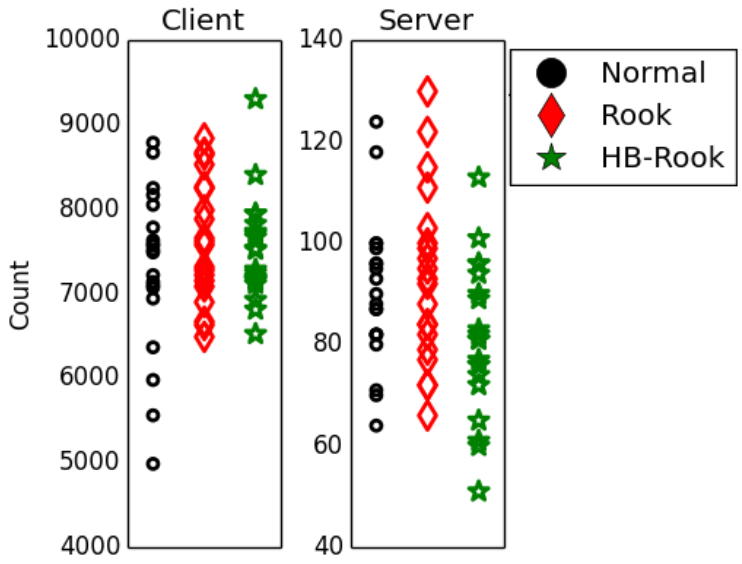

Figure 2.10: Sample of counts of distinct unigrams for one client field and one server field.

between the three categories. In Fig. 2.8 there is a slightly lower average distance for the normal Rook traffic; this is somewhat odd since the lower KS distance means the Rook samples are more similar to the baseline normal sample than the other normal samples are.

Game-Specific n-gram Analysis

The final analysis we conducted was a game-specific n-gram analysis: this is an analysis of the values observed in the mutable fields of our implementation of Rook for Team Fortress 2. Using the same packet parsing module, we constructed lists of unigrams, bigrams, and trigrams for each mutable field for both the client and server. This analysis is similar to the analysis of variable-bit-rate encoding in VoIP used to distinguish languages spoken without decrypting the data-stream [128]. It is important to note that for any of the following attacks, the censor must have a game-specific implementation of data gathering and analysis tools, not general-purpose statistics like those used above. This would require an adversary to build a parsing engine and potentially manually generate a list of the useful features for

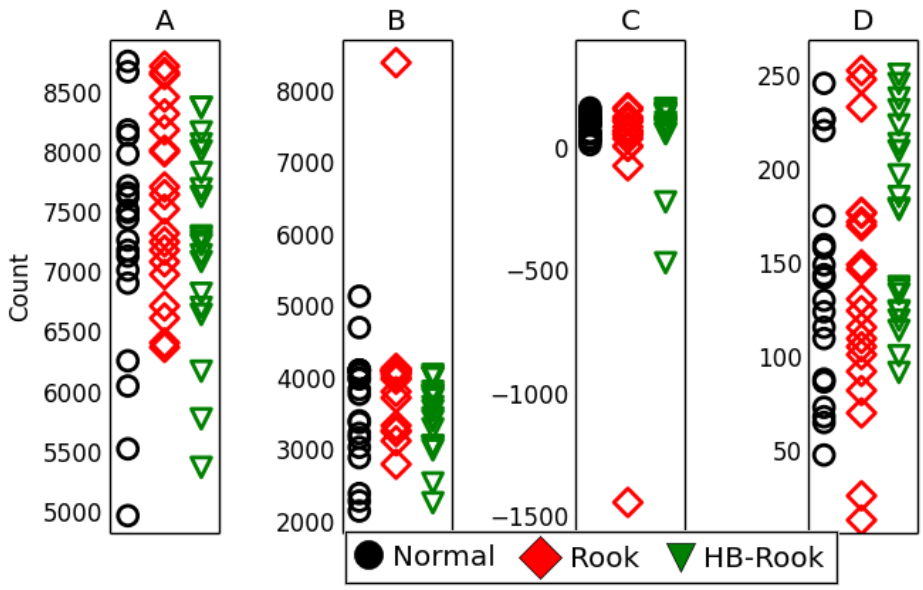

Figure 2.11: Adjusted counts of distinct trigrams from client traffic. A few outliers for both HB-Rook and Rook.

detecting a specific game. Further, for the actual use the adversary must do a full capture of the traffic and, if blocking is desired, parse and analyze it in relative real-time.

In the following section we show and discuss the results of analyzing solely the unigram and trigram data. Analysis of the bigram data showed the same results as trigram data and is excluded for brevity. The labels A, B, C, and D. on the graphs are references to different mutable fields Rook uses. They represent actual game information fields, but are relabeled for simplicity.

Count of Distinct Trigrams. The first analysis we performed was a comparison of the number of distinct trigrams in each sample. An n-gram is distinct if Rook only uses values for mutable fields that have been observed in normal game data, so the number of distinct unigrams is never increased by running Rook. The count of unigrams in both normal and Rook samples can vary, which leads to correspondingly amplified variation in trigram counts (see Fig. 2.10). To help correct for this, we computed an approximation of the function of unigrams-to-trigrams and used this function to baseline each trigram count sample before

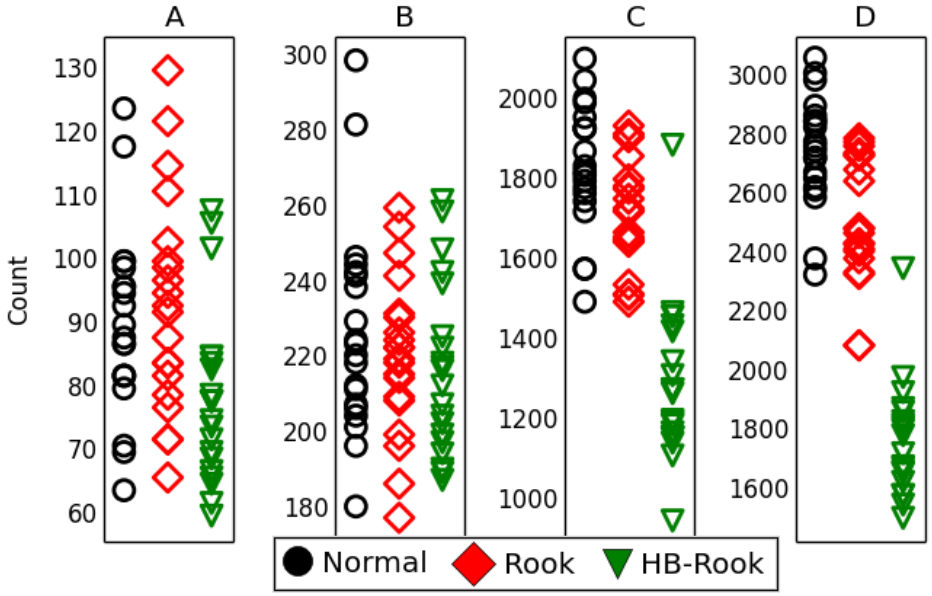

Figure 2.12: Adjusted counts of distinct trigrams from server traffic. HB-Rook is clearly distinguished over several fields from both normal and Rook samples.

analyzing. The results in Figs. 2.11, 2.12 show that there are a few outliers, but most of the normal samples and typical Rook samples are very similar. The HB-Rook samples, however, are clearly distinguishable in the server trigram counts. We believe this is because the server will be replacing so many mutable fields that would normally contain new values with previously seen values stored in its symbol table that the total number of distinct trigrams is significantly reduced. A detector could potentially be developed to detect the normal Rook traffic in the same way, but it would either have a very high false-negative or false-positive rate. Additionally, we provide mitigating solutions to this attack (if it proved practical) in Section 2.6.

Frequency Distribution. We also performed a comparison of the frequency distribution of trigrams. If Rook causes unlikely values to occur more frequently then there could be a shift to a more uniform frequency distribution versus the normal traffic. Figs. 2.13 and 2.14 show the results of a KS-2S test on frequency distribution across different mutable fields.

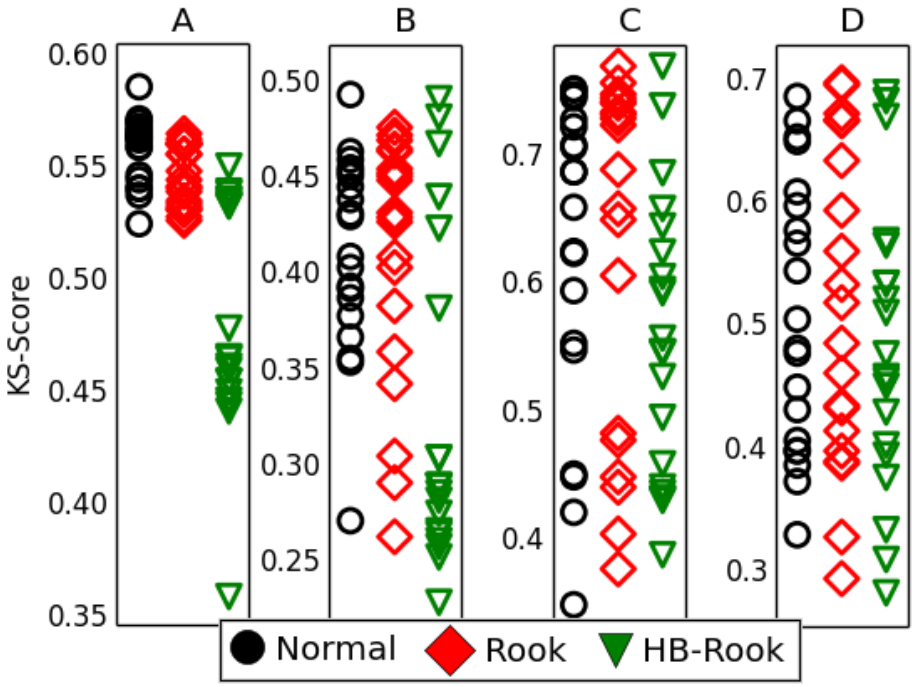

Figure 2.13: Results of the KS test performed on the frequency distributions of client trigrams. One field where HB-Rook is fairly distinguishable but due to being more similar to the baseline sample than normal or Rook samples.

For most fields the KS-distances are intermixed, showing poor distinguishability. In a few cases Rook or HB-Rook appear distinguishable by a smaller KS-distance than the normal captures, indicating they are more similar to the baseline-normal traffic. We are not able to explain this difference since the normal samples and the baseline sample were created in the exact same way. Given the small differences in KS-distance this could just be a statistical anomaly arising from our sample sizes.

Single-Frame Anomalies. In the case of client-commands, the client could send repeats of the same command as a result of normal gameplay (for example, holding the forward button to continue moving forward). If such a repeating field was a mutable field in our Rook implementation then Rook could inject a different value into the field for a single-frame in the middle of a run of repeating values. This would create what we call a single-frame

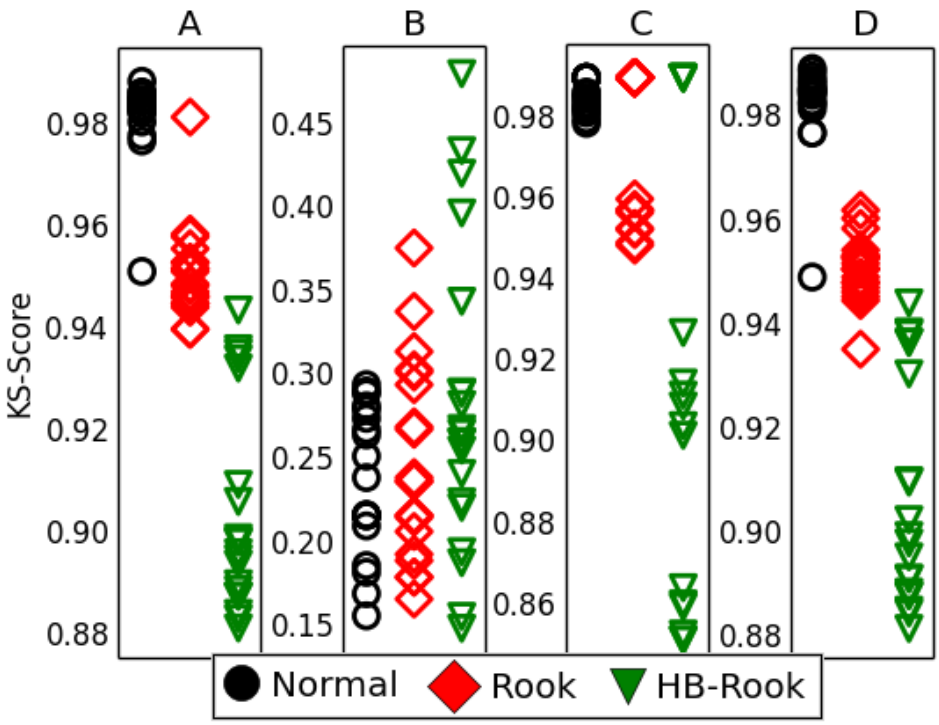

Figure 2.14: Results of the KS test performed on the frequency distributions of server trigrams. Two fields where Rook is distinguishable as being more similar to the baseline than either normal or HB-Rook samples.

anomaly, where the first and third values of a trigram are the same value, but the second is different. Our analysis shows no consistent difference between normal gameplay and Rook use (see Fig. 2.15). We do not show this analysis for the server's packets because there are no repeating sequences.

Machine Learned Classifier

None of the individual statistical analyses provided a perfect method for detecting Rook use. However, some of the analyses showed promise, particularly for identifying HB-Rook. Therefore, we hypothesized that we could use machine learning to build a classifier based on a linear combination of these tests that might have more success against Rook.

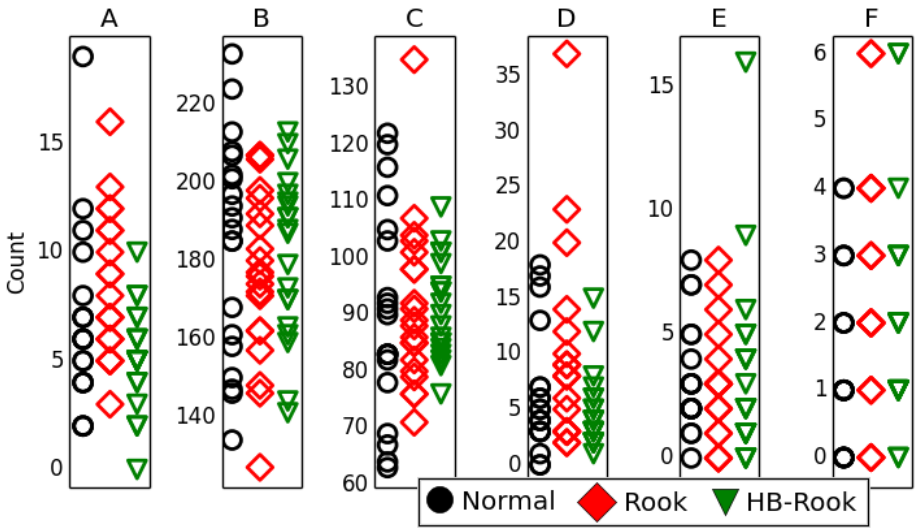

Figure 2.15: Count of Single-Frame Anomalies in client samples. Showing a few outliers from Rook and HB-Rook but overall not very distinguishable from normal samples.

KS-Test Classifier. To perform this analysis we split our datasets into equal-sized test and training samples. Each sample consisted of a vector combining the traffic shape and the KS test results for unigrams, trigrams, and overall packet characteristics. We then generated classifiers using a variety of models available in the SciKit Learn library [99]. In evaluating these models we were most-interested in the specificity (portion of Rook samples we detect) and fall-out (portion of normal gameplay samples falsely classified as Rook): the first metric is useful for understanding how likely we will actually detect users of Rook, but the second is arguably more important, because we assume in the wild there will be orders-of-magnitude more normal game players, so a high false positive rate would be prohibitive to practical use.

The best classifier type for both Rook and HB-Rook was the Decision Tree Classifier [113]. In the case of HB-Rook the classifier had a specificity of 90%, and never falsely classified normal traffic as Rook use. It relied on just two types of data: the KS distance of the trigrams of mouse x-axis movement sent from client to server, and the KS distance of the trigrams of positions of one of the avatars sent from server to client. This reinforces suspicion from our initial statistical analysis that HB-Rook is not secure against a sophisticated adversary.

When attacking the more conservative version of Rook our classifier had less success. The Decision Tree Classifier only identified 50% of the Rook samples, furthermore, it misclassified 10% of normal samples as Rook samples. This is particularly a problem in the expected use-case, where normal game players are orders of magnitude more common than Rook users. However, under narrower circumstances where a particualr individual is suspected for Rook use for other reasons, this classifier may be useful.

Raw Field Classifier. We also analyzed creating a classifier from just the raw data about field values extracted from traffic, rather than computing KS tests over the distributions first. To keep the size of input data reasonable, we imposed a "window-size" on the amount of data that would be in a single sample. This window-size reflects the number of packets in each sample, so a window-size of 10 means all the mutable field values of 10 consecutive packets made up the input vector. We generated classifiers using the models provided in the SciKit Learn library [99] and using window sizes between 10 and 1000.

Compared to the KS-test based classifiers, above, we had little success, even on HB-Rook. The most successful classifiers were Multilayer Perceptrons: the model with the best precision was based on just a 10-sample window-size, and a fallout of only 0.09%, but also only a specificity of 0.82%. A window-size 100 classifier was more successful at finding Rook, with a specificity of 42.9%, but also a 29.7% fall-out, significantly worse than the KS-test based classifiers. The classifiers trained on the stealthy version of Rook did not achieve even these levels of success.

2.4.4 Summary of Evaluation

In summary, we evaluated the security of Rook against six different major attack vectors. It is robust against standard anti-mimicry, DPI, traffic shape analysis, and generalized statistical attacks. Further, it is resilient against game-specific statistical attacks but may be vulnerable to a focused adversary devoting resources to detecting use of Rook on a specific game leveraging full packet captures and machine-learning to generate classifiers. We be-

lieve this represents an improvement in the state-of-the-art in this field and still meets a high level of security since the resources to mount an attack are at the upper bound of our threat model, requiring multipacket statistical analysis using game-specific knowledge.

2.5 Related Work

There have been many different approaches taken to enabling censorship circumvention or surveillance avoidance in the past. These have ranged from manipulating traffic shape attributes such as packet timing and size [40, 53], to TCP, RTP, and UDP header bits [84, 92], up to picking specific payloads such as sending specific DNS-lookups to pass information [13]. Recently there have been many systems developed which try to slip past monitors by disguising themselves as normal traffic [55, 79, 102, 124, 135].

Timing Steganography. There have been several iterations of systems based on using packet timings to covertly communicate [40, 53]. These systems are potentially harder to detect than Rook because the packet timings they are modifying are already impacted by what other processes are running on the machine, which is outside of the attack scope. However, an active adversary can potentially block the channel by adjusting the timing of packets slightly to destroy the information, or drastically reduce the bandwidth, without necessarily impacting the legitimate application use. Despite these drawbacks, a timing-based covert channel like CoCo could actually work excellently alongside Rook. The bandwidth gain would be modest, about 5-10 bits/second, but the two systems operate orthogonally to one another and so the risk of detection would merely be whichever system is the most detectable.

Header Value Steganography. Many fields in standard network protocol headers (TCP, UDP, RTP, etc.) have been found to be useful for steganographic purposes [76, 84, 92]. These include the least-significant-bits of the timestamp, padding values, initial sequence numbers, and various flags such as do-not-fragment. These covert channels have an advantage in that they are ubiquitous to all applications with any standard type of network traffic and so can

easily bypass any application filtering an adversary could put in place. However, attacks against the covertness of several of these channels have been shown [107, 108]. Because some of these attributes are not normally used, statistical attacks can detect anomalies from their increased use as covert channels.

Furthermore, these schemes are based on the application ignoring whether these fields are set or not, so an adversary can simply normalize them to deny the covert channel. Rook modifies data that is used by the application, so this type of attack degrades legitimate use (as discussed in Section 2.4).

Application Protocol Mimicry. There have been several recent systems based on transforming the appearance of traffic to evade censorship [89, 122, 124]. These and others have culminated in the development of Tor's Obfsproxy [27] which uses modules called pluggable transports to reform traffic to look like an arbitrary protocol that is believed to be uncensored or unmonitored. These types of approaches generally suffer from weakness to active probing: if either end is not actually running the application the altered traffic would be produced by, probes from adversaries will be ignored. As shown by Houmansadr et al., it is very difficult to try to accurately mimic how a real application will respond to an arbitrary probe, particularly those which would cause errors [54]. Since Rook actually runs the game client and server, an active adversary's probes will be responded to in exactly the same manner as a normal game client and server. Furthermore, depending on the application being mimicked, an adversary could attempt to determine the legitimacy of packets by running them through its own copy of the application. If the mimicry is shallow (to increase bandwidth or reduce the work required to create the channel) then these packets will probably fail this test, providing another way for an adversary to spot mimicry.

Application Subversion. Since these attacks on mimicry-based systems have been published, there have been several new systems proposed which hide data at the application-layer, rather than inserting it at the transport layer [48, 55, 79, 102, 135]. These approaches are the most similar to Rook, in that they are essentially sending correct application data. How-

ever, they require the adversary to not be able to see what that data is. Unlike Rook, all of these approaches rely on an encrypted channel between the ends of the application. Current events show that some censors will force man-in-the-middle attacks or subverted versions of programs to be used to allow breaking this encryption. Rook does not its applications traffic to be encrypted to remain secure.

Castle. On 3/19/2015 a paper by Hahn et. al. was published online that also uses games as covert channels. Their system was named Castle. On 3/20/2015, we released this as a tech report. The two works were concurrent and the similarity of naming is purely coincidental; Castle and Rook use distinctly different mechanisms and types of games to operate and also target different threat models.

2.6 Discussion and Future Work

Rook is a new approach to an established problem of censorship resistant communication. We argue that online games provide an excellent form of cover for secret communication and have enough mass appeal that a censor would be reluctant to outright block their traffic. The evaluation of our implementation of Rook for Team Fortress 2 shows it to be robust to all currently known forms of network interference and censorship. Further, the evaluation shows Rook is resistant to potential new forms of censorship, such as deep-packet statistical analyses and application-specific analyses, that may become common tools for censors in the future.

Rook also aims itself at somewhat under-represented facets of censorship resistance: establishing safe communication entirely within a censor's region of control, and emphasizing keeping plausible deniability for its users. The current implementation of Rook is functional in exchanging message undetected; however, we believe it could be expanded upon in the following ways:

The first and most obvious extensions from a utilitarian perspective is the implementation of Rook for more games. We developed the Rook code in a modular fashion so that new modules for interpreting different game packet protocols can be easily added.

From a system design perspective, a secure bootstrapping mechanism for finding and contacting Rook servers would be a significant addition to the usability of the system, and could potentially also be conducted over the same online game, but perhaps with a higher-latency mechanism. However, this is a major challenge in circumvention systems in general that has not been solved.

Another improvement that could be made to Rook is making the symbol tables dynamically self-adjusting to attempt to better preserve the traffic statistics. Essentially each side would monitor how the packets it altered have impacted a set of statistics and then modify their symbol tables to try to minimize the statistical deviance created by hiding data. This would inevitably reduce bandwidth to some extent; however, if statistical attacks are a valid and massively deployed form of detection for Rook, the tradeoff would be worthwhile to defeat such attacks.

In the space of attacks, Rook prompts several ideas of more advanced attacks potentially applicable to both itself and other application-layer circumvention systems. Markov-modeling based attacks are potential detectors of Rook use. The attacker would generate a markov-model of user behavior based on parsing large normal traffic captures for the game. The attacker would then try to detect Rook traffic by looking for play that is particularly different from the model. However, there may not exist a model that fits all normal players well enough while still showing Rook as significantly different.

2.7 Conclusion

In this chapter we presented Rook, a system designed to provide low bandwidth low latency surveillance resistant communication using the network traffic of online games. Rook represents one of the first circumvention systems to utilize online games as a cover for secret communication. Beyond its novelty, Rook represents a useful addition to the space of existing circumvention techniques and systems. Unlike many other systems, Rook focuses on providing secret communication within a surveillor's sphere-of-control, and presents greater secrecy and deniability than previous systems by virtue both of how its communication is

hidden and by it being hidden in online game traffic instead of other applications that would show more differences between legitimate users and surveillance circumventing users.

Our implementation of Rook shows that it lives up to its goal of providing bandwidth high enough for chat, including over the OTR protocol, while remaining undetectable using any mass attack methods currently known to be employed by adversarial regimes. Furthermore, Rook presents a fundamentally greater challenge to detect than what most current circumvention systems present. An adversary would have to commit resources both to develop an attack against a particular implementation of Rook, and allocate computational resources to each game connection they find suspicious in order to defeat Rook. Even under targeted attack, we argue novel attack techniques would need to be developed to definitively detect Rook communications.

We have released our code along with developer documentation to help others interested in censorship resistant communication extend Rook to operate with many more games [100].

Chapter 3

BIFOCALS: EXAMINING WEB TRACKING DEFENSES UNDER TWO LENSES OF INQUIRY

In this chapter focus shifts from government-type surveillance to the surveillance conducted by the online advertising ecosystem. In particular, the web-tracking portion of this ecosystem that seeks to catalog all the webpages a user visits. I begin by describing some background on web-tracking and defenses to it, and then relate the work my collaborators and I performed to assess the actual efficacy of a range of tracking defenses to better understand which of these defenses work and why.

3.1 Motivation and Overview

Tracking user browsing behaviors and targeting ads based on that information is a common web practice. Many users perceive this surveillance practice negatively [43, 85, 97] and are adopting defenses in response. A dizzying array of defenses are now available to users, including ad-blockers, tracking-blockers, private browsing modes, and a variety of browser settings.

These defenses use different techniques and, hence, could have different impacts on a user's trackability. For example, disabling JavaScript execution prevents tracking that relies on JavaScript, but many trackers also use fallback, non-JavaScript methods. As another example, AdBlock Plus blocks requests to blacklisted advertising and tracking domains but will do nothing to prevent tracking by a non-blacklisted domain.

The fact that different defenses employ different techniques raises the question: which approaches provide better tracking protection for users?

Previous studies have attempted to answer this question for some types of defenses [32,

83, 88, 103, 105, 125]. These studies have all evaluated tracking defenses solely by measuring the *mechanisms* of tracking — e.g., the presence of specific JavaScript snippets, analysis of cookies, or requests to domains on public blacklists of trackers. However, measuring the effects of defenses on tracking mechanisms is *only one* lens through which one can evaluate the efficacy of tracking defenses; in this chapter we propose and evaluate a second, complementary lens that, combined with the first lens, provides a more comprehensive treatment.

Evaluating Defenses via Tracking Outcomes. In this work, we argue that tracking defenses should be evaluated not only based on tracking mechanisms, but also tracking *outcomes*. Specifically, we explore the use of targeted advertisements — often a reason that websites deploy tracking in the first place — as an additional lens for evaluating tracking defenses.

Using both of these lenses provides several benefits: (1) outcomes let us evaluate tracking defenses that are fundamentally different in function; (2) mechanisms let us identify individual trackers and the amount of potential tracking that occurs; (3) combining both lets us strengthen previous results by confirming them with new types of data, while also highlighting differences that show previous mechanisms-only studies over- or underestimate the efficacy of a defense; (4) measuring both tracker and ad network actions also lets us infer back-end information-sharing relationships.

Prior work has demonstrated that web trackers employ diverse mechanisms, including setting tracking cookies or other forms of in-browser storage, executing scripts, and making (sometimes nested) third-party requests [103]. Given this complexity, we observe that any study focused only on specific mechanisms — such as how often sites contact third-parties — lets us evaluate only the efficacy of defenses that specifically act on *those* mechanisms. For example, measuring the impact of a defense by counting the number of third-parties contacted when visiting a webpage does not accurately measure the impact of a defense that prevents tracking by deleting cookies.

Additionally, we can never be sure that any set of mechanisms we choose to measure is complete, as new tracking mechanisms are continually being developed [32]. For example,

when Eckersley published the first academic work about using fingerprinting techniques for tracking, there was some evidence that these techniques were already being used in the wild [30]. Further, in their recent tracking measurement, Englehardt et al. discovered new types of fingerprinting techniques never previously discussed in academic literature [32]. In both cases any measurement based solely on well-established tracking techniques would have missed these new mechanisms of tracking.

Conversely, the presence of a targeted advertisement demonstrates that a tracker has succeeded in re-identifying a user — regardless of which mechanisms were employed to do so. Thus, if we can evaluate the efficacy of defenses at limiting such outcomes, then we can compare — head-to-head — two different defenses that each act against two different web tracking mechanisms.

As with tracking mechanisms that previous works relied solely upon, tracking outcomes also have limitations — they provide a lower-bound on when a user is tracked and do not reveal the details of how the tracking is done. Thus, we performed a combined analysis of *both* tracking mechanisms *and* outcomes to overcome the limitations of each method when used alone.

In the process, we solve a key methodological challenge: how to use ads as a metric for tracking when many defenses interfere with ads in addition to interfering with tracking — as many popular tracking defenses do (detailed in Section 3.4.2).

Evaluation of Tracking Defenses. We use our combined *outcomes* and *mechanisms* methodology to evaluate a wide array of tracking defenses. This array includes ad-blockers (many of which claim to block tracking as well as ads), tracking-blockers, and tracking-related browser settings. We investigate these defenses against over 1,000 third-parties that appear on 49 popular domains, using observations from 1,800 simultaneous web crawlers — measuring targeted ads requires significant data collected per site to generate statistical confidence in the results [70].

Our combined approach let us comparatively evaluate technically-disparate defenses, like ad-blockers and private browsing, where other studies have focused on comparing more sim-

ilar types [88, 103, 125]. We also find conflicting results with some of these studies on the efficacy of defenses. Most notably, disabling JavaScript has been among the most effective defenses at reducing tracking mechanisms — which our own mechanisms measurements confirm. However, measuring tracking outcomes show this defense to actually be less effective.

Measurements of mechanisms — both our own and previous works [103, 125] — show disabling JavaScript to be one of the most effective tracking defenses. However, measuring tracking *outcomes* shows this defense to actually be much less effective.

In other cases, our results agree with recent studies of defenses like AdBlock Plus, uBlock Origin, and Privacy Badger [88]. The fact that this is not always the case (see above) shows that the agreement of our results strengthens previous estimates of these defenses' efficacy.

Our results — and the future application of our evaluation methods to new defenses as they arise — can empower privacy-conscious users to make informed decisions about which tracking defenses to use or avoid. Our methods can also empower developers by giving them tools to evaluate and improve their defenses. Finally, the use of tracking outcomes enables evaluation of defenses against new tracking techniques before they are even publicly known, because the outcomes of the tracking will still occur even if the mechanism has not yet been understood.

Our Contributions. Our work dives deeply into the efficacy of different web tracking defenses, comparing a broad range of defenses and measuring both the mechanisms and outcomes (specifically, targeted ads) of tracking. We contribute:

- A methodology for measuring both mechanisms and outcomes with different tracking defenses — including those that block ads — and an implementation of this methodology in a web crawler.
- An evaluation of nine tracking defenses with varied goals and functionalities, including: browser settings (Do Not Track, disabling third-party cookies, disabling JavaScript), user practices (browser and system data clearing), and browser extensions (AdBlock Plus, Crumble, Disconnect, Privacy Badger, and uBlock Origin) as well as the effects

of browsing from the E.U. rather than from the U.S.

- Recommendations for users and developers about which tracking defenses are most effective and why. Among these, we recommend that defenses use aggressive blacklists and block requests, not just cookies.
- Confirmation of previous defense evaluations with new methods and results showing some defenses are less effective than previous methods observed.

3.2 Background

This section provides an overview of tracking and targeted advertisements on the web.

3.2.1 Tracking Mechanisms: Cookies and More

Many approaches exist for tracking users online. Below, we briefly describe three of them.

Cookies. Cookies, the most common form of user tracking on the Internet today, store arbitrary information sent in response to a client's HTTP request; the client returns this information on future requests to the same server. Originally implemented to allow session state in the HTTP protocol [67], cookies are now also employed by all major trackers [44].

Cookies set by a specific domain are sent back only to that domain, so, to track a user directly via cookies, the tracker must have content on each page the user visits. Typically, the tracker is included in the HTML as *third-party* content; hence, cookies it sets are *third-party cookies*. Some browser settings and extensions treat these differently, such as by blocking *third-party cookies* [4, 45].

Other Local Storage. There are many other mechanisms in modern browsers that provide the same storage-and-retrieval functionality as HTTP cookies. These include Flash Cookies, HTML5 Local Storage, image caching, and various other mechanisms [6, 57, 59, 109].

Browser and Device Fingerprinting. Several fingerprinting mechanisms have been studied (e.g., [30, 90, 91, 94, 131]). Unlike other tracking methods, fingerprinting identifies users based on their device or browser attributes (e.g., IP address, browser type, fonts) instead of

by using identifiers stored in their browsers. Thus, users cannot delete their tracking cookie to prevent tracking.

3.2.2 Tracking Outcomes: Targeted Ads

A key goal of tracking users is to enable targeted advertising. Since serving targeted ads generally requires tracking users, we measure targeted advertising as another metric to evaluate the degree to which users are tracked — and thus the effectiveness of tracking defenses.

We define *targeted* ads as any ads served to a user because of their browsing history, which is commonly known as *behavioral targeting*. We define *untargeted* ads as ads shown to a user that are not based on stored information about them. Ads may still be context-specific, e.g., showing ads for rock climbing equipment on an outdoors website [46], without being *targeted* according to our definition. However, a rock climbing ad served to a user visiting a political news site because the ad network knows the user previously visited a rock climbing site *is* a targeted ad according to our definition.

Direct and Indirect Targeting. Another important concept in our investigation is relationships between trackers and ad networks. Continuing the prior example, the ad network must reidentify the user as the same user that a tracker previously observed visiting the rock-climbing site. This requires an information sharing relationship between the tracker and ad network, which we classify as either *direct* or *indirect*.

In a *direct relationship* the tracker and ad network are the same domain. In this case the re-identification is straightforward: the tracker places a cookie or otherwise identifies the user to begin with, then that cookie or identifier is retrieved when the user later visits another site.

An *indirect relationship* is more complex. It is easy for a tracker to share its back-end data about a user to an ad network. However, the ad network has no way to re-identify that user when they visit a site because the ad network cannot access cookies set by the tracker. Cookie syncing is the typical technique for solving this problem [63]: in this technique the

user must visit a webpage that includes both the tracker and ad network as third-parties. The tracker then purposefully leaks the value of its own cookie to the ad network, allowing the ad network to match the user with data the tracker shared on the back-end.

Defense	Type	Block Cookies	Modify Cookies	Block Scripts	Ad-Blocking	Heuristics	Delete Data
No-3rd-Party	Setting	X	-	-	-	-	-
No-JS	Setting	-	-	X	Incidental	-	-
DoNotTrack	Setting	-	-	-	-	-	-
Clear Browsing Data	Setting	-	-	-	-	-	X
AdBlock Plus	Extension	X	-	X	X	-	-
Crumble	Extension	-	X	-	-	X	-
Disconnect	Extension	X	-	X	Incidental	-	-
PrivacyBadger	Extension	X	X	X	Incidental	X	-
uBlock Origin	Extension	X	-	X	X	-	-

Table 3.1: The different types and features of defenses evaluated in this study. Block Cookies can refer to either or both the setting or sending of cookies. Block Scripts refers both to blocking the execution of scripts and the blocking of requests. Heuristics refers to systems that examine content or actions to identify tracking rather than simply using the domain or URL.

3.3 Tracking Defenses

Our work aims to study the efficacy of tracking defenses. As we have seen, there are many different types of tracking defenses, both external and included directly in browsers. We pick representative examples of different types of defenses to cover a wide spectrum of technical approaches (see Table 3.1); our selection process resulted in selection of nine implementations of tracking defenses for the Chrome browser:

- **Browser Settings:** (1) block third-party cookies and site data, (2) disable JavaScript, (3) send the Do Not Tracker header flag, (4) clear browser data.
- **Browser Extensions:** (1) Adblock Plus, (2) Crumble, (3) Disconnect, (4) Privacy Badger, (5) uBlock Origin

3.3.1 Browser Settings

The Chrome browser includes advanced settings that can affect tracking. Although most of these do not explain that they may reduce tracking, privacy advocates commonly recommend them, e.g, [81].

Block Third-Party Cookies and Site Data. The "block third-party cookies and site data" option in Google Chrome blocks third-party domains from setting cookies. However, known methods enable trackers to circumvent this [93].

Disable JavaScript. Disabling JavaScript execution can prevent tracking by mechanisms that require JavaScript to operate. However, this does not preclude inclusion of a third-party tracker in HTML. Extensions can disable JavaScript, and the whitelisting of scripts from certain domains can improve the user experience.

Do Not Track Header. The Do Not Track (DNT) header is a flag browsers send with HTTP requests to indicate a user preference to not be tracked [82]. Previous studies have shown conflicting results regarding the use of this flag on tracking behavior: one study showed that some trackers respected the flag and did not track the user [103], while another showed

it to have no impact on targeted advertising [15], which is based on tracking.

Clear Browser Data. This defense is an automated version of the "Clear browsing data" action in the Chrome browser. Our implementation calls the `chrome.browsingData.remove` API to delete all history, cached data, cookies, downloads, and localStorage. This functionality is equivalent to entering a fresh private browsing mode, like Chrome's Incognito mode, before each page visit.

3.3.2 Browser Extensions

Many browser extensions prevent tracking. We surveyed the mechanisms employed by these varied defenses and chose five representative approaches we discuss below.

How ads are dealt with by the tracking defense is a major source of differences. Some extensions are explicitly designed as ad-blockers; others claim they block only ads that appear to track the user (which we call *incidental blocking*); still others avoid blocking ads at all while still preventing tracking. Ad-supported websites are increasingly using ad-blocker-detecting code to prevent visitors from seeing content with an ad-blocker enabled [95, 97]. Users can either disable their tracking defense on these sites—if it is detected as an ad-blocker—or choose a less aggressive ad blocking defense. Our study provides an answer to which choice is preferable.

Another common difference between tracking defenses is how they decide when to take action. Some use centrally curated blacklists of domains or URL paths (such as `*/econda/*.js*`) to identify a tracker and then block requests, delete cookies, or take other actions. Other extensions rely on heuristics, such as the entropy of a value in a cookie, to classify content as potentially tracking the user. Both approaches have trade-offs: blacklists must be kept up-to-date [51]; heuristics can be inaccurate or not identify the mechanisms of some trackers; both could be overzealous and block content needed for website functionality.

Finally, which tracking mechanisms a defense addresses can also vary. Some defenses solely block or anonymize cookies; some also block other client-side storage mechanisms;

more aggressive defenses block requests altogether.

AdBlock Plus [98]. AdBlock Plus, the most popular ad-blocker with over 10 million Chrome users [41] claims to block "ALL annoying ads, malware and tracking." It does so by blocking requests to URLs that match its central blacklists. AdBlock Plus includes an "Acceptable Ads" policy that whitelists specific ads that it considers non-intrusive. It does not specify whether it considers tracking behavior when determining if an ad is acceptable or not.

Crumble [4]. Crumble prevents tracking by intercepting third-party cookies that are sent out on requests and changing the cookie information that is sent to prevent unique identifiers from being passed to trackers. To avoid breaking the functionality of some websites, Crumble does not indiscriminately block all third-party cookies. It claims to select cookies by "identifying the type of the cookie and not by keeping a blacklist of trackers" [5]. Theoretically, this lets it block trackers that are new or have changed their domains.

Disconnect [60]. Disconnect prevents tracking by blocking third-party requests on a blacklist maintained by the extension. This prevents trackers from setting cookies or other storage and prevents cookies, if they were set previously, from being sent on requests to blacklisted sites.

Privacy Badger [31]. Privacy Badger prevents tracking by a variety of methods, including blocking the setting and sending of some cookies and other storage, as well as blocking some third-party requests entirely. It combines a small blacklist with heuristics to determine if particular content is attempting to track the user. It also tries to determine if an entity it decided to block is required for the functioning of a particular page; if so, it dynamically unblocks it but tries to strip identifying information from requests to preserve as much user privacy as possible without breaking website functionality.

uBlock Origin [50]. The uBlock Origin extension primarily seeks to block ads, but it also blocks trackers. It uses a centralized blacklist system to identify and block requests to trackers.

3.4 Methodology

Our investigation evaluated the efficacy of nine distinct tracking defenses by crawling the web and analyzing both the *mechanisms* and *outcomes* of tracking that we observed. We describe the components of our methodology and framework below, using the following outline:

1. Our Dataset: we provide a high-level description of the data we gather and its scope.
2. Crawling Methodology: we create a novel methodology to gather data on both the mechanisms and outcomes of tracking while using defenses that interfere with advertising.
3. Goldfish: we devise a client-side sandboxing defense, which we call Goldfish, to provide a benchmark for validating our framework's results and determining whether browser fingerprinting affects our results.
4. Crawler Framework: we detail the framework architecture we used to gather the mechanisms and outcomes data for our evaluation.
5. Mechanisms Analysis: we describe our analysis of tracker request and cookie data.
6. Outcomes Analysis: we describe our analysis of targeted advertisements.
7. Experimental Setup: we detail the input set and running environment of our crawls.

3.4.1 Our Dataset: Two Measurement Goals

The objective of our study was to evaluate tracking defenses through two different lenses: measuring tracking mechanisms and tracking outcomes. The dataset we ultimately gather includes HTTP requests, cookies, and ads, from observations by 1,800 simultaneous web-crawlers (see Section 3.4.4). As prior works [23, 69, 70] have shown, evaluating outcomes, such as targeted ads, requires a significant amount of data because outcomes are statistically inferred, whereas studies of mechanisms need only visit a site a few times to measure that facet of tracking [103].

This increased data requirement restricted the number of websites we could crawl: each defense evaluation required running 150 crawlers for 4 days, or about $700 in EC2 costs per

defense. These resource requirements restricted our crawl set to 49 domains (see Section 3.4.7).

However, it is actually the *third-party domains* that are important for our evaluation of the differences in tracking. Further, not all third-parties are equivalent in importance for studying privacy impacts. As Englehardt et al. argue [32], the *prominence* — essentially actual user privacy impact — of a third-party is based on both the third-party's reach and the rank of the websites it reaches. This approach models the prominence of third-parties as a power-law distribution. Thus, there would be rapidly diminishing returns in the prominence of additional third-parties we would observe by crawling more websites.

In these terms, our study succeeds in evaluating tracking defenses against the entities that most greatly impact the average user's privacy: our crawlers encounter 1,009 unique third-party domains, including 56 of the 60 different third-parties identified by recent studies to be the most prominent and concerning from a privacy perspective [9, 32, 88, 132].

3.4.2 Crawling Methodology

We use a novel *seed and crop* methodology to let us gather targeted ads while fairly evaluating defenses that block ads. We visit two types of domains: (1) *seed domains*, or domains that we hypothesize users will see targeted ads for after visiting (e.g., e-commerce websites, like `zappos.com`), (2) *crop domains*, or domains that show many ads (e.g., news websites, like `cnn.com`).

When visiting seed domains, the defense is enabled and prevents any tracking it normally would, and tracking mechanisms (requests and cookies) are recorded. When visiting crop domains, defenses are disabled so that ads can be logged without interference.

We describe which domains we use and how we select them in Section 3.4.7 below. Figure 3.1 shows a conceptual diagram of how trackers, ad networks, crop domains, seed domains, and our crawlers interact.

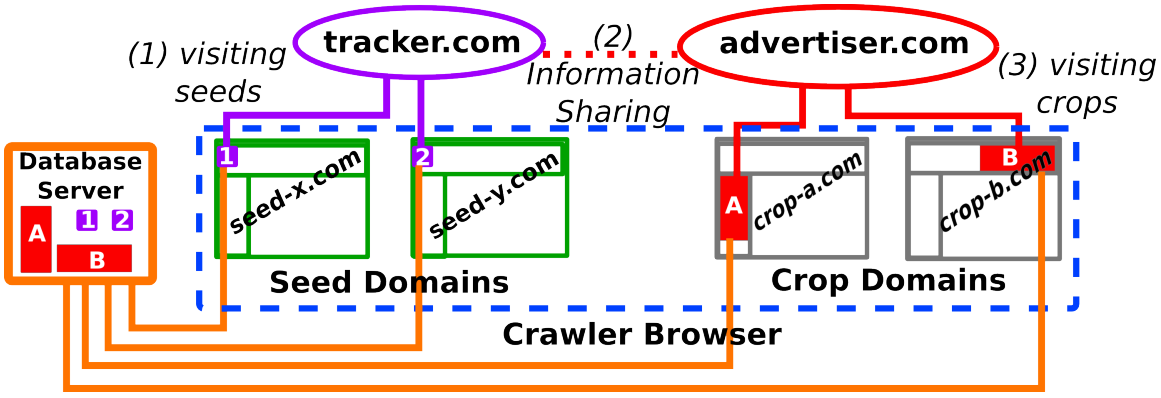

Figure 3.1: Methodology overview: (1) When a crawler visits seed domains, trackers on the page identify and track the crawler, and our framework records these tracking requests and cookies. (2) The trackers can then share this tracking data with other trackers and ad networks on the Internet, independent of the crawler. (3) When the crawler visits a crop domain, an ad network — possibly the same entity as the tracker in (1) or a different entity the tracker shared information with — identifies the crawler as having visited a seed domain previously and serves a targeted ad related to that domain, and our framework scrapes and logs this ad (A or B). During crawling, ads and requests are logged to our database server by the crawler.

3.4.3 Goldfish: Assured Deletion of Client-Side Data

To provide a benchmark of a defense that is 100% effective at removing client-side tracking data — and hence is only trackable using fingerprinting approaches (see Section 3.2.1) — , we devised a sandboxing solution, called Goldfish[1]. We used a Docker container [56] to provide a fully sandboxed file system and processes without the overhead of a traditional VM. We then reset the container to a clean state after each page visit to ensure all client-side storage is erased.

To ensure that the Docker container did not unintentionally alter the results for Goldfish, we ran the crawlers for every defense inside a Docker container. However, we only reset the container for crawls employing the Goldfish configuration.

The fact that this benchmark receives no targeted ads shows that browser fingerprinting

[1]Goldfish have notoriously short memories, just like our sandboxed browser.

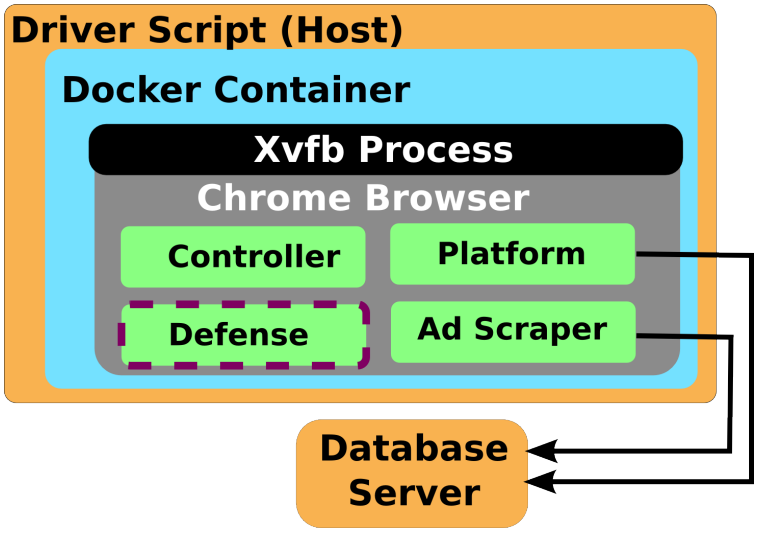

Figure 3.2: Crawling architecture, detailed in Section 3.4.4.

techniques (see Section 3.2.1) are not being used to affect the outcomes that we measure. If fingerprinting were being used for targeting ads, we would expect to see its outcome in Goldfish, since Goldfish does not prevent fingerprinting.

3.4.4 Crawler Framework

We developed a software framework for crawling the web to gather our measurements, which we plan to release as open source upon publication. Crawler components (Figure 3.2) perform the following tasks:

1. Toggle extensions and browser settings on/off before each page visit
2. Automatically navigate to a list of URLs
3. Manage Chrome running inside a Docker container
4. Gather data on requests and cookies (mechanisms)
5. Scrape advertisements (outcomes)
6. Report data back to a central database server

Driver Script and Docker Container. A *driver script* running in the host controls the

crawler. The driver fetches and executes jobs from the *central database server*. The driver manages the *Docker container*, *browser*, and toggling of defenses.

Browser Extensions. To drive the web browser itself, we opted to adapt an existing framework designed to crawl websites and log requests [104] (the *platform* and *controller* extensions shown in Figure 3.2). This extension connects to the database server and automatically browses URLs in sequence.

The *platform* extension also has hooks into the Chrome browser APIs for HTTP request handling and JavaScript cookie setting. These hooks record the tracking mechanisms data for our evaluation (see Section 3.4.5).

The platform also receives data from the *AdScraper* extension when visiting crop domains. These data are forwarded to the database server to be used for the outcomes component of our evaluation. The *AdScraper* extension is a version of AdBlock Plus [98], which we modified to scrape ads instead of blocking them; the AdScraper logs the URL and then attempts to fetch a copy of the content and hash it for each advertisement. Hashing the actual ad content is important since the URL often differs across multiple instances of the same ad. Early experiments found that logging every request AdBlock Plus listed as an ad captured many tracking pixels, web analytics that JavaScript includes, and other third-party content which slowed down both crawling and analysis. As a result, we implemented heuristic filtering to remove tracking pixels (based on size) and included JavaScript (based on URL extension).

3.4.5 Mechanisms Analysis

We analyze our mechanisms data by investigating which third-parties are contacted when visiting each seed domain and then classifying whether these third-parties are trackers. We use three metrics to classify trackers: cookie feature heuristics, an anti-tracking blacklist, and fingerprinting script lists.

Cookie Feature Heuristics. In order to classify cookies as tracking-capable we compare cookie values across page visits and across separate crawlers visiting the same page. If the

cookie has a unique value for each crawler but this value is consistent across page visits for each crawler, then the cookie is classified as tracking-capable. Some cookies embed dynamic values, like timestamps, in the value. This causes the overall cookie value to be different after each response, despite being a tracking cookie. To ensure we identify these tracking cookies, we manually classified cookies that had unique values after each response.

This approach to classifying trackers is also important to correctly measure the efficacy of defenses that involve cookie manipulation or deletion (i.e., Crumble, Goldfish, and clearing browser data). If we only classified trackers using blacklists, we would incorrectly assert these defenses failed just because they contact third-party trackers without actually being tracked by them.

Anti-Tracking Blacklist. After the preceding analysis, we applied the popular EasyPrivacy tracking blacklist [34] to classify the remaining unclassified third-parties as trackers or non-trackers. This layer of classification assumes that a request to a domain on the EasyPrivacy list results in trackers, so it likely overestimates actual tracking that occurs. Moreover, this classification does not accurately measure defenses that alter or delete cookies without preventing requests. However, it is a commonly used methodology, so we include it to allow direct comparison to other works [88, 105, 125].

Fingerprinting Scripts. Finally, we attempted to identify trackers that may have used fingerprinting techniques by searching for known fingerprinting scripts and domains from previous works on fingerprinting [32, 94] and analyzing logs of potential fingerprinting API usage. However, we found no evidence of such fingerprinting activities operating on the client-side; this is not surprising given that previous studies have found fingerprinting scripts on only a small fraction of sites [32].

3.4.6 Outcomes Analysis

Our analysis pipeline for the ads we scraped (i.e., the outcomes of tracking) begins by extracting targeted ads from the pool of all scraped ads and attributing those targeted ads

to specific seed domains. We use the Sunlight statistical framework [70] to do this and produce hypotheses with significant statistical confidence (details below). We then manually verify ads that Sunlight classifies as targeted to eliminate false positives.

When a crawler with a defense is served a targeted ad for a particular seed it demonstrates that the defense failed to prevent one or more trackers on that seed from tracking the crawler.

Sunlight Analysis. Sunlight is a statistical framework that uses machine learning to identify ads that are targeted [70]. Sunlight provides statistically significant hypotheses linking a crawler visiting a specific seed domain to that crawler later being served a specific ad. We provide a brief overview of how Sunlight functions below, but refer to the original work for further details [70].

First, Sunlight uses logistic regression to generate hypotheses for each ad hash. Each hypothesis is a set of weights, one for each seed domain, representing the likelihood that visiting that seed caused this ad to be seen. The weight of a domain reflects the degree of impact it has on whether this ad is shown. Second, Sunlight creates interpretable hypotheses from weights by selecting the most highly-weighted domains (those most likely to cause the ad). E.g., an interpretable hypothesis would be of the form: this ad is served to crawlers that visit `gap.com`. Third, Sunlight tests these interpretable hypotheses against a null hypothesis to determine if they are actually predictive. Testing is performed on a set of testing data withheld from analysis earlier, and a p-value is generated for each hypothesis. Fourth, Sunlight adjusts these p-values to compensate for the multiple testing problem using the Holm-Bonferroni method [52]. We take as the final output all those hypotheses with $p < 0.05$ significance; e.g. this ad is served to crawlers that visit `gap.com` with this p-value, for each unique ad observed.

3.4.7 Experimental Setup

We crawled 29 seed domains and 20 crop domains (see Figure 3.2). The seed domains were selected by taking the top four sites in several Alexa categories of shopping sites, the

Seed Domains
6pm.com, airforce.com, baggu.com, buildabear.com, crownawards.com, donaldjtrump.com, doversaddlery.com, drbronner.com, forever21.com, gap.com, goarmy.com, hideflifestyle.com, hillaryclinton.com hm.com, htd.com, inthelighturns.com, listenup.com, marines.com, mattandnat.com, navy.com, onecall.com, personalcreations.com, personalizationmall.com, rasmussen.edu, theaterseatstore.com, thingsremembered.com, urbandecay.com, weightwatchers.com, zappos.com,
Crop Domains
bloomberg.com, cbsnews.com, chicagotribune.com, chron.com, cnbc.com, cnn.com, foxnews.com, hollywoodreporter.com, huffingtonpost.com, latimes.com, nbcnews.com, news.yahoo.com, nypost.com, nytimes.com, sfgate.com, theguardian.com, thehill.com, usatoday.com, weather.com, wsj.com

Table 3.2: List of seed and crop domains. We crawled 10 internal pages of each domain. Seed domains are picked for being likely to track a visitor to later target with ads. Crop domains are picked for serving lots of ads, and thus likely serving targeted ads.

four main branches of the U.S. military, and the U.S. presidential candidate websites, as of October, 2016. The 20 crop domains were selected from the top Alexa sites in several news categories. The full set of URLs we visited was generated by manually selecting 10 URLs from each domain (i.e., current news stories and various products). In total our crawlers loaded over six million pages and made nearly 200 million HTTP requests.

We consulted with the Sunlight authors and conducted tests to find the minimum amount of crawling required to still obtain accurate results. Each experimental run consisted of 150 crawlers visiting a set of input sites. The input set differed for each of the 150 crawlers: it consisted of the pages of a seed domain selected with a 50% chance (on average, half of all crawlers will visit each seed domain) and then the entire set of pages for each crop domain (every crawler visits every crop domain). This partial overlap ensures there is sufficient data from both crawlers that did and did not visit a particular seed domain for Sunlight

to correctly determine which seed domain caused a targeted ad. The order of all URLs was randomized, which means that the crawler visited seed and crop sites in an interleaved manner. The entire set was visited 10 times to give sufficient opportunity for tracking and subsequent targeting.

Our ad targeting results are based on performing twelve experimental runs: one for each defense, one with no defenses enabled, one with no defenses enabled hosted in the E.U., and one with our Goldfish configuration. The set of URLs for the 150 crawlers was identical across the different experimental runs. For each defense, the defense extension was loaded only on pages belonging to seed domains in order to avoid interfering with ad targeting on crop domains.

Our data was gathered in simultaneous runs from October 12th to October 15th, 2016.

Effect of EC2. Our crawlers were hosted on Amazon EC2 instances to provide the scale necessary for simultaneous runs of the experiments. EC2 instances use IP addresses that can be fingerprinted as coming from an Amazon datacenter; therefore, we also performed an analysis of differences based on crawler IP addresses to determine the impact EC2 had on tracking and targeting. We present the results of this testing below.

We crawled our input set from two IP addresses associated with a university. Additionally, we performed crawls from EC2, with all traffic proxied through our lab machines, in case some other aspect of crawling from EC2, like machine hardware, might affect advertising.

When comparing the proxied EC2 machines to the local machines, we found only a handful of domains contacted exclusively by one set or the other. However, these unique domains appeared in less than 0.05% of page loads, so we attribute them simply to churn in which third-parties were contacted on any given page load. Comparing EC2 machines with EC2 IP addresses to local machines, we found that two third-party domains of interest — `quantcount.com` and `optimatic.com` — were requested on 3.7% and 2.8% of pages, respectively, for crawlers hosted on local machines, but were never requested for crawlers on EC2 IPs. The significantly higher frequency of these requests compared to those that appear on less than 0.05% of pages suggests that these entities were discriminating against EC2

IPs. However, since this impact was the same across all our crawlers, our comparison of the efficacy of different defenses against one another remains valid, and we still see significant differences between defenses in the results that follow.

3.5 Results

This section analyzes our results, compares them to previous studies, and provides an analysis of their differences. We begin by showing the results of our baseline crawl with no defenses. We then show how our combined mechanisms and outcomes analysis refines and validates previous defense assessments, and provides the capability to measure new defenses. Finally, we provide more in-depth analysis of why defenses perform differently and how they could be improved.

Seed Domain	The domains we visit with defenses on and do *not* scrape ads from.
Crop Domain	The domains we scrape ads from and do *not* run defenses while visiting.
Ad Network	A domain serving ad content to a crawler on a *crop domain*.
Third-Party	A domain included on *seed domains* and contacted by a crawler but may or may not be engaged in tracking (e.g. they may not set a cookie). Third-parties are also present on *crop domains*, but we do not examine these because our defenses are not enabled on them.
Tracker	A domain included on *seed domains* that appears to track a crawler based on cookie analysis and could enable *ad networks* to later target a crawler.
Targeted Ad	An advertisement that is shown to some crawlers by ad networks based on those crawlers having visited specific *seed domains*.
Observation	Each time an ad is scraped by the Ad-Scraper in a crawler.
Direct Targeting	When the *tracker* and *ad network* for a targeted ad are the same domain.
Indirect Targeting	When the *tracker* and *ad network* for a targeted ad are *not* the same domain.

Table 3.3: Definitions for terms used in Section 3.5.

3.5.1 The Baseline Experience

To put our defense efficacy results into perspective, we first present the results of our investigation for the Default configuration: Chrome with default settings and only our crawler infrastructure extensions installed (i.e. no tracking defense extensions).

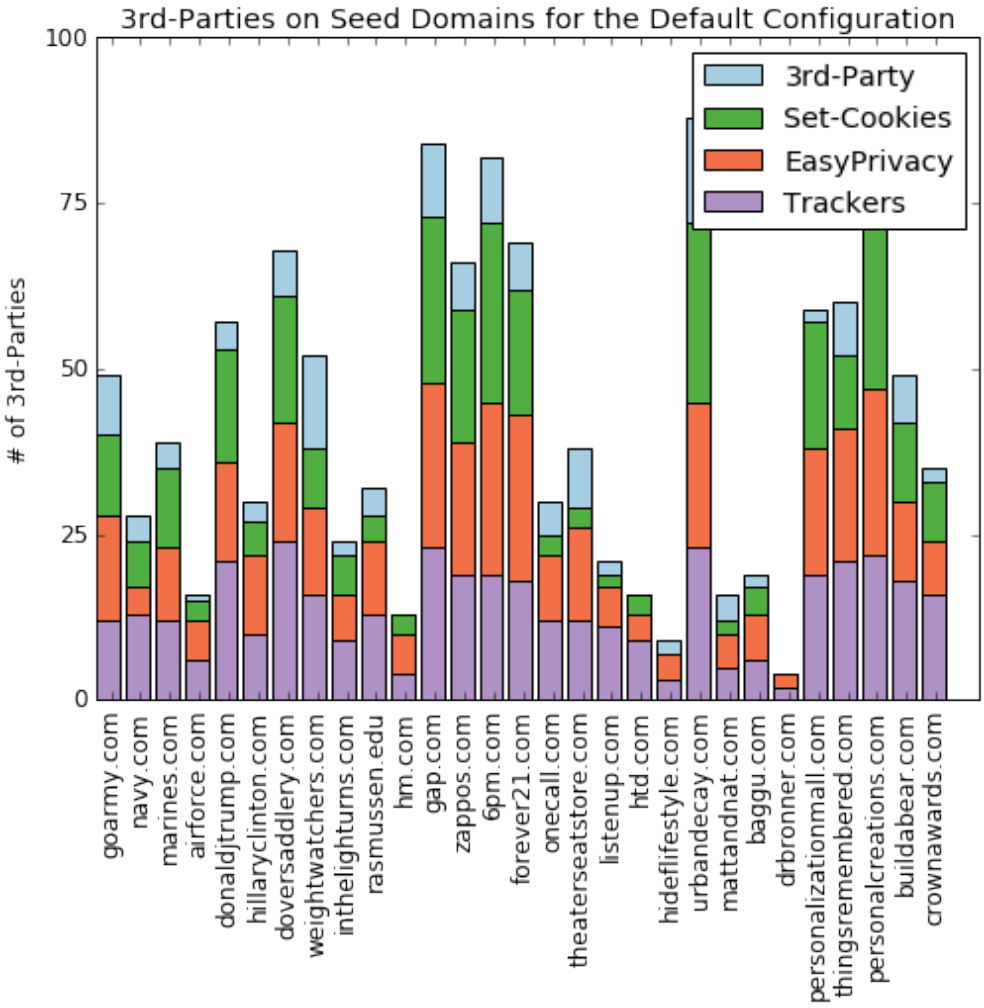

Figure 3.3: The number of third-parties contacted on a visit to each seed by the Default crawler configuration (no defense). The stacked bar shows for each third-party whether it set a cookie, was on the EasyPrivacy list, and appeared to be successfully tracking the defense based on cookie analysis.

Through our mechanisms lens, we observed 464 unique third-parties on seed domains, and 1,009 on seeds and crops combined, including 56 out of the 60 most prominent third-parties identified by recent works [9, 32, 88, 132]. The distribution of these third-parties and trackers varied greatly over the set of seed domains from as few as 4 third-parties (2 trackers) on `drbronner.com` to as many as 88 third-parties (and 23 trackers) on `urbandecay.com` (Fig.

3.3).

In targeted ads, we observed 202 unique ads targeting default configuration crawlers. These ads were served by 11 different ad networks and were targeted based on a crawler visiting one of 18 different seed domains (Fig. 3.5). Overall there were 94,303 observations of targeted ads served to the 150 default crawlers (Fig. 3.6).

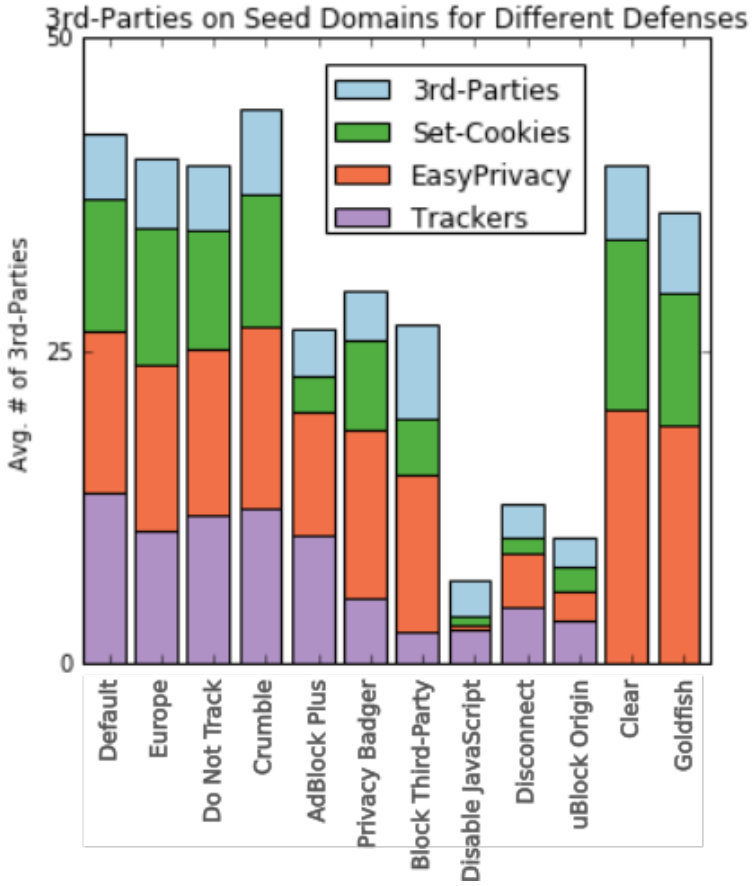

Figure 3.4: The average number of third-parties contacted across all seeds when using each defense. The stacked bar shows for each third-party whether it set a cookie, was on the EasyPrivacy list, and appeared to be successfully tracking the defense based on cookie analysis.

3.5.2 Overall Efficacy of Defenses

Previous defense evaluations have fundamentally relied on mechanisms-only methodologies. As a result, it is possible that prior conclusions about defense efficacy have been incomplete. In this section, describe how our results agree *and* disagree with previous works, and we present results for new defenses not previously assessed in the literature.

Contrary Results

We first describe cases where our two-source evaluation provides results that contradict previous mechanisms-only studies — directly demonstrating the need for a multi-lens measurement of defenses.

Block Third-Party Content. Previous studies have determined disabling third-party content to be effective at reducing tracking to varying degrees [32, 103]. Our mechanisms results support these findings: blocking third-party content reduces *trackers* encountered by more than the two most effective extensions — uBlock and Disconnect (see Figure 3.4).

However, our outcomes data (see Figures 3.5 and 3.7) show that many ad networks were still able to target, and thus must have still tracked, the crawlers with third-party content blocked. This result shows that blocking third-party content was not as effective at preventing tracking as mechanisms data alone suggest, and highlights the importance of having a second source of data for evaluation.

Disabling JavaScript False Efficacy. Similarly, previous studies [103, 105] concluded that disabling JavaScript is one of the most effective defenses. Again, our own mechanisms results agree, showing one of the most drastic reductions in trackers (and third-parties overall) when crawlers disabled JavaScript. However, again our outcomes data provides a more complete assessment: as with blocking third-party content, crawlers disabling JavaScript are still successfully targeted — and thus tracked — on many seeds. Thus, while disabling JavaScript appears to reduce the raw number of trackers and third-parties encountered, it appears to not reduce actual tracking — as indicated by *outcomes* — as much.

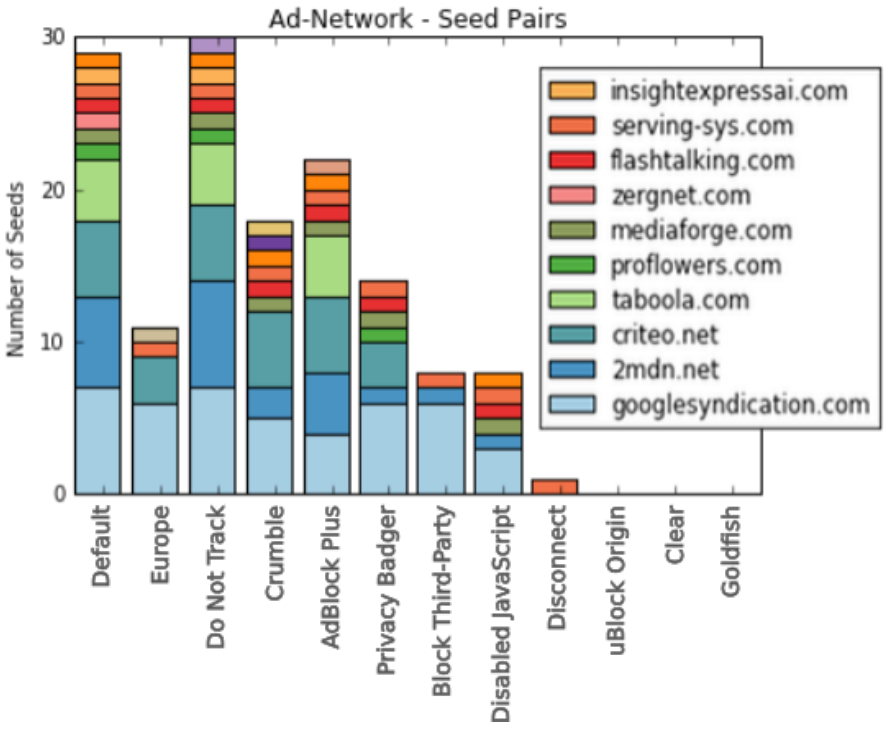

Figure 3.5: Ad networks that served targeted ads to crawlers using each defense. The internal divisions show the number of seed sites for which the ad network served targeted ads.

Validated Results

As the above examples demonstrate, outcomes can provide contrasting results to those derived from mechanisms. The fact that such contradictions do sometimes arise gives us more confidence in the evaluation of a defense when both data sources agree. Below we describe the cases where our evaluation confirms prior findings.

Four Extension Efficacies Validated. Unlike disabling JavaScript and blocking third-party content, our results agree with previous assessments for four of our defensive extensions [88, 125, 132]: AdBlock Plus, Disconnect, Privacy Badger, and uBlock Origin.

As prior works have, we determine the ranking of these defenses with uBlock being the most effective, followed by Disconnect, then Privacy Badger, and finally AdBlock Plus as the least effective of the four (see Figures 3.4 and 3.5). In particular our results confirm those of

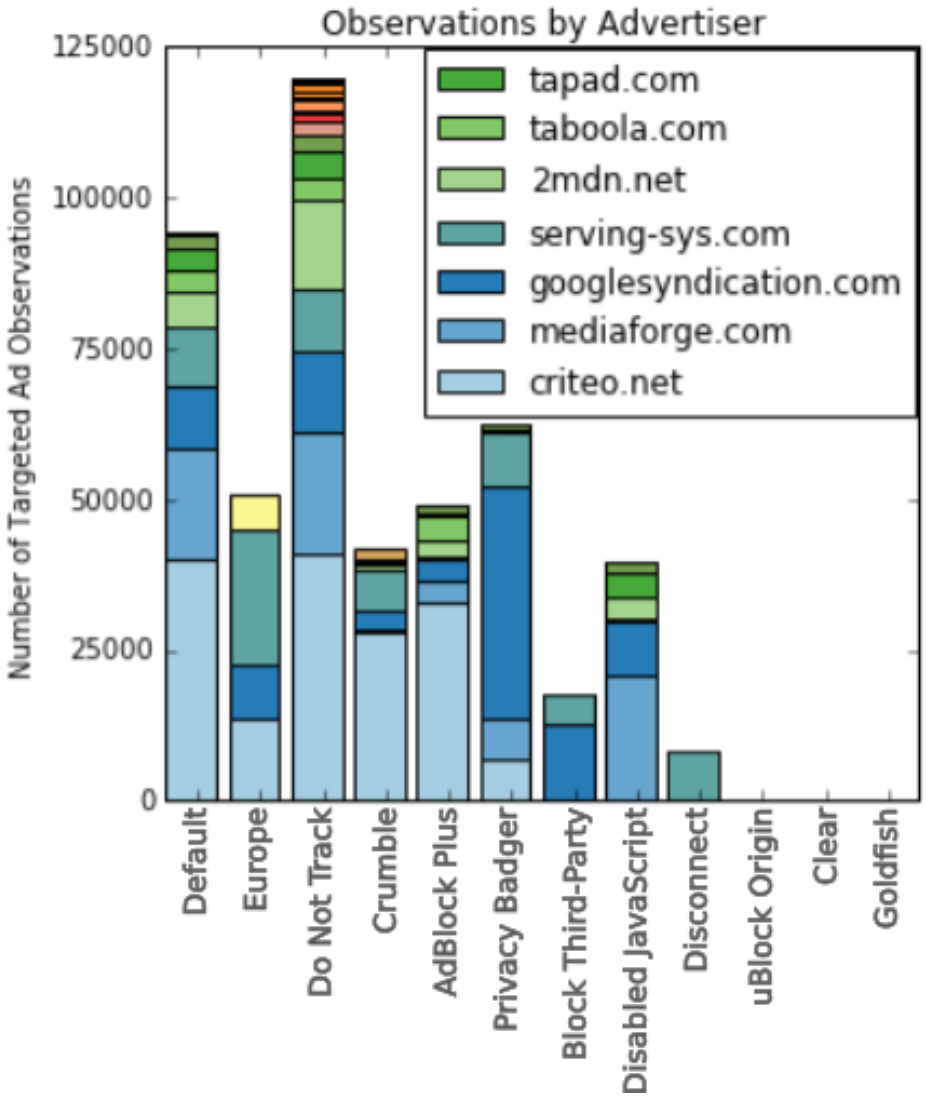

Figure 3.6: The number of times crawlers of each defense observed a targeted ad. The internal divisions show which ad network served these targeted ads.

Merzdovnik et al. [88], closely matching the comparative differences in mechanism reduction and further supporting these differences with agreement from our outcomes data.

Do Not Track. Impact of the Do Not Track header has had mixed evaluations in previous works. Two studies have found that it modestly reduces tracking [103, 105], but several other studies have not found this reduction [7, 15]. Our own results support these latter results:

we see no reduction in mechanisms or outcomes from enabling the Do Not Track header.

Crawling from Europe. In addition to crawls from the U.S., we also ran a baseline crawl from a data-center in Frankfurt, Germany, because different tracking and privacy legal regulations could theoretically change how users from this location are treated. Specifically, if these regulations caused users not to be tracked, then a potential tracking defense could be to use proxies located in the E.U. Recent work [37] found some indications that the country of the server, not the client, was the main determining factor for tracking activities but was unable to draw firm conclusions. Our mechanisms data agrees with these results, as we see little change in trackers when browsing from the E.U. versus U.S. to our U.S.-based seeds. Targeted ads were somewhat reduced; we hypothesize that this is due to the advertiser only purchasing ads for users in the U.S.

Results of Novel Assessments:
Crumble, Clear, and Goldfish

As described above, our study provides new insights and strengthens assessments of previous works using our bifocal approach. However, we also provide evaluations of several new defenses. In particular, these defenses have in common that they function by manipulating or deleting cookies, which makes them unable to be analyzed using many previous works' methodologies. Our results for these new defenses diverge greatly:

Crumble is one of the least effective defenses. It does manage to defeat all trackers used by `taboola.com` (see Figure 3.5), but otherwise shows little reduction in mechanisms or outcomes.

Clear and Goldfish, however, both showed the greatest reduction in trackers and no successful targeting (see Figure 3.7). The two datasets agreed in this case, and the absence of targeting outcomes allowed us to have more confidence that our choice of mechanisms to measure has not obscured persistent tracking. If our classification of trackers had been inaccurate or fingerprinting had been used, then our conclusion that these defenses were

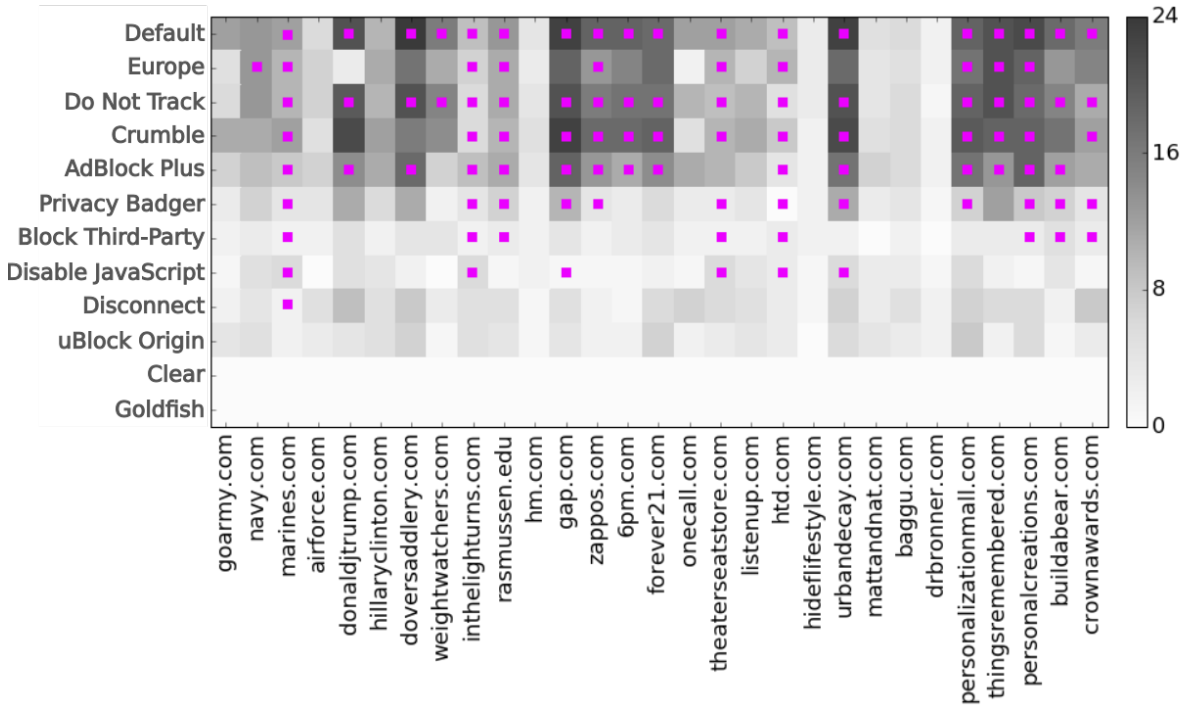

Figure 3.7: Both mechanisms and outcomes for each seed and each defense. The gray-scale shows the number of trackers that are contacted. A pink dot indicates that a targeted ad for this seed was later observed by crawlers using this defense, indicating that the crawler was successfully tracked on this seed and later targeted.

effective could have been incorrect. Instead, the lack of tracking outcomes reinforced the assessment that all tracking mechanisms were indeed prevented.

3.5.3 Defense Recommendations

In this section, we explore the implications of our results to generate recommendations for improving defenses and analyses in the future.

Browser Settings Have Poor Efficacy. Browser settings like disabling JavaScript and blocking third-party content appear effective through the lens of tracking mechanisms. However, our outcomes data show that these defenses are circumvented by trackers on a third of the seeds on which the Default configuration crawlers are tracked (see Figure 3.7). While

the number of trackers may be reduced, trackers are clearly still functioning and sharing this information with ad networks, so the overall user privacy gain is limited.

Some Blacklists Are Better Than Others. The only browser extension defense that prevented all tracking outcomes was uBlock Origin. While Disconnect also performed well, it was still successfully tracked and retargeted by `serving-sys.com`. Our analysis of third-party requests on seed domains with each defense showed that uBlock Origin had a more extensive blacklist than AdBlock Plus or Disconnect.

Comparatively, AdBlock Plus showed little reduction in tracking mechanisms (Fig. 3.4). It showed some reduction in targeted ads, but failed to eliminate any of the three most prevalent ad networks (Fig. 3.5). When we analyzed the mechanisms data (see Section 3.5.4 for details) we found that these ad networks were using *indirect* targeting (the trackers and ad networks were different entities) when serving ads. However, rather than many small trackers slipping through the cracks in various defenses and feeding information to the ad networks, we found well-known domains like `doubleclick.net` responsible for enabling all the targeting by ad networks `googlesyndication.com` and `2mdn.net`. Given this, it should be easy for these defenses to improve significantly by just blocking a few prominent trackers.

The results of these blacklist-based defenses emphasizes there are significant differences in efficacy depending on the blacklist, and the difference in both mechanisms and outcomes confirms recent evaluations of these types of defenses [88].

Privacy Badger's Heuristics Are Modest. Privacy Badger's design uses heuristics to determine if actions a site attempts, such as setting a cookie or sending a request, are tracking behaviors, and then blocks the behavior if it is classified as tracking. In addition, Privacy Badger adds more domains to a learned blacklist over time as it observes them repeatedly engage in tracking behavior. In our evaluation, we chose to evaluate an initially untrained version of Privacy Badger that then acquires training during the crawl, to represent its effectiveness upon first installation by a user.

By analyzing mechanisms data over time, we determine that Privacy Badger required

visits to approximately 20 first-party domains (about 200 page visits, or 3 hours of crawling) to reach maximum efficacy.

Overall, we suggest that Privacy Badger incorporate a blacklist like uBlock's in addition to its heuristics. If that is technically infeasible, then we recommend distributing a "warmed-up" version of Privacy Badger that has been pre-trained on a set of sites like the Alexa Top-1,000, so that privacy-sensitive users are immediately protected.

Cookie Deletion Works. Our analysis of cookie-deleting solutions finds them to be very effective in our dataset. While deleting cookies and other local storage is a theoretically sound approach to preventing tracking (except fingerprinting), it was not tested in previous works. This may be because accurately capturing the impact of these defenses is more difficult: for example, some previous methodologies only count third-party requests [125] without analyzing whether these requests contain information that could be used for tracking; that measurement strategy will not capture the impact of cookie manipulation. Our methodology, including measuring tracking outcomes, is able to measure cookie-deleting approaches and finds them effective.

Some Tracking Is Resilient. Our combination of mechanisms and outcomes data also lets us infer information-sharing relationships between trackers and ad networks, much as Bashir et al. [9] recently focused on. We can use these inferences to better understand the differences in defenses we evaluated, which is another benefit of combining both sources of data. We provide an example of this type of investigation below:

For most of our crawlers, `serving-sys.com` tracks crawlers visiting `marines.com` (a seed site) and later targets ads at them on on various crop sites (see Figure 3.7 for affected defenses). Generally this indicates a straightforward *direct* tracking and targeting relationship.

However, when third-party cookies are disabled, `serving-sys.com` cannot track the crawler — its cookie fails to be set. Despite this disruption, `serving-sys.com` still serves a targeted ad to crawlers with third-party cookies disabled. In this case, we detect `doubleclick.net` as the only entity that still successfully sets and receives tracking cookies on `marines.com`.

Thus, it appears that `serving-sys.com` falls back to receiving information from `doubleclick.net` — a case of *indirect* targeting — when it cannot track the crawler itself. This kind of fallback capability is important to understand when trying to assess tracking, and shows that complementing mechanisms data with outcomes can help surface multi-tracker interactions that circumvent defenses in this way.

3.5.4 Mechanisms Analysis

This section describes how we infer the back-end information sharing relationships between trackers and ad networks. The overall process is to eliminate possible trackers leading to a targeted ad by examining which trackers are elminated by defenses that also eliminate the ad. We use the example of ads served by `googlesyndication.com` for `rasmussen.edu` to explain this process.

First, we generate a set of possible trackers by finding the intersection of all trackers we observe for crawlers using defenses that see this ad. Assuming the same tracker is responsible for this tracking in all cases then that tracker should be observed when any of these crawlers visit this site. However, this set could contain potential trackers that are not actually tracking the crawler. For example, `google.com` itself is generally not involved in tracking and advertising, but it often shows up as a tracking-capable third-party. To eliminate cases like these, we take a union of all the trackers contacted when crawlers with defenses that are *not* targeted by this ad. This represents a set of trackers we know are ineffective at tracking the crawlers because the ad fails to be targeted at the crawler even when these trackers are present.

We then take the disjunction of the set of potential trackers and the set of ineffective trackers. The resulting set represents the candidate trackers that could feasibly track a crawler on the seed and then facilitate ads for that seed being targeted at the crawler when the crawler later successively visits a crop.

When a single ad network serves ads for multiple seeds — such as `googlesyndication.com`, `2mdn.net`, and `criteo.net` — we can apply Occam's Razor to further narrow the selection of

trackers likely responsible for the tracking. In these cases, we repeat the preceding procedure for each of the seeds for which the ad network serves targeted ads, and take their disjunction. This is how, for example, we determine that `googlesyndication.com` and `2mdn.net` appear to use `doubleclick.net` as the tracker providing them with the ability to target crawlers. `Doubleclick.net` is contacted on every seed those ad networks serve ads for when visited by any crawler that receives ads from these ad networks. Conversely, `doubleclick.net` is not contacted by any crawlers using defenses that are not targeted by these ad networks. This creates a very clear case where `doubleclick.net` is the obvious culprit for enabling `googlesyndication.com` and `2mdn.net` to target crawlers.

3.6 Discussion

Having presented the detailed results of our investigation, we discuss the three main takeaways from our findings: the utility of measuring tracking defenses from different vantage points (outcomes and mechanisms), advice to developers for improving defenses, and recommendations to users of which current defenses to use.

Measuring Outcomes. As we demonstrated in the analysis of our results, using two independent perspectives of tracking measurement let us conduct a more in-depth evaluation of defenses. We found that disabling JavaScript and blocking third-party content were less effective than a mechanisms-only approach would (and has) concluded. This also means that when our outcomes and mechanisms data agree — as with the success of uBlock and Clear, or the relative failure of AdBlock Plus and Crumble — we can have more confidence in these conclusions.

In this way, our investigation complements existing evaluations of defenses: where there is disagreement we arrive at a more refined understanding; where our study agrees we provide support for existing evaluations from a novel and independent source of data.

Developer Advice. The two most successful defensive extensions were uBlock and Disconnect, both of which use blacklists and one of which was *entirely* successful at preventing

tracking outcomes over our dataset. This finding suggests that aggressive blacklisting is currently the most effective strategy and should be implemented by developers. While heuristic approaches — like Privacy Badger — are promising, including a blacklist alongside heuristics may be a better approach, as long as blacklists remain effective.

Our results also suggest that local content deletion is effective. Whether deleting cookies is preferable to blocking requests entirely depends on a number of usability factors — e.g., how inconvenient it is to be constantly logged out of accounts versus potentially breaking webpages by blocking requests.

User Recommendations. Our results provide suggestions for users about which defenses to use: uBlock Origin was clearly the best defensive extension, confirming another recent result [88]. Despite its success, some unique tracking cookies were still observed when using uBlock. Our results show that if a user wants to be entirely free of commonly encountered trackers they should use fresh private browsing instances for each website they visit. This strategy is additionally useful for users who must disable ad-blockers to use particular websites.

Our results also showed that defense which try to respect ads and only prevent tracking — Crumble, Privacy Badger, and arguably AdBlock Plus with its Acceptable Ads program — are all less effective than more aggressive defenses like uBlock Origin. This suggests that although ad-respecting tracking defenses can work in theory, their current incarnations are not the most effective tracking defenses.

3.7 Related Work

Previous works have evaluated tracking defenses to greater and lesser degrees. Many evaluations have been a secondary component of studies focused on measuring web tracking as a whole [32, 83, 103]; these typically evaluate only a few defenses rather than providing a comprehensive analysis. Some more recent works have focused on tracking defenses, often ad-blockers and different blacklist selections [88, 101, 105, 125, 132]. Both of these areas of work have only used the *mechanisms* of tracking for their analysis of defenses. One work by

Balebako et al. [7] attempted to evaluate defenses via outcomes. However, their methodology did not allow evaluation of defenses that blocked ads and could only use text-based Google search ads.

Our work complements existing studies in two ways: first, we add *outcomes*-based measurements — including evaluations of ad-blocking defenses — which lets us build upon previous results with a new source of data and analysis; second, we evaluate several defenses prior works do not (including manipulating and clearing cookies, as well as a novel sandbox-based approach), and cover a wider range of defense types under a single methodology to allow comparison between these types.

We build several pieces of our investigation on top of previous web tracking works: we base some of our tracking mechanism detection and classification on Roesner et al. [103]; we also use recent studies of browser fingerprinting to try to detect fingerprinting script use [2, 32]. Finally, we use Lecuyer et al.'s work (Sunlight) to identify targeted ads in our outcomes analysis [70].

There exist several other approaches for detecting targeted advertising [9, 15, 23]; like Sunlight, all of these suffer from comparably higher crawling requirements than traditional mechanisms-only measurement studies.

Finally, Bashir et al. [9] measured both tracking mechanisms and outcomes to investigate information-flow between trackers and ad networks, using a similar size input set as in our work. They explore direct and indirect relationships between trackers and ad networks; however, they do not investigate the impact of these relationships on the effectiveness of any defenses.

3.8 Conclusions

In this chapter we provided an evaluation of nine common tracking defenses, an experimental sandboxed browser (Goldfish), and browsing from a different tracking regulatory regime (E.U.). We use a novel methodology combining measurements of tracking *mechanisms* and tracking *outcomes* (targeted ads) for this evaluation against over 1,000 third-parties (in-

cluding 56 of 60 third-parties identified as prominent privacy threats in recent work). We use the additional type of data we gather — outcomes-based measurements — to correct and confirm prior findings and to provide evaluations of additional types of defenses not previously assessed. Additionally, we provide an open-source platform that enables others to use our investigative methodology. Finally, we use the insights of our investigation to provide recommendations to developers and users about how defenses could be improved and which defenses are currently most effective.

Chapter 4

ADINT: EXPLOITING AD TARGETING FOR INTELLIGENCE GATHERING

This chapter broadens the scope from web-tracking specifically to understanding the surveillance threats posed by the online advertising ecosystem more holistically. Specifically, I relate research my collaborators and I conducted to assess the threat posed by individuals gaining access to the surveillance apparatus built by advertisers, and how targeted advertising could be used for intelligence gathering.

4.1 Motivation and Overview

It is well known that nation states have significant intelligence gathering capabilities[1] and, indeed, the Snowden and post-Snowden revelations speak to the extent that at least one nation state (the United States) has instrumented the Internet to monitor people [39]. A key question underlying this work is: is it possible for entities with significantly fewer resources — such as ourselves, as researchers, or any other small group of individuals with only a modest budget and without the ability to instrument the Internet themselves — to compete with the intelligence gathering capabilities of nation-state actors that can exploit privileged network positions? While we posit that the answer to this question is in general "No," we do make the following observations: (1) the web advertising ecosystem (like nation state network instrumentation) *is* expansive, with the ability to reach billions of people around the world [24]; (2) this ecosystem is built to collect, store, and analyze incredibly diverse details about users, from their physical locations to their personalities and interests; (3) this ecosystem *is up-for-sale to anyone.*

[1]The Merriam-Webster dictionary defines "intelligence" as "information concerning an enemy or possible enemy or an area" [87].

We thus refine our fundamental research question to the following: is it possible for individuals (as third party ad purchasers) to leverage the web advertising ecosystem for intelligence gathering capabilities and, if so, how much intelligence might those individuals be able to harvest? Namely, can individuals or small organizations purchase targeted ads, use the web tracking ecosystem's tendrils to reach out to potential targets around the world, and then use the delivery of those ads to targeted individuals as a means for collecting intelligence? We find that the answer to this question is "Yes". Inspired by the U.S. government's naming scheme for different categories of intelligence gathering capabilities [18], including SIGINT (signals intelligence, like radio interception) and HUMINT (human intelligence, like espionage), we dub intelligence gathered through the advertising ecosystem as ADINT (advertising-based intelligence). We stress that ADINT is a capability available to anyone with a modest budget.

Our study makes three key contributions.

- First, we surface the potential for ADINT as an intelligence gathering capability in 2017 — in an era in which intelligence gathering is a topic of international debate [35], in an era in which actors with unprivileged network positions (ranging from terrorists to businesses) are seeking to compete with nation-states [106], and in an era where consumers appear to be losing the web tracking privacy battle against advertisers [68].
- Second, we conduct a rigorous evaluation of ADINT capabilities. We conduct a deep case study to gauge actual ADINT capabilities, using a canonical demand-side provider (DSP, the entities that provide targeted advertising) in Section 4.4. To complement this deep dive, we perform an analysis of 20 other DSPs, to identify and explore the breadth of ADINT capabilities available (Section 4.5).
- Third, we step back and explore the implications of our findings, and of ADINT in general, to key stakeholders, including ADINT operators, potential targets of ADINT, advertisers, and policy makers (Section 4.6).

To foreshadow some results, we find that an individual or small group — such as ourselves — with a $1000 US Dollar budget can use targeted ads to track the locations of tar-

geted individuals as they move from home, to work, and to other sensitive locations. We find that we can target ads to users of specific applications and at specific locations, which means that one can use purchased ads to count the number of Grindr (a gay online dating app) users or Quran Reciters (a religious app) users in a building, such as a government building or a bar. We find that we can use targeted ads to learn when a person is using a specific application (e.g., when a targeted individual is using Talkatone, a messaging app); a natural extension could be to observe whether two targeted individuals are using the same app at the same time, thereby yielding potential side-channel information about communications patterns. Building on our broad analysis of 20 other DSPs, we further identify numerous other intelligence capabilities.

We now begin, in Section 4.2, with background on the advertising ecosystem at large.

4.2 Background & Related Work

We begin with background on web tracking and advertising (Section 4.2.1) and then turn to related works (Section 4.2.2).

4.2.1 Online Advertising Background

Getting an ad to a user on a webpage or app today is a complex task involving a number of distinct entities. To better understand how ADINT operations function, we provide some background on these entities and processes (also see Figure 4.1).

Audience and Publisher. The audience is the users that will see the ads. They see ads when they interact with content from a publisher. Publishers are the owners of websites or apps; they decide where ads are placed inside their content. The publisher generally does this by including ad-libraries (provided by SSPs, below) in their website or app [26].

Supply-Side Provider (SSP). The supply-side provider (SSP) manages publishers and facilitates selling ad inventory — the ad-spaces in a publisher's content — by auctioning it to Demand-Side Providers (DSPs, below) [14]. Ads are more effective at "converting" an audience (i.e.,buying something) when they are specifically targeted. Thus, the more information

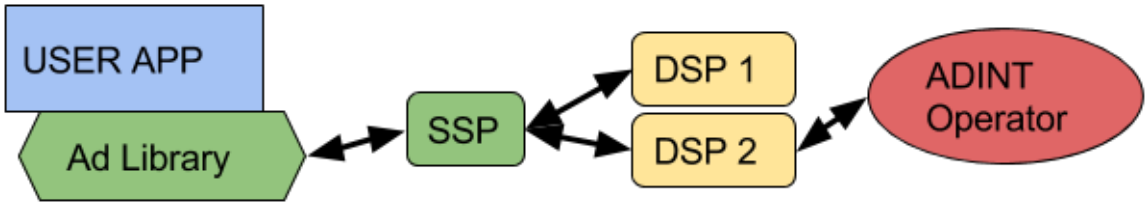

Figure 4.1: Overview of the ad-serving process. Arrows are HTTP(S) requests and responses.

an SSP can provide to bidders the higher the bids they will get. SSPs often provide the ad-libraries that publishers put in their content in order to perform this information-gathering and ad inventory auctioning automatically [112]. The most basic tracking information these ad-libraries send the SSP is a cookie or Mobile Advertising ID (MAID), described below.

Demand-Side Provider (DSP). The demand-side provider (DSP) manages advertisers and bids on ad inventory from SSPs it is connected to [14]. Advertisers are entities that have advertisements they want shown, such as Coke or Nike. DSPs facilitate purchasing an ad slot and serving an ad on behalf of an advertiser. Depending on the DSP, the advertiser will upload the actual ad content they want shown or they can host it on a third-party (or their own) servers and provide a URL for the DSP.

The DSP also provides *ad targeting* on behalf of an advertiser. The types of targeting provided can vary greatly from DSP to DSP; we provide a more detailed survey of these targeting options across different DSPs in Section 4.5. The DSP uses the selected criteria to filter ad inventory and only bid on spaces that will be served to audiences matching the targeting. The information for this targeting decision can actually come from several places: (1) the information the SSP gathers directly from the audience may be forwarded to the DSPs; (2) the DSPs may keep their own data about individuals and reidentify the audience based on information provided by the SSP (such as cookies or a MAID, see below); (3) the DSP could also use a data management platform (DMP) to provide information about an audience [64].

Cookies & Mobile Advertising ID (MAID). Third-party cookies are the classic method

for tracking audiences [11]. Typically an advertiser facilitates a DSP setting a cookie on a user when they visit the advertiser's website. DSPs and SSPS then synchronize cookies [63] so that a DSP can reidentify the same user and decide what ad to serve them when an SSP auctions an ad-slot.

For ADINT purposes we typically do not expect targets to visit the ADINT operator's website. Fortunately, active types of ad content, such as Flash or HTML5 ads, can make requests independent of target interaction. These requests can then set cookies on the target's device and facilitate later retargeting.

The Mobile Advertising ID (MAID) has a purpose similar to tracking cookies. Because of the architecture of mobile operating systems, each app has its own cookie store. This makes each app appear as a different user to traditional cookie tracking and hinders targeted advertising. The solution to this is a sort of whole-device cookie. Originally this relied on permanent device identifiers that were not practically user-resettable. However, on Android phones that use the Google PlayStore, the Google Advertising ID (GAID) has now been introduced that provides an ability to reset the identifier from deep within the settings app.

4.2.2 Related Works

The advertising ecosystem and its security and privacy implications has been a significant area of research, e.g., [2, 7, 10, 32, 65, 66, 103, 126]. Malicious use of advertising *content* (malvertising) has recently been on the rise [73, 80, 133, 134]. Additionally, one recent work showed that some ad-libraries allow sensitive data to be extracted without even exploiting them [110]; we explore extending ADINT operations with malvertising in Section 4.6. The behavior of mobile ad-libraries have also been studied [22, 110], and found that ad-libraries often have poor security as well as fradulant behavior.

Related to advertising and nation state-based intelligence gathering, Englehardt et al. examined the capabilities of an intelligence agency, in a privileged but passive network position, to track user browsing via intercepting the tracking cookies used by advertising networks [33]. In 2010 Korolova conducted an advertising inference attack to learn information users had

uploaded to Facebook but not shared publicly. Their attack showed the idea of extracting information via ad targeting was feasible. Since 2010, Facebook and the rest of the advertising ecosystem have changed significantly; following the Snowden disclosures many technology companies sought to portray a more user-friendly and anti-surveillance image [74, 86], but it is unclear if this has impacted ad tracking. Additionally, Korolova showed an attack on Facebook users; however, as both a social media platform and ad-network Facebook is an anomaly in the advertising ecosystem and so attacks against it do not necessarily generalize to most ad-networks. For example, Korolova extracted information that users *explicitly* chose to share with some entity — information added to their Facebook profiles — albeit not necessarily publicly or with the adversary. We focus on more typical ad-networks, which could not have been used to perform Korolova's attack because of different targeting capabilities.

4.3 Research Questions

We define ADINT as the use of the online advertising ecosystem to collect intelligence about targets, where the party collecting that intelligence (the *operator*) is doing so as an advertiser purchasing ads. The global reach and sophisticated targeting mechanisms of online advertising networks led us to hypothesize that ADINT could return significant information about targets to the operator.

This work is therefore about scientifically studying, and evaluating, the above hypothesis. In particular, we sought answers to the following questions:

1. *Possibility.* Is ADINT possible? Can an operator purchase online ads from a DSP and, as a result of the standard DSP ad display and reporting process, harvest intelligence about targeted individuals?
2. *Capabilities.* What types of intelligence can the operator extract about targeted individuals, using ADINT?
3. *Operational Aspects.* What are the resources required for a successful ADINT campaign, including cost and any necessary preconditions for the attack to be successful? How reliable is ADINT as a mechanism? How efficient is ADINT?

To foreshadow answers to these questions, we find that, for an example DSP focused on mobile advertising, an operator first needs to learn the device identifier for a target's mobile phone. The operator can learn the target's mobile phone device identifier in a number of ways, as we explore. Subsequently, after a $1000 deposit, the operator can learn if a target visits a pre-defined sensitive location within 10 minutes of the target's arrival at that location with high reliability if the target uses an applicable mobile phone application periodically (every 5 minutes, in our tests) at that location. In the above description, location determination is an example of a *capability*, and the need to first learn the device identifier of the phone, the cost, and the needed to obtain the target's location are example *operation aspects*.

We expand on these goals in subsequent sections, as we explore and define the ADINT threat model in more depth. We explore these questions in two ways. First, we conduct an in-depth case study of a single DSP. This case study enables us to empirically evaluate the constraints of an archetypical advertising platform. Second, we survey 20 other DSPs, with the goal of developing a rich and broad perspective on the full extent of ADINT capabilities. We turn to our case study next.

4.4 Case Study

To answer our research questions about the real world capabilities of ADINT, we first conducted a case study of an archetypical DSP focused on mobile advertising. This DSP has a moderate initial cost, and supports a diversity of targeting criteria, including targeting users based on location and demographics. Table 4.5 in Section 4.5 provides an overview comparison of this case study DSP to other DSPs. We choose to focus our evaluation most deeply on leveraging this DSP for intelligence gathering about a target's physical location. We focus first on location as the intelligence objective because, unlike the target's demographics or what apps the target chooses to use, location is dynamic and often changing . Thus location-inference as an intelligence objective is challenging for an operator, and we seek to asses the ability to accurately and quickly learn a target's location. We turn to other

intelligence objectives toward the end of this section.

We conducted our evaluation in two stages. First, we devised a set of benchmarking experiments to develop an understanding of the capabilities and limits of our DSP under controlled conditions. Because a target's location can be always changing, our benchmarks stress-test our adversial capabilities of extracting dynamic information about a target. Second, we created a set of realistic, end-to-end proof-of-concept attack scenarios, and we used these scenarios to evaluate more concrete uses of ADINT.

4.4.1 Case Study Threat Model

To ground our investigations, we now define the threat model for our case study. We discuss a wide variety of potential ADINT operators in Section 4.6. Here, however, we provide a more narrowly focused threat model for our case study. We focus first on location as an intelligence objective, and hence our threat model is formulated around that goal.

The operator's goal is to remotely surveil a specific target over time and obtain sensitive information about that target. The operator wants to know where the target goes, where they live, and other sensitive information such as what apps he or she uses (which could reveal information about them as people).

In this threat model the ADINT operator requires several preconditions to be true:

1. The operator can serve ads using our DSP.
2. The target uses a smartphone or other mobile device and uses apps containing ads able to be served by our DSP.
3. The operator knows the target's device's Mobile Advertising ID (MAID) for some attacks.

For precondition (1), to serve ads using our DSP the operator needs $1,000 for a deposit and to possess a website for the ads to direct to.

For precondition (2), we note that our case study DSP, like many DSPs, serves ads to numerous popular apps. Table 4.1 summarizes the apps that we tested. We explore ADINT's use in the desktop environment and in web ads in Section 4.5 and 4.6 but focus on mobile

ads here. Mobile ads are particularly interesting for location attacks, since people move with their devices.

For precondition (3), obtaining the target's MAID can be done in several ways. In our experiments, (A) we sniff network traffic of target devices to obtain the MAID, which is often sent to ad-exchanges unencrypted. Examples of an operator that could do this include: anyone temporarily in WiFi range of the target when they are on an unsecured network; similarly anyone capable of temporarily intercepting cellular traffic of the target (an increasingly easy attack [16, 25]); or anyone with temporary access to the WiFi router the target uses. An important aspect of all three of these scenarios is that the operator only needs to perform this step once and can then perform ADINT attacks on the target at arbitrary distances and while they are connected to arbitrary networks. Additionally, (B) we experimentally verified that an operator can also obtain the MAID if the target clicks on any of the operator's earlier ads. The MAID can (C) also be exfiltrated via JavaScript in ads in some major ad-libraries [110]. Although we did not do so, it is also possible to (D) purchase the target's MAID online [42].

In some cases it is not actually necessary to identify the target via their MAID. E.g., some of our attacks identify the targets they are interested in by location, so the ad can be precisely targeted at the location of interest rather than specific user devices.

Importantly, our threat model does *not* assume the target will interact with the ads in any manner. Furthermore, we do not include any active content — such as JavaScript or Flash code — in our ads. Refraining from active content allows our case study methods to apply to other DSPs, even those that only allow static image ad content, as some do (see Section 4.5 for more details). Only using static image ads also shows our attacks are not dependent on client-side details such as which ad-library our ad is served to.

The targeting-based ADINT operators that we evaluate here are composable with active ad content to create enhanced ADINT capabilities; we explore this extension of ADINT in Sections 4.5 and 4.6.

Figure 4.2: Facsimile user benchmark testing setup.

4.4.2 Methodology

We used a mix of 10 real user devices and 10 facsimile user devices. The latter enabled rigorous testing of inconvenient scenarios, the former enabled our study to be grounded in use on real users. We describe details below.

Facsimile Construction. We created facsimile user profiles as follows. We used a factory-reset Moto-G running Android 4.4.4 with a new SIM card as the base device representing a facsimile user. The "user" then creates Google, Facebook, and Twitter accounts. Each facsimile user has a different name and birthdate, but all are 27 year-old women. The user also logs into the local WiFi network. The user then downloads the apps we chose to test (see Table 4.1) as well as the Simple GPS Coordinate Display and Packet Capture apps for obtaining ground truth phone-GPS and network data in some experiments. Finally, for measuring benchmarks we gathered the Mobile Advertising ID (MAID) from each user and uploaded them to the DSP as an audience for targeting.

Real User Collection. We recruited real users by emailing our organization for personal Android phones. We evaluated them on benchmarks 2, 3, and 4 (see below for details) to determine how reliably we could serve our ads and if there was any difference in cost from facsimile users.

Apps Tested. We selected apps to test by analyzing a list of the apps our DSP could serve ads to. Since we were primarily focused on location-targeting, we selected an app to conduct

App	Installs	Location Ads
The Chive	5-10M	✓
Grindr	10-50M	✓
iFunny	10-50M	
Imgur	5-10M	
MeetMe	1-5M	✓
My Mixtapez Music	10-50M	✓
Talkatone	10-50M	✓
TextFree	10-50M	✓
TextMe	10-50M	✓
TextPlus	10-50M	✓
Words with Friends	50-100M	

Table 4.1: Apps we actively tested. These are the most popular apps among those our DSP could serve ads to.

the majority of our evaluations on that had the largest user-base and also allowed location targeting. This app was Talkatone — a free text messaging app listed as having between 10-50 million users. We tested 10 other popular apps to ensure we could also serve targeted ads to them (although not all allowed location targeting), see Table 4.1.

Experimental Actions. Our users can be in either an *active* or *inactive* state: in the *active* state the app is open and the device is awake; in the *inactive* state the app is in the background and no ads were being loaded.

Ad Creation and DSP Use. Since we were advertising on behalf of our organization, we obtained approved static image ad content. Thus, when we served these ads to the general populace we were simply conducting real advertising for our organization. When we needed to create numerous ads (as in the location attacks described later) we used the Sikuli automation tool [130] to automate the creation of ads. A screenshot of our DSP's ad targeting is included in Fig. 4.3. While we performed very odd targeting compared to a normal advertiser, we never received any negative communication from our DSP over the three-month period we used them.

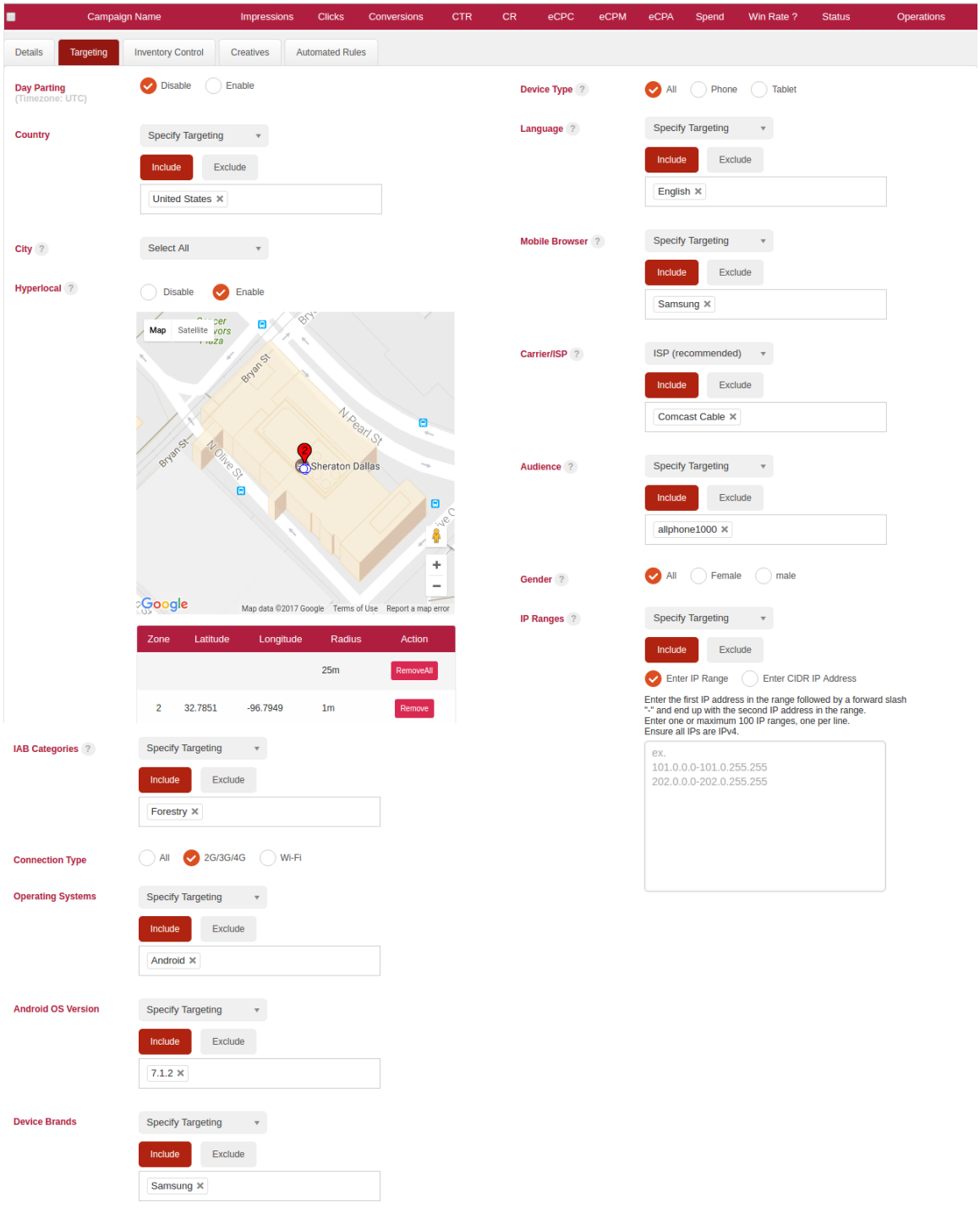

Figure 4.3: Screenshot of the targeting selection page in our DSP showing the wide range of targeting options available, including demographics, locations, and IP addresses.

	Serve Delay	**Report Delay**
Mean	2m46s	6m38s
Max	3m20s	6m48s
St. Dev.	0m24s	0m11s

Table 4.2: Observed delay from campaign activation to first ad serve and from serving an ad to the DSP reporting it.

4.4.3 Benchmark Evaluations

We begin our assessment of ADINT using a series of isolated benchmark measurements. The purpose of these is to establish the limits of various aspects of the ad-serving ecosystem and the DSP's relationship to the app. Having concrete limits allows us to understand what types of real attacks are feasible rather than constructing attacks by trial-and-error.

Benchmark 1: Delay to Service. We first performed two benchmarks: (1) how long the delay is between activating an advertising campaign targeted at the advertising ID of our users and the first ads actually being served. (2) how long it takes from an ad being served until our DSP's reporting interface shows it was served.

We first activate the advertisement and then enter the active state for the user, timing how long it takes to receive our ad. We then time how long it takes for the ad to be reported as served by our DSP's reporting interface. We perform this Benchmark for each user 10 times and show the distribution of times in Table 4.2: on average a campaign served its first ad within 2m46s, and never took longer than 3m20s. Our DSP reported ads in 6m38s on average, although some took up to 10s longer to be reported.

These benchmarks show that ADINT operations can be dynamic on a timescale of minutes: new ads, for a new intelligence-gathering campaign, can be active within minutes and the information gained by an ADINT operation can similarly be known within minutes.

Benchmark 2: Overall Ad Win Rate and Affordability. How frequently targeted ads will be served — and whether they are affordable to individual actors — is critical to how ADINT can be used by operators with modest resources. This benchmark examines how often our ad wins its ad auction when the financial investment of the operator is only

moderate.

Our case study DSP, like most DSPs, allows the advertiser to specify the per-impression bid and the ad auction is then run as a second-price auction, so we only pay the second-highest bid [119]. We conducted win rate and cost tests with bids of $0.05, $0.005, and $0.0005 per-impression and then creating ads targeted at our facsimile users, our real users, and a set of untargeted ads that could be served to anyone. Testing against real users important to ensure the cost of each ad served is not prohibitively high for ADINT operators with small budgets. We found our win-rate diminished significantly with bid: $0.05 won 96% of auctions, while $0.005 only won 52%, and $0.0005 only won 15%. However, we found even bidding $0.05 per-impression resulted in paying $0.005 per-impression on average because of the second-price auction.

When targeted at both real and facsimile user devices, our ads with a bid of $0.05/impression won 90% of auctions and cost no more than $0.02/impression. This means ADINT ads are reliable because they will be consistently served and they are readily affordable to even low-budget operators. We use the highest bid for all subsequent experiments.

Benchmark 3: First-Ad Dominance. This benchmark measures how often our ad is the first ad served after an app is opened. This is important for tracking when a target visits a particular location if the app may only be open for a short period of time.

To measure this benchmark we enter the active state by opening the app and then wait for the first ad to appear. We record whether this ad was ours or not, and then return to the inactive state for 1 minute. We repeat this cycle 10 times for each user. We also conducted this benchmark test with our real user phones to validate that a potentially richer advertising profile did not cause our ads to be shown less reliably.

We find for real and facsimile user devices that our ad is the first ad 79% of the time. This means we can reasonably rely on our ad being served even when a user only uses an app briefly.

Benchmark 4: Repeat-Ad Dominance. Complementary to the above benchmark, what

percentage of ads shown over time in an app are ours is also useful to know for certain attacks. In particular, we could compute how long a user used an app if we know how often ads are fetched and how often our ad is the ad shown, as we demonstrate later in our attacks.

To measure this benchmark we enter the active state for a single user and app for 3 minutes. During the 3-minute period each user sees approximately 16 ads. We record how many ads are served during this time and whether they are our ad or others.

We find for real and facsimile user devices that our ads account for 81% of the ads shown while an app is kept open. This means we can rely on our ad continuing to be served and thus potentially track how long a target has an app open.

Benchmark 5: Location Precision. Our DSP allows "hyperlocal" targeting by inputting GPS coordinates and a radius around them to target ads. Our DSP only allows 4-decimal places of accuracy on these GPS coordinates (approximately 4-11 meter resolution, depending on latitude, which we simplify to 7m) and a minimum radius of 1-meter, so the most precise we can expect this targeting to be is 8m. However, we did not trust that our DSP was necessarily as accurate as its interface claimed. Additionally, smartphone localization can be inaccurate. Therefore, we conducted a series of tests to measure the real world precision of location targeting.

We first recorded network traffic of the app and ad-library and compared the GPS coordinates sent to a ground-truth sample of a GPS app displaying the current location. We found that the ad-library sends the exact same geolocation API coordinates to the advertising ecosystem as the GPS app displays.

To test the precision of actual ad-serving, we created ads targeted at the GPS location of the phone, truncated to 4-decimal places. This ad was always successfully served to all phones. This benchmark does not address the possibility of location ads being inaccurately served to users outside the targeted area: we evaluate this in the next benchmark.

We find that the device's most precise location is transmitted to the ad exchange, and that our DSP does in-practice offer 8m precision, depending on latitude.

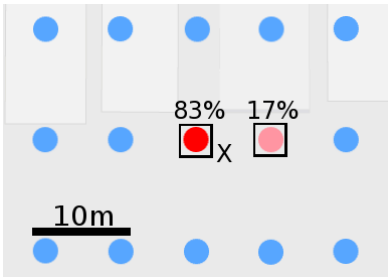

Figure 4.4: Grid of ads used for testing accuracy. Each dot is an ad, the boxed dots are the ads that were served with the percentage of the trials they were served on, the X marks where the devices were actually located.

Benchmark 6: Location Accuracy. Importantly, the last benchmark does not assess the *accuracy* of the GPS targeting in terms of serving the ad targeted closest to the user and not some other nearby ad. Our sixth evaluation was to evaluate the accuracy of these hyperlocal ads and, in particular, whether ads might also be served to other nearby locations.

We created a grid of hyperlocal ads spaced the minimum distance apart (8m), see Figure 4.4. We then placed the phones at the same position and waited for a stable GPS location, then entered the active state for three minutes. We observed that 83% of our ads served[2]. were the current phone GPS rounded to the nearest 4-decimal GPS coordinate. The other 17% of cases were a different ad targeted at the 4-decimal truncation of the phone GPS coordinates. It is unclear why this nondeterminism existed: this occasional error was observed across multiple phones and the GPS coordinates did not appear to change during the experiment.

We find that every hyperlocal ad served was within 8m of the true device location[3]*, despite also being close to other targeted locations. Thus we can surveil locations at 8m resolution across large spaces by creating these grids of ads.*

Benchmark 7: Location Delay. The temporal dimension of location-targeted ads is also important for cases where the user may be in a location for a short amount of time, or for

[2]Our ads were served 146 times to the 10 phones over three minutes.

[3]While always a location within 8m, a device did not always trigger the *same* hyperlocal ad

attempting to track a user as they travel from place to place. This benchmark measures two metrics:

1. How long the user continues receiving location-based ads for a location X after leaving that location;
2. How long a user must be in a new location X before receiving a location-based ad for that location.

In conducting this measurement we found the two actually had significant impact on each other: if a user moved from some location A to another B and both A and B had ads targeted at them, the user would continue to receive ads for A for between 3-5 minutes after arriving at B. Subsequently they would receive ads for B. If instead the user had not recently received location-targeted ads, then ads for B were shown almost immediately.

We find that tracking a target from one location to another requires them to have been in the new location for 4 minutes. However, serving a location ad to a target not recently location-targeted sometimes requires <1 minute.

4.4.4 End-to-End Attack Evaluations

Conducting the above benchmarks is important for two reasons. (1) it develops a foundational understanding of the capabilities and limits of using our DSP for ADINT. (2) it allows us to intelligently design end-to-end attacks with confidence, rather than find which attacks are feasible through trial-and-error. We provide several realistically motivated attacks using ADINT below, but also believe our benchmark analysis allows others to more easily assess if other attack ideas are feasible. In each of these attacks, unless noted otherwise, we tested the attack on all 10 facsimile user devices and used Talkatone as the generic app the target opened.

Determine Daily Routine. The objective of this attack is to learn the target's home, office, and frequent hangout locations. We first gather the MAID from the target via sniffing WiFi traffic; we then create a grid of targeted ads around the city the target lives in.

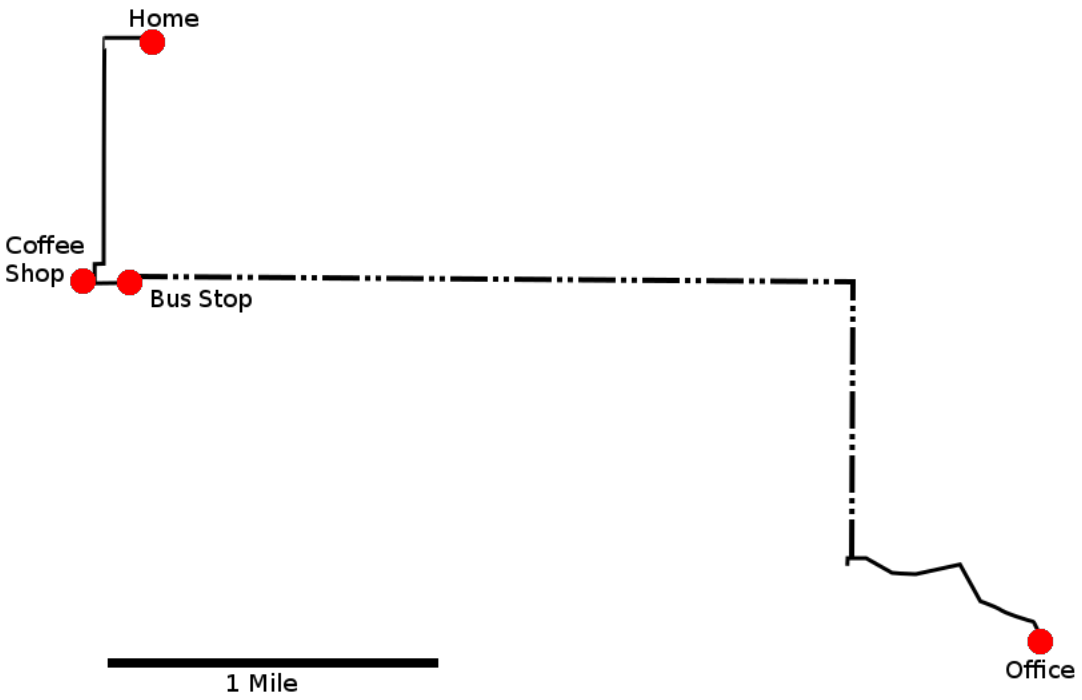

Figure 4.5: Commute route proceeding from left to right: red dots are the locations we observe ads targeting (home, coffee shop, bus stop, office); the solid line is walking; the dashed line is busing.

The target then followed a normal daily routine of commuting 2.5 miles to work — 2 miles by bus and 0.5 miles by walking (see Figure 4.5). The target activated their app at least once during their commute while: at home in the morning; walking to the bus stop; at a coffee shop near the bus stop (see Fig 4.6); waiting for the bus; on the bus; walking to the office; and in the office. We conducted this experiment for 7 days with all 10 of the facsimile user devices.

The ads targeted closest to their locations in the home and office were served within the first day in each trial. The ads for the coffee shop and bus stop were only intermittently served — although all were served at least once to each phone within the trial period.

This difference can be attributed to the requirement of benchmark 7, location delay: the

ad for the coffee shop or bus stop would only have been served if the target stayed there for at least 4 minutes. This also demonstrates why no other intermediate locations during the commute are targeted: the target was constantly moving.

We find that operator using this DSP can purchase passive ads and use those ads to determine the home and office locations, as well as any stops longer than 4 minutes, in the movement of a target.

Sensitive Visits. The second location attack we were interested in is whether we can detect when a target visits a location just once or very infrequently. Example locations of interest might include specialized medical centers, religious centers, known activist meeting points, weapons stores and weapons ranges, corporate offices of a business competitor, and so on. As with the daily routine attack, we begin by sniffing the target's MAID from WiFi traffic. Rather than creating a grid of ads throughout the city, we create a set of five ads, each targeting a 5m-radius point just inside five different buildings on our university campus: these single-point ads are designed to mimic having a set of sensitive locations the attacker hypothesizes the target might go and wait to be served, like those mentioned above.

The target then visited one of the five locations and activated their app in 2 minute intervals, representing idly sitting in a waiting room or lobby. In all cases the ad targeted to that location was served within 10 minutes. In some cases, ads were served within just 1 minute of arriving at the location.

We find that an operator using this DSP can purchase passive ads and use those ads to detect a target visiting a specific location within 10 minutes of their arrival there.

Crowd Enumeration. A corollary to tracking an individual across locations is tracking unknown individuals at a specific location: e.g., enumerating individuals in a protest at a specific time and location. For this attack, we assume an existing crowd at a certain location. We then create a targeted ad for that location and set the ad to only be served to each device once-per-day (a standard option across DSPs). This way, we can count the number of unique devices being used at that location by simply counting the number of ads

Figure 4.6: Checking apps in a coffee shop.

served. The precision of our crowd targeting is subject to benchmark 5: location precision, and should be 8m at its maximum precision.

We tested this attack with a simulated crowd of our 10 facsimile users. We found our ad was successfully shown to all of them within 8 minutes (including the 3.3 minutes of ad activation time determined by benchmark 1). Each device only received the ad once, preventing us from over-counting. In practice this attack would likely under-count, since it will only count devices that are in-use and using apps with ads in them.

Although using active ad content for ADINT is outside our threat model, this attack could naturally be enhanced by using JavaScript in an ad to set cookies or fingerprint the device for later retargeting. Further, any member of the crowd that clicked the ad would provide their MAID and other fingerprinting features to us automatically. Some users might accidentally click an arbitrary ad, but an ad whose content is pertinent to the members of the crowd (e.g., advertising to donate to a cause associated with the event) might cause a

large portion of the participants to click on the ad.

Using our DSP's reporting, an operator using this DSP can purchase passive ads and use those ads can get a count of devices at the location, when they were there, and some device characteristics such as: gender, device maker, OS, network, and which app the ad appeared in. However, this count is contingent on those devices using apps with ads in them while at the location.

Sensitive App Enumeration. We focused most of our case study on location-tracking as the operator's intelligence goal, and we did so because location — unlike other may other properties — has the potential to be highly dynamic. Having assessed the ability to extract fine-grained location intelligence about a target, the extraction of more static intelligence might seem unsurprising. However, we sought to validate that assumption. Here we focus on one such intelligence goal: the determination of what apps a target might be using. Merely the possession of some apps can be considered sensitive information for a variety of reasons: pregnancy trackers, depression journals, psychiatric drug conditions, and diabetes trackers can all indicate health conditions; dating apps can indicate relationship or sexual preferences; religious text and prayer apps can indicate religion and devoutness.

We begin by sniffing the target's MAID from WiFi traffic, and then create ads targeted at that MAID. Because of the reporting features of our DSP, we do not even need to guess at specific sensitive apps: we simply serve a variety of ad content (to ensure our ad could be shown in any app) and observe what our DSP reports to us about which app our ads were served in (see Fig. 4.7). We use the Grindr app, which is clearly sensitive to possess in some circumstances [96, 111] on each of the facsimile users to verify this attack works.

While the only sensitive apps we tested was Grindr, we provide a list of other potentially sensitive apps our DSP buys advertising inventory for in Table 4.3. Additionally, ads on websites are also reported back in this way. Our case study focuses on in-app advertising, but website visits could certainly be just as sensitive to users as what apps they have installed.

Additionally, variants of this type of attack that do not require a specific user's MAID are also feasible. E.g., discovering if anyone (or how many people) in a certain area are

Date	Campaign	Inventory Source	Apps/Sites	Bid Price	Imp.	Clicks
20170407	C1	Xapads	Grindr_iOS -- 99x617184	$50	6	0
20170407	C1	Smaato	EnFlick_TextNow_INAPP_Android	$50	6	0
20170407	C1	Smaato	EnFlick_TextNow_INAPP_Android	$50	5	0
20170407	C1	Adbund	Madgic-USWest|Grindr - Gay and	$50	5	0
20170407	C1	Inneractive	GO_SMS_PRO -- 620974	$50	5	0
20170407	C1	MobFox	iFunny :) -- 171137	$50	5	0
20170407	C1	MobFox	iFunny :) -- 170365_602789	$50	5	0
20170407	C1	Inneractive	GO_Keyboard_Emoji_Sticker	$50	5	1
20170407	C1	MobFox	Grindr - Gay chat, neet & date	$50	5	0
20170407	C1	Smaato	GO Speed - Android_e698766325	$50	5	0
20170407	C1	Smaato	MeetMe - Android_MeetMe_Android	$50	5	0

Figure 4.7: Cropped screenshot of our DSP's report page; each impression lists what inventory it came from: e.g., "Grindr_iOS", "Grindr - Gay chat, meet & date" and "Madgic-USWest—Grindr" all correspond to an ad being served in the Grindr app.

using certain apps. This is similar to the crowd enumeration attack (above), but extends it to examining what apps individuals in an area are using. The inverse objective — finding where the users of certain apps are — by simply creating a grid of location ads targeted at those apps and observing which ones are served. To avoid violating real users' privacy we did not evaluate these attacks. For example, even testing with our facsimile user phones in our building could expose to us the number of real Grindr users in our building.

We find our targeted ads are shown in Grindr immediately after the activation time. These are then reported back to our DSP as being served in the Grindr context, informing us that the target has Grindr installed.

App Usage. To complete our study of potentially non-obvious information that an operator can extract about a target, we now turn to app usage behaviors. In addition to enumerating apps installed, ADINT can be used to discover when and for how long a target uses an app. We begin the attack by sniffing the target's MAID from WiFi traffic and then creating ads

Gay Dating Apps	
Grindr	Hornet
Jack'D	Romeo
Wapa	Wapo
Dating Apps	
Meet24	MeetMe
Moco	Tagged
Torrenting	
BitTorrent	FrostWire
uTorrent	
Other	
Adult Diapering Diary	Hide My Texts
Hide Pictures Vault	Pregnant Mommy's Maternity
Psiphon	Quran Reciters

Table 4.3: Example Potentially Sensitive Apps

Real Usage	30s	180s	300s
Avg. Estimated Usage	36s	172s	294s
Std. Deviation	7.75s (25.8%)	23.7s (13.2%)	29.8s (9.9%)
Max. Abs. Error	90s (30.0%)	60s (33.3%)	15s (50.0%)

Table 4.4: Actual time the app was open and our estimation of the time based on the number of our ads we served and an average of serving our ad 80% of the time any ad is served.

targeted at that MAID.

To test this attack on our facsimile users we activated the app for one of three different durations: 30 seconds, 3 minutes, and 5 minutes. We then attempted to use the resulting report of ads served to calculate when the user began using the app and for how long. We then estimated how long the app was used by multiplying the number of times our ad was shown by the refresh-rate (11 seconds for Talkatone) and the reciprocal of how often the ad served is expected to be ours (81% of the time, based on benchmark 4):

$$Time = AdImpressions AdDominance * RefreshRate$$

The difference in actual in-app time vs. estimated time is shown in Table 4.4; we generally see that accuracy is poor for predicting short usage sessions (30 seconds), but grows increasingly accurate for longer sessions (3 and 5 minutes).

We find that we can know how long an app was open to within 20% of actual usage time

with 95% confidence when apps are open for 5 minutes.

By targeting multiple users, for example, it might also be possible to use this method to infer information about who is talking with whom (based on whether two users are using the app at similar times, over a sufficiently long duration). An operator's ability to infer this information is directly related to the public debate about whether communications metadata (information about who is talking with whom, and when) should be private or not [21].

Other Intelligence Objectives. Our case study DSP also offers advertisers the option to target ads at specific demographics or user interests, which could be valuable to certain operators. We chose not to purchase and experiment with demographic- and interest-based targeted ads for several reasons. First, demographics and interests, unlike location, are often less dynamic, and hence many of our benchmarking questions do not apply. Second, and most importantly, is that any evaluation would have required sending demographic- or interest-based targeted ads to either real or facsimile users. Sending such ads to our real users could have compromised their privacy, which we did not want to do. To test with our facsimile users, we would have first had to populate our facsimile user profiles with faux demographic and interest information. Given the financial incentive of DSPs to have high confidence in their ability to deliver targeted ads to users, we posit that our evaluation of targeted ads sent to these faux profiles would largely be an evaluation of our ability to create faux profiles, and not an evaluation of ADINT as a vehicle to learn demographic and interest information about targets.

User Defenses. Tracking defenses are generally harder for users to obtain on mobile devices: unlike desktop devices where the browser is the primary program invovled in tracking and most browsers are built to allow customization via extensions which can then act as privacy defenses. Instead, in the mobile world, each app is separately involved in tracking, so only the operating system can effectively provide tracking protection across apps. Pertaining to our case study, we tested two simple actions a user might expect to be effective: (1) opting out of targeted ads in the Google Ad settings does not affect our targeting, because this only

applies to ads going through Google's ad serving, not all advertising occurring on this device. (2) Resetting the MAID (also under Google Ad settings) resulted in our ads targeted to the old MAID still being shown up to five hours later despite verifying that the new MAID was sent out to ad-exchanges as soon at it was reset. If the ad content itself allowed the MAID to be captured (as discussed in Section 4.5) or the target had subsequently clicked on the ad, then the new MAID could be easily associated with the target. This also means advertisers could easily reidentify the user while they are transitioning MAIDs, although we do not know if any advertisers engage in this practice.

4.4.5 Case Study Summary

Our case study provides a systematic evaluation of the DSP for use in ADINT operations. Our benchmarks demonstrate the baseline capabilities of this DSP and the attacks we build upon these benchmarks demonstrate that an ADINT operator can use this DSP to perform a variety of surveillance attacks. These attacks include determining:

- How many users are in a location
- Locations a target visits (even only once)
- When a target is in a location
- What apps a target has installed
- When and how long apps are used by a target

These results answer our broader research questions about the validity of ADINT as a tool for surveillance in the context of an archetypical DSP: an operator can purchase targeted ads with this DSP, and use those ads to extract intelligence about targets. Further, our attacks show the scope of this information includes sensitive data like location and installed apps, Finally, the resources required to utilize the DSP are modest: $1,000 and a website. We now turn to enhancing our understanding of our research questions by stepping back and evaluating the advertising ecosystem at large (Section 4.5).

4.5 Survey of DSPs

The preceding section provided an in-depth case study evaluating the feasibility of several types of surveillance using ADINT. We now step back and evaluate the DSP landscape by conducting a survey of capabilities and limitations of 21 DSPs. This survey of DSPs demonstrates that the features of our case study DSP used for ADINT are prevalent in the ecosystem, i.e., that other DSPs could have been used for the same attacks explored in our case study. Our survey also shows new and different ADINT capabilities, such as targeting individuals by PII (name, email, and physical address).

We selected the 20 additional DSPs from several sources: a previous examination of four DSPs for use in unconventional display advertising by Zimmerman [136]; three general Internet companies that offer advertising (Bing, Facebook, and Google); and other DSPs representing a variety of costs and specializations. Table 4.5 shows the features of these DSPs.

To gather our results we contacted each DSP with a series of questions about its capabilities between November 2016 and May 2017. If the DSP was free, we explored its capabilities by creating an account and using the service. Otherwise, we participated in a guided demo by a sales representative. We were careful not to mislead the DSPs we contacted, and our organization was interested in using our results to inform future advertising decisions for itself.

	Min. Cost	Targeting									Content			
DSP		Demographics	Interests	PII	Cookie/MAID	Device	Network	Location	Domain/App	Search	HTML	Flash	3rd-Party	Web Beacon
Case Study DSP	$1,000	✓	✓	-	✓	✓	✓+	✓+	✓	-	✓	-	✓	✓
Admedo	$5,000	✓	✓+	✓	✓	✓	✓+	✓+	✓	-	✓	✓	✓	✓
AdRoll	$0	-	-	✓	✓	-	-	-	-	-	-	-	-	-
AdWords	$0	✓+	✓+	-	✓	✓	✓+	✓+	✓+	✓	✓	✓	-	✓
Bing	$0	✓	✓	-	✓	✓	✓	✓	✓	✓	-	-	-	-
Bonadza	$300	✓+	-	-	✓	✓	-	✓+	✓	-	✓	-	✓	✓
BluAgile	$1,000	✓+	✓	-	✓	✓	✓+	✓+	✓	-	✓	✓	✓	✓
Centro	$5000 / month	✓+	✓+	✓	✓	✓	✓+	✓+	✓	✓	✓	✓	✓	✓
Choozle	$99 / month	✓+	✓+	✓	✓	✓	✓+	✓	✓	-	✓	✓	✓	✓
Criteo	$0	-	-	-	✓	-	✓	✓	-	-	✓	✓	✓	✓
ExactDrive	$50	✓	✓	-	✓	-	✓	✓+	✓	✓	✓	✓	✓	✓
Facebook	$0	✓+	✓+	✓	✓	✓	✓	✓+	-	-	-	-	-	-
GetIntent	$0	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Go2mobi	$0	-	-	-	✓	✓+	✓+	✓	✓+	-	✓	✓	✓	✓
LiquidM	$1,000	✓	✓	-	✓	✓	✓	✓+	✓	-	✓	-	✓	✓
MediaMath	$50,000 / month	✓+	✓	✓	✓	✓	✓	✓+	✓	-	✓	✓	✓	✓
MightyHive	$2,000	✓	✓	✓	✓	✓	✓	✓+	✓	✓	✓	✓	✓	✓
Simpli.fi	$10,000	✓	✓+	-	✓	✓	✓+	✓+	✓	✓	✓	✓	✓	✓
SiteScout	$500	-	✓+	-	✓	✓	-	✓	-	-	-	✓	-	-
Splicky	$0	✓	✓	-	✓	✓	✓	✓+	-	-	-	-	-	-
Tapad	$2,000	✓	✓+	✓	✓	✓	✓	✓+	✓	✓	✓	✓	✓	✓

Table 4.5: Summary of DSP features: targeting features can be used by ADINT operators for gathering information.

4.5.1 Targeting Criteria

As shown by our case study, the targeting criteria a DSP allows are critical to its use in ADINT: these criteria provide the fundamental limits of what kind of information an operator can obtain. Table 4.5 shows that targeting criteria can vary greatly between DSPs.

DSPs have developed new features over time to create more value for their advertiser customers. One example is Facebook: in 2010 Korolova conducted early work on targeted advertising privacy threats, but could only target individuals by finding unique combinations of basic demographic characteristics and distinctive "Likes" [65]. Now, however, Facebook has added the ability to upload lists of email addresses or phone numbers to target matching individuals. DSPs continue to push the envelope in a number of areas, such as linking users across devices and linking offline actions to online users.

In order to create a useful resource for future ADINT research, we aggregated the raw capabilities into broader categories of targeting. A "✓" represents a common baseline capability in an area (e.g., targeting age within demographics is common to every DSP offering demographic targeting). If a DSP has more extensive features it is denoted with a "+".

Demographics. 80% of our DSPs provide the baseline demographics options: targeting based on age, gender, and language.

35% of our DSPs provided more extensive demographic targeting. This included features such as race (AdWords, BluAgile, Centro, Facebook, MediaMath), sexual preference (Adwords, Facebook), finances (BluAgile, Centro, Choozle, ExactDrive, Facebook), homeownership (Facebook), political affiliation (AdWords, Choozle, Facebook), and employer (Centro, Choozle, Facebook).

Interests. 80% of our DSPs allow interest-based targeting. The baseline case is using the 361 different interests from the Interactive Advertising Bureau (IAB).

Most of these are innocuous, but some — such as A.D.D., AIDS/HIV, heart disease, incest support, incontinence, immigration, legal issues, various religious categories, or U.S. military careers — could certainly be sensitive.

40% of our DSPs allowed targeting more advanced interests. These included pages, articles, or groups liked on Facebook or on- and offline purchases (Choozle, Facebook, GetIntent, MediaMath).

Personally Identifiable Information. 40% of our DSPs allowed targeting ads based on information that is generally unique to individuals. All of these cases supported targeting users based on email addresses. Depending on the service, some required large minimum numbers of emails to use this targeting features: Facebook requires 20+ emails, AdWords required 1000+ emails, and Centro uses another service — LiveRamp — which requires 100,000+ emails. The other DSPs (Admedo, AdRoll, Choozle, MediaMath, MightyHive, Tapad), however, did not list minimums. Of course, as the work by Korolova that prompted Facebook to institute their 20 user minimum pointed out [65], these minimums can be circumvented using fake accounts; indeed, we conducted a preliminary experiment and found uploading 19 entirely spurious email addresses (not even connected to fake Facebook accounts) allowed us to target ads at a test user. In addition to email addresses, some DSPs allow targeting based on other PII: Facebook allows targeting based on phone numbers; MightyHive supports targeting based on names and physical mailing addresses.

Cookies/MAID. Every DSP allows targeting users based on cookies or mobile advertising ID (MAID). Either of these could be obtained by an ADINT operator if the user ever clicks on their ad. They can also be obtained from sniffing network traffic. Finally, active ad content can be used to potentially acquire either identifier, as described more below.

Device. 85% of our DSPs support targeting based on device properties. These properties include the browser brand (in the case of web ads), the operating system type, and the device type (i.e., phone, tablet, etc.). In some cases the exact operating system version (Centro, Facebook, LiquidM, MightyHive, Splicky) and specific device make and model (Centro, Facebook, MightyHive, SiteScout, Tapad) can be specified.

Network. 85% of our DSPs allow targeting based on network characteristics. The basic version of this is targeting devices on cellular versus WiFi networks. 35% of the DSPs

also allow arbitrary IP white- and blacklisting (Admedo, AdWords, Bing, BluAgile, Criteo, Centro, Choozle, Go2Mobi, Simpli.fi).

IP targeting could be used for ADINT in several ways: it can act as a proxy for location if the IP address for a particular company, home, or open WiFi network is previously known. IP can also stand in as a semi-unique identifier if something more specific like a cookie or MAID cannot be used. Finally, IP can be used to link devices or individuals that frequently share the same NAT'd network.

Location. All but one of our DSPs allows some form of location-based targeting. The basic version of this (just a "✓") is restricted to city or ZIP code level of granularity. In addition to limited specification of granularity, the methods used to determine a target's location may be imprecise in these cases, such as using a simple Geo IP database lookup.

35% of our DSPs provide more advanced location targeting — typically termed "hyper-local" — which varies significantly in granularity. AdWords and Facebook are the least granular, only allowing targeting radiuses of approximately 1-mile. Both also allow exclusionary radiuses, but it is unclear whether these can be arbitrarily overlapped to craft very small targeting areas.

Other DSPs typically provide 1-meter radius granularity (Bonadza, Centro, LiquidM, MediaMath, Simpli.fi, Splicky, Tapad), although as our case study shows this does not necessarily mean 1-meter accuracy is possible in practice.

The potential uses for location targeting in ADINT are straightforward. For less granular services, a target could be localized to a city or county. With hyperlocal targeting, ADINT could certainly be used to find where an individual lives, works, or is currently located — as demonstrated in our case study.

Domain/App. 75% of our DSPs allow the advertiser to specify which context — domains (for web ads) or apps (for mobile ads) — their ads will appear in. In Adwords and Centro this specification can include particular pages within a domain. Additionally, even if not restricted, the context is reported for each ad shown.

Restricting ads by domain or app could be used to find people that read certain websites or use certain apps. Furthermore, the reporting of websites and apps could allow an ADINT operator targeting a particular individual to see what websites they visit or apps they use — as shown with sensitive apps in our case study.

Search. 40% of our DSPs supported using search keywords for targeting. Search terms can also be indirectly targeted via interests because the interest profile of a user is partially populated based on what they search for. However, these DSPs are those that specifically allow search keywords to be targeted with ads. Search targeting could be used in a manner similar to targeting domains and apps. People searching for specific terms could be enumerated and marked for further ADINT surveillance. Enumerating search terms of an individual is more difficult, because the ads must be targeted at specific search terms, rather than being able to simply serve the ad to any search and then receive a report of what search term it was served to.

4.5.2 Ad Content

DSPs also vary in what kind of content they allow as ads. The most restrictive DSP in content, Facebook, only allows advertisers to upload static images and then enter text into form fields. Other DSPs, however, allow much more complex content. While our case study is focused on use of ad targeting for surveillance, the content of ads is also useful, as we explore below.

HTML5. 75% of our DSPs allowed uploading full HTML5 ads for distribution. These are typically allowed to include image, HTML, CSS, and JavaScript files. Most DSPs specify that an auditing procedure will take place, but this varies. Choozle stated there was no auditing procedure; Adwords has a fully-automated auditing framework that can be tested against; other DSPs did not detail what auditing entailed. In general HTML ad content can be assumed to allow the advertiser to use Web Beacons (see below).

JavaScript can also be included as part of these ads, but certain functions may cause the ad to be rejected. However, using JavaScript to perform operations like browser or device

fingerprinting, setting cookies, or extracting MAIDs [110], does not violate client security and so could be approved by the auditing process.

Flash. Although Flash is supposedly being phased out, 70% of DSPs still support Flash-based ad content. Similarly to HTML, these ads would allow an advertiser to use Web Beacons to report back about the user. Depending on the auditing techniques, arbitrary JavaScript could also be executed from a Flash ad [136], which would allow the same capabilities as HTML5 content, discussed above.

Third-Party. 70% of our DSPs allowed third-party ad serving. In this setup, rather than uploading the ad creative to the DSP's servers, the advertiser instead provides the URL of their ad to the DSP, while the actual ad content is hosted on a dedicated hosting platform or on the advertiser's own servers. This architecture is used for a variety of reasons in practice, but it allows potentially greater evasion of auditing procedures for the purposes of ADINT — just as it has been used in past malvertising campaigns [134].

If a DSP allows an advertiser to serve ads from their own ad server, then the content can essentially be arbitrary, since the advertiser could selectively send the objectionable ad content only to real targets, while sending a benign version to any auditing requests the DSP might try perform [80].

Web Beacon. 75% of our DSPs support using Web Beacons via either HTML or Flash ad content. In the context of ad content, Web Beacons simply make a request to the advertiser server, which allows the advertiser to immediately know its ad was served. This request is sent when the ad is loaded, not dependent on any user action. The request also allows the advertiser to learn the IP address, User Agent String, and Cookie, of the device that was just served the ad. This can allow easy reidentification of the device from previous ADINT actions, as well as potentially provide new information, such as if their IP address changed.

4.5.3 DSP Survey Summary

Aside from the intricacies of individual DSPs, our survey demonstrates that our case study DSP is hardly a unique case. If our case study has shown ADINT is possible when using that DSP, then our survey shows these operations and even more are possible using these other DSPs.

As can be seen in Table 4.5, DSPs can vary greatly from one another across targeting criteria, content options, and costs. One of the rarest capabilities is targeting based on PII, such as uploading a list of emails or real names to a DSP for targeting. For ADINT purposes, PII-targeting would be the easiest way to start targeting a specific individual (more about these approaches in Section 4.6). However, the other targeting capabilities of these DSPs are often lesser — only Centro and MediaMath offer both PII and hyperlocal targeting, and at $50,000/month MediaMath may fall outside the resources of some potential ADINT operators. Search-targeting capabilities are similarly somewhat rare.

Across ad content there are few DSPs with significant restrictions: only Facebook, AdRoll, and Splicky restrict ad content to static images and text. Thus there is considerable opportunity to couple active ad content with advanced targeting options for increased ADINT capabilities, a potentially rich area of future work.

4.6 Discussion

As the preceding two sections demonstrated, ADINT — the use of ads to extract information about a target — is a practical new tool for third-party, non-nation state intelligence gathering. Here we explore the implications of the introduction of this new intelligence category: how ADINT operations could be conducted for different types of intelligence objectives and who might be interested in the capabilities that ADINT provides.

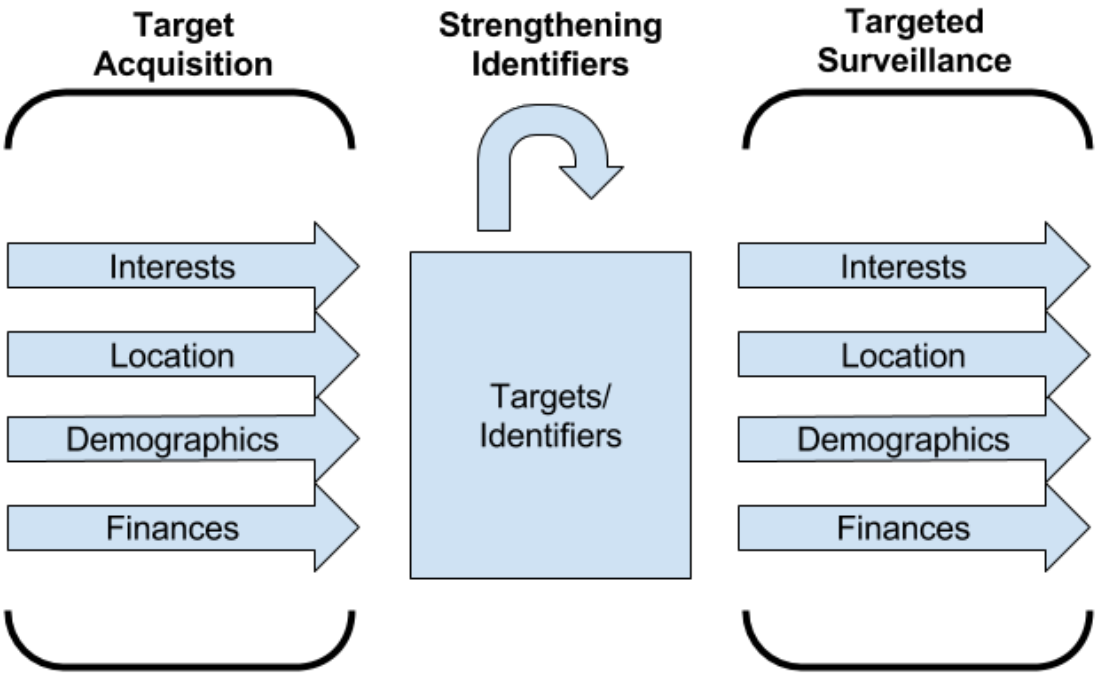

Figure 4.8: ADINT operation stages.

4.6.1 Anatomy of an ADINT Operation

Adapting tools designed for targeted advertising to intelligence gathering requires some additional indirection that is not necessary when using purpose-built intelligence collection tools. Below we provide an abstracted pipeline of how we envision that a complex ADINT operation might generally proceed. This pipeline applies to ADINT operations trying to deeply surveil a target, like most of our case study attacks; it is not as applicable when the operator is conducting simpler one-time attacks, such as crowd enumeration (see Section 4.4.4). The pipeline consists of three stages: (1) target acquisition, (2) strengthening identifiers, and (3) targeted surveillance, which we describe below.

Target Acquisition. The target acquisition stage of an ADINT operation is for operators that know *what kind* of targets they want to surveil, but do not have identifiers for those targets. The more in-depth attacks demonstrated in our case study — like tracking the user

in the physical world or finding sensitive apps they have installed — require specific identifiers to extract information, so an operator needs to obtain identifiers for their targets to conduct these more powerful attacks.

In our case study, this stage involved sniffing the MAID of our targets from network traffic. However, in other ADINT operations a number of different techniques could be used: e.g., obtaining an IP address of a target can be done with a simple Web Beacon and no user interaction; more sophisticated ad content or convincing the user to click on an ad can convey even stronger identifiers (like setting cookies or extracting the MAID).

Using ads for target acquisition allows the targeting features discussed in Section 4.5 to be used. This gives ADINT operators the ability to turn abstract target criteria — such as being a member of a specific religion, using a specific app, or being at a specific location — into a list of identifiers for targets matching that criteria. For example, the operator could use location-targeted ads to a specific government building to learn the IP addresses (with Web Beacons) or the MAIDs (with more sophisticated ad content) of those inside.

Strengthening Identifiers. The second stage is strengthening the identifiers generated by target acquisition. This may be necessary because of the variance in targeting and content capabilities of different DSPs. For example, an operator might want to know every user that visits a particular webpage. AdWords allows very fine-grained web-page targeting but also has stricter content auditing that might restrict an ad to only being able to extract the IP address of a user. Thus, in this example the operator is only able to obtain IP addresses in the target acquisition phase; this does not allow location tracking, since the target's IP address, if connected to WiFi, will change as they move between networks.

To overcome this limitation, the operator can use a DSP with laxer auditing, like Choozle, to retarget the IP addresses with ads that extract a MAID. Then the operator can proceed to use the MAID, a stronger identifier, to perform location tracking attacks with a DSP like the DSP in our case study.

Targeted Surveillance. The third stage is using targeting identifiers in ADINT operations

to learn new facts about the targets. As shown by our case study and survey in Sections 4.4 and 4.5, there are many different types of information an operator can learn using ADINT. Our case study focused on physical location and app usage as information that we could rigorously test on facsimile users. However — as our survey of DSPs shows — the space of information that DSPs can target is much larger and touches on many different aspects of targets. For example, various personal details like sexuality, religion, finances, and search terms can all be targeted.

4.6.2 Prospective Operators

The technical capabilities of ADINT are important, but equally important is who might be interested in using ADINT and for what purposes. These capabilities — like learning a target's location or interests — are already in the hands of nation-state actors, at least for some nation states and in some jurisdictions. However, the relatively low cost and ease of use of ADINT operations opens up these capabilities to whole new swathes of operators; the reduced cost could also incentivize existing users of other intelligence gathering methods to switch to ADINT for some operations. Below we provide several examples of hypothesized ADINT operators and how they might use its capabilities. We focus on just a few for brevity, but note that there are many other potential ADINT operators as well.

Ideological Vigilantes. Across the globe there are many instances of individuals and groups organized with the objective of enforcing cultural norms or ideological positions on members of their community. E.g., the Occupy Pedophilia movement in Russia [8], Muslim Patrols in East London [58], or white supremacy groups in the United States [36]. These types of groups can range greatly in objectives, violence of actions, size, and resources. However, all of them typically act by finding individuals that violate the group's ideology and concentrating their force on that individual.

Depending on the types of targets they are seeking, the targeting criteria provided by ad-networks could directly facilitate finding targets. E.g., an anti-gay group could conduct target acquisition by serving ads in gay apps or location-targeting gay bars and extracting

identifiers. That information alone could be sufficient for the group's purposes, e.g., if that information exposed the number of gay people at a specific location. The group could also use ADINT to gather more information about the targets prior to carrying out some other nefarious objectives.

Law Enforcement. Law enforcement organizations already have access to many types of intelligence gathering tools. However, as our case study demonstrated, ADINT can be comparatively cheaper than existing tools [121]. Furthermore, the indirect nature of ADINT may also mean it is not subject to the same privacy restrictions that other forms of police surveillance are.

Law enforcement could have a wide variety of uses for ADINT. The target acquisition phase of an operation could include getting identifiers for all members of a violent mob or protest by targeting that location; simply demanding them from detained suspects; or gathering them in association with certain crime-related search keywords. In the targeted surveillance stage, location data is the most obvious type of intelligence law enforcement would be interested in collecting on its targets, but search keywords, app usage, or sites visited could also reveal details important to stopping or prosecuting a crime.

Criminals. Stalkers, burglars, and blackmailers, can all make use of ADINT's democratization of intelligence. Depending on how a stalker chooses their victims, a variety of target acquisition criteria could be used, including frequenting the same coffee shop pr having a particular ethnic background. Once a target is acquired, a stalker might closely parallel the very attacks we performed in our case study, such as finding a victim's home, office, and hangouts.

A burglar might select targets by financial categories or past purchases of luxury goods, and then use location attacks to find where the target lives and ensure they are away at the time of the crime. Blackmailers could similarly use financial targeting to find worthwhile victims and then further targeting to gather exploitable intelligence: e.g., knowing when the victim visited a brothel.

Business. We also envision numerous business-related use cases for ADINT, ranging from media-related uses to financial investing. A paparazzi might send targeted ads to the home locations of celebrities using a DSP like Choozle that supports PII-based targeting. The delivery of those ads could leak sensitive information about those celebrities to the paparazzi, such as what apps they use or what interests they have. The paparazzi could also use those initial ads, with active content, to learn the celebrities' MAIDs, after which they could track those celebrities over time. Other journalists might use similar techniques to glean information about politicians and other high-profile individuals.

Financial investors might, instead, acquire identifiers for venture capitalists or executives of companies using PII targeting. Subsequently using the location targeting capabilities from our case study, it could be possible to determine when the VCs or executives visit other companies, possibly indicating when there might be a large round of funding, acquisition, or big announcement, thereby providing valuable investment-related information.

4.6.3 Potential Defenses

We posit that ADINT will remain a viable option for intelligence gathering for many years, because it uses the targeting capabilities of the advertising ecosystem, which continue to increase. However, there are some steps that individuals, organizations, and the advertising ecosystem could take to prevent, mitigate, or detect the use of ADINT. Most of these represent relatively short-term improvements, while the current course of the advertising ecosystem is distinctly towards greater user tracking and profiling, which will only increase the capabilities of ADINT in the future.

User Behaviors. The specific attack instances we explore in our case study could be defended against by either user behaviors or the development of client-side technical defenses. Specifically, users or a system modification could constantly reset device MAIDs to prevent ads being targeted at them. However, if this became a common practice then the advertising ecosystem, app developers, and operating system developers would all be incentivized to create a new MAID-like feature, so this is not a practical long-term approach. Furthermore,

the fact that changing a device MAID did not impact it being targeted with our ads for five hours, it is unclear that this kind of solution would necessarily work.

Users could avoid many ADINT attacks by never using apps or visiting websites with ads. However, it is clear that entire segments of apps and websites are made financially viable through advertising, so this does not appear to be a reasonable approach for all users to take. Users may be able to avoid some of the most aggressive ADINT attacks by avoiding apps that use ads and request many privacy violating permissions, like fine-grained locations. However, this is not necessarily practical if a user wants apps of a certain type. E.g., many dating app now uses location data and many contain ads. Furthermore, pushing users to only non-advertising apps and websites essentially puts a financial price on protection from ADINT that not all users will be willing or able to pay.

Targeting Thresholds. One approach to preventing ADINT is to prevent targeting of individuals with ads. If an ad must target a sufficiently large group, then it should be harder or impossible to extract information about an individual with significant confidence.

Some DSPs already include thresholds like this: Google AdWords requires an "audience" of MAIDs to include at least 1,000 MAIDs and Facebook targeting via email address requires a minimum audience of 20 emails. This latter restriction is apparently directly in response to previous research on ad privacy in Facebook [65]. However, in both cases an ADINT operator can circumvent these defenses by using spoofing and/or sybil accounts. The operator uploads a list of fake identifiers along with the real target. In the case of MAIDs, an operator can make the fake MAIDs appear real by forging ad requests with random MAIDs in the parameters. For an account-connected identifier like an email address for a Facebook account, this is slightly more involved, but fake Facebook accounts can be created or simply bought [17].

An alternative approach proposed has been adding noise to the process, either the targeting decisions or the reporting or both [65, 75], similar to differential privacy approaches in other domains.

Ultimately, DSPs implementing targeting thresholds or injecting noise into the advertising process create a cat-and-mouse game with ADINT operators. It is possible that a

particularly extreme set of changes could prevent ADINT altogether. However, all of these countermeasures impose a cost on legitimate advertising customers, and this cost will create a market for less-restrictive DSPs. This market dynamic makes it unlikely that the open-market advertising-ecosystem will develop effective defenses against ADINT. An alternative approach is government regulation, which could act outside of the economics-driven practices of ad-networks to provide incentive for categorical elimination of ADINT capabilities.

Verification of Advertisers. An important aspect of ADINT is that it opens up the advertising ecosystem surveillance capabilities up to many potential operators. In particular, an ADINT operator claiming to be an advertiser does not need to actually represent a legitimate enterprise. This means arbitrary individuals can use ADINT, because they do not need to be affiliated with any legitimate business. Stronger verification by DSPs to ensure a would-be advertiser represents a legitimate business could significantly reduce the set of possible ADINT operators. Of course, an operator could use an organization they actually work for to buy advertising. However, in that case they would at least have to fear repercussions from their employer if their activities were discovered.

Detection of ADINT Operations. Aside from actually preventing ADINT operations, detecting them could also be of interest. In this case it is useful to have aggregated data from many users to detect if certain individuals are being targeted using ADINT. An example of this would be a company trying to detect if its employees are targets and able to deploy monitors on their network and/or their employee's devices to gather the required data.

The fact that legitimate targeted ads could also be specific to a single individual in a company means this approach could lead to false positives. However, combined with more in-depth analysis of the ads themselves — such as what domains the ads link to — this approach could potentially discover ADINT operations, particularly those by less sophisticated operators. Unfortunately this approach is highly reactionary and may not catch an ADINT operation before it has already gathered whatever information it needed.

4.7 Conclusions

We consider ADINT — the use of the advertising ecosystem, by third-party advertisers, to extract intelligence about individuals of interest. We evaluated the capabilities of ADINT in a case study of location and app surveillance with one DSP. We additionally surveyed 20 other DSPs to evaluate the variety of capabilities and costs they have for use in ADINT and demonstrate the potential for many more types of attacks. Finally, we created a conceptual framework for the execution of ADINT operations in a variety of cases and explored potential ADINT operators in the future.

Chapter 5

CONCLUSION

The preceding chapters have delved into research questions in several important areas of user surveillance. While the specific adversaries and goals of each varied, the overall trend of this research shows that modern technological innovation, particularly the Internet, has facilitated increasing surveillance of individuals. In particular, the cost of this surveillance has decreased to the point that governments can feasibly conduct mass surveillance of their entire populace, while corporations actually find it profitable to do so in an attempt to increase advertising effectiveness. Further, in this pursuit of economical surveillance, the advertising ecosystem has extended the capabilities of modern surveillance down to a price-point that the average individual could easily access if they so chose.

Parts of this dissertation were aimed at building or evaluating defenses against this type of surveillance. However, these efforts have produced mixed results that should be understood fully. In the first case, Rook does provide robust defense against many types of surveillance adversary. However, it is also a fairly burdensome tool to use compared to baseline messaging programs. The kinds of trade-off required for Rook use are common across circumvention systems: evading state surveillance is not free. Overall surveillance evasion of this type is an arms-race, and Rook shows the defensive side can take another step to defeat the current level of surveillor. Unfortunately, Rook does not provide any sort of bound on when this arms-race might end, or whether the defender would necessarily be the winner.

In the second case we examined privacy defenses against current web-tracking. Again our results are mixed: the two most effective defenses have flaws of different types. Clearing browser state after each page visit is clearly impractical for most users; using uBlock Origin is more practical for individual users, but is untenable for truly widespread adoption due

to its impact on advertising-revenue for free sites and services. However, the effectiveness of these defenses does provide evidence that privacy defenses can work. This suggests that further development of defenses that can protect user privacy while preserving advertising revenue is feasible; further, development of such tools is likely the only long-term solution to user privacy.

The final section of this dissertation, the evaluation and systematization of ADINT as an intelligence tool, showed that the low cost of surveillance is bringing advanced capabilities down to well-within in the individual's budget. This is particularly concerning for the future, because there is no way to institutionally hold individuals accountable for their surveillance abuses the way a democracy can constrain its intelligence agencies. The past shows that the advertising industry has continued to develop more pervasive and invasive tools for user surveillance, rather than self-regulating to protect user privacy. While user defenses like those examined in Chapter 3 can reduce some information ADINT uses, many types of sensitive information are gathered from services selling their users' data. This form of privacy invasion is difficult to defend against from the client perspective, because many services have legitimate reasons to request this data from users in order to provide features. For example, a weather app needs to continuously access its user's location to provide notifications about changes in weather. Therefore, governmental regulation on the kinds of individually-accessible surveillance tools, like ADINT, seems the most reasonable answer. These regulations could include placing limits on: how user data can be gathered; how it can be sold; how advertisers are allowed to target users; and how easy it is for an individual to get access to the advertising ecosystem.

This dissertation demonstrates that surveillance in general is increasingly pervasive in what it can gather from targets, and increasingly accessible in who can feasibly access this data. However, both technology and policy can provide solutions to help roll back the tide of surveillance. In Chapters 2 and 3 we found that users can successfully prevent surveillance — metadata communications surveillance and web-tracking, respectively — provided they know the surveillance exists and can access the proper defenses. The cat-and-mouse nature of

surveillance means these technological defenses will need continuous improvement as new techniques are developed. Chapter 4 exposed some greater challenges, because ADINT leverages information users voluntarily provide to services, when this information is subsequently sold. Technological defenses to ADINT activities are trickier; using something like an aggressive adblocker could mitigate portions of ADINT, although it can be difficult to employ an adblocker in all circumstances, particularly in the mobile context. Large organizations could also deploy technology to monitor advertising traffic to its members and detect or potentially prevent ADINT operations before any information is gained. Finally, if adnetworks are willing to self-regulate, then automatic auditing techniques could be developed to try to identify ADINT operations being conducted. Each of these potentially creates its own arms-race situation for sophisticated ADINT operators to circumvent these controls. However, this will raise the bar for ADINT use overall which could prevent many malicious actors from using it. Another approach is more stringent legal regulations on the advertising ecosystem to prevent the kind of information collection and use that enables ADINT.

In conclusion, surveillance is a growing problem that requires the attention of researchers, engineers, and lawmakers to help solve. Surveillance is increasingly pervasive and available, but a combination of technological and legal solutions can stop this trend and recover the privacy of individuals in the future.

BIBLIOGRAPHY

[1] Ruba Abu-Salma, M. Angela Sasse, Joseph Bonneau, Anastasia Danilova, Alena Naiakshina, and Matthew Smith. Obstacles to the Adoption of Secure Communication Tools. In *IEEE Symposium on Security and Privacy*, 2017.

[2] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The Web Never Forgets: Persistent Tracking Mechanisms in the Wild. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2014.

[3] Ross J Anderson and Fabien AP Petitcolas. On the Limits of Steganography. *Selected Areas in Communications, IEEE Journal on*, 16(4):474–481, 1998.

[4] AVG Technologies. Crumble. `http://innovation.avg.com/projects/crumble/`.

[5] AVG Technologies. Introducting Crumble - Surf Without Surveillance. `http://now.avg.com/introducing-crumble/`.

[6] Mika Ayenson, Dietrich James Wambach, Ashkan Soltani, Nathan Good, and Chris Jay Hoffnagle. Flash Cookies and Privacy II: Now with HTML5 and ETag Respawning. *Social Science Research Network Working Paper Series*, 2011.

[7] Rebecca Balebako, Pedro Leon, Richard Shay, Blase Ur, Yang Wang, and L Cranor. Measuring the Effectiveness of Privacy Tools for Limiting Behavioral Advertising. In *Web 2.0 Security and Privacy*, 2012.

[8] Tom Balmforth. In Russia, Violent Videos Show a Startling New Form of Gay Bullying. *The Atlantic*, August 2013. `https://www.theatlantic.com/international/archive/2013/08/in-russia-violent-videos-show-a-startling-new-form-of-gay-bullying/278294/`.

[9] Muhammad Ahmad Bashir, Sajjad Arshad, William Robertson, and Christo Wilson. Tracing Information Flows Between Ad Exchanges Using Retargeted Ads. In *Proceedings of the 25th USENIX Security Symposium*, 2016.

[10] Jason Bau, Jonathan Mayer, Hristo Paskov, and John C. Mitchell. A Promising Direction for Web Tracking Countermeasures. In *Web 2.0 Security and Privacy*, 2013.

[11] Hal Berghel. Caustic Cookies, 2001. `http://www.berghel.net/col-edit/digital_village/apr-01/dv_4-01.pdf`.

[12] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-Record Communication, Or, Why Not to Use PGP. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 77–84. ACM, 2004.

[13] Ron Bowes. DNS Cat. 2014.

[14] Interactive Advertising Bureau. IAB Interactive Advertising Wiki, 2017. `https://wiki.iab.com/index.php/Category:Glossary`.

[15] Juan Miguel Carrascosa, Jakub Mikians, Ruben Cuevas, Vijay Erramilli, and Nikolaos Laoutaris. I Always Feel Like Somebody's Watching me: Measuring Online Behavioural Advertising. In *CoNEXT*, 2015.

[16] Giuseppe Cattaneo, Giancarlo De Maio, Pompeo Faruolo, and Umberto Ferraro Petrillo. A Review of Security Attacks on the GSM Standard. In *Information and Communication Technology-EurAsia Conference*, pages 507–512. Springer, 2013.

[17] Doug Bock Clark. The Bot Bubble. 2015.

[18] Robert M Clark. Perspectives on Intelligence Collection. *The intelligencer. Journal of US Intelligence Studies*, 20(2):47–53, 2013.

[19] Richard Clayton, Steven J Murdoch, and Robert NM Watson. Ignoring the Great Firewall of China. In *Privacy Enhancing Technologies*, pages 20–35. Springer, 2006.

[20] Cindy Cohn. The Burr-Feinstein Proposal Is Simply Anti-Security. *Electronic Frontier Foundation*, April 2016. `https://www.eff.org/deeplinks/2016/04/burr-feinstein-proposal-simply-anti-security`.

[21] David Cole. We Kill People Based on Metadata. *The New York Review of Books*,

10:2014, 2014.

[22] Jonathan Crussell, Ryan Stevens, and Hao Chen. Madfraud: Investigating Ad Fraud in Android Applications. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 123–134. ACM, 2014.

[23] Amit Datta, Michael Carl Tschantz, and Anupam Datta. Automated Experiments on Ad Privacy Settings. In *Proceedings on Privacy Enhancing Technologies*, 2015.

[24] Jacob Davidson. Here's How Many Internet Users There Are. 2015. `http://time.com/money/3896219/internet-users-worldwide/`.

[25] Doug DePerry, Tom Ritter, and Andrew Rahimi. Cloning with a Compromised CDMA Femtocell. 2013. `https://www.defcon.org/images/defcon-21/dc-21-presentations/DePerry-Ritter/DEFCON-21-DePerry-Ritter-Femtocell-Updated.pdf`.

[26] Google Developers. Google Ads, 2017. `https://developers.google.com/ads/`.

[27] Roger Dingledine. Obfsproxy. 2014.

[28] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. Technical report, DTIC Document, 2004.

[29] Dante D'Orazio. China Passes Controversial Anti-Terrorism Law to Access Encrypted User Accounts. *The Verge*, December 2015. `https://www.theverge.com/2015/12/27/10670346/china-passes-law-to-access-encrypted-communications`.

[30] Peter Eckersley. How Unique Is Your Web Browser? In *Proceedings of the International Conference on Privacy Enhancing Technologies*, 2010.

[31] Electronic Frontier Foundation. Privacy Badger. `https://www.eff.org/privacybadger`.

[32] Steven Englehardt and Arvind Narayanan. Online Tracking: A 1-Million-Site Measurement and Analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1388–1401. ACM, 2016.

[33] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W Felten. Cookies That Give You Away: The

Surveillance Implications of Web Tracking. In *Proceedings of the 24th International Conference on World Wide Web*, pages 289–299. ACM, 2015.

[34] fanboy, MonztA, Famlam, and Khrin. EasyPrivacy. `https://easylist.to/easylist/easyprivacy.txt`.

[35] Julia Fioretti. EU-U.S. Commercial Data Transfer Pact Clears Final Hurdle. *Reuters*, 2016. `http://www.reuters.com/article/us-eu-dataprotection-usa-idUSKCN0ZO0SH`.

[36] Mark Follman. The Terror Attacks Trump Won't Talk About. *Mother Jones*, February 2017. `http://www.motherjones.com/politics/2017/02/terror-attacks-trump-wont-talk-about-white-supremacists`.

[37] Nathaniel Fruchter, Hsin Miao, Scott Stevenson, and Rebecca Balebako. Variations in Tracking in Relation to Geographic Location. *arXiv preprint arXiv:1506.04103*, 2015.

[38] John Geddes, Max Schuchard, and Nicholas Hopper. Cover Your ACKs: Pitfalls of Covert Channel Censorship Circumvention. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 361–372. ACM, 2013.

[39] Barton Gellman and Laura Poitras. U.S., British Intelligence Mining Data From Nine U.S. Internet Companies in Broad Secret Program. *The Washington Post*, June 2013.

[40] John Giffin, Rachel Greenstadt, Peter Litwack, and Richard Tibbetts. Covert Messaging Through TCP Timestamps. In *Privacy Enhancing Technologies*, pages 194–208. Springer, 2003.

[41] Eyeo GmbH. Chrome Webstore: AdBlock Plus. `https://chrome.google.com/webstore/detail/adblock-plus/cfhdojbkjhnklbpkdaibdccddilifddb`.

[42] Go2mobi, 2017. `https://www.go2mobi.com`.

[43] Rafi Goldberg. Lack of Trust in Internet Privacy and Security May Deter Economic and Other Online Activities. *National Telecommunications & Information Administration*, 2016.

[44] Richard Gomer, Eduarda Mendes Rodrigues, Natasa Milic-Frayling, and MC Schraefel. Network Analysis of Third Party Tracking: User Exposure to Tracking Cookies

Through Search. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on*, volume 1, pages 549–556. IEEE, 2013.

[45] Google. Clear, Enable, and Manage Cookies in Chrome.

[46] Google. Targeting Tools. `https://www.google.com/ads/displaynetwork/manage-your-ads/targeting-tools.html`.

[47] Glenn Greenwald, James Ball, and Dominic Rushe. NSA Prism Program Taps in to User Data of Apple, Google and Others. *The Guardian*, June 2013.

[48] Bridger Hahn, Rishab Nithyanand, Philippa Gill, and Rob Johnson. Games Without Frontiers: Investigating Video Games as a Covert Channel. *arXiv preprint arXiv:1503.05904*, 2015.

[49] Kashmir Hill. Edward Snowden Implores Tech Community To Build More Secure Products. *Forbes*, March 2014. `https://www.forbes.com/sites/kashmirhill/2014/03/10/edward-snowden-implores-tech-community-to-build-more-secure-products`.

[50] Raymond Hill. uBlock. `https://github.com/gorhill/uBlock`.

[51] Raymond Hill. uBlock and others: Blocking ads, trackers, malwares. `https://github.com/gorhill/uBlock/wiki/uBlock-and-others\%3A-Blocking-ads\%2C-trackers\%2C-malwares`.

[52] Sture Holm. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[53] Amir Houmansadr and Nikita Borisov. CoCo: Coding-Based Covert Timing Channels for Network Flows. In *Information Hiding*, pages 314–328. Springer, 2011.

[54] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. The Parrot Is Dead: Observing Unobservable Network Communications. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 65–79. IEEE, 2013.

[55] Amir Houmansadr, Thomas Riedl, Nikita Borisov, and Andrew Singer. I Want My Voice to Be Heard: IP Over Voice-Over-IP for Unobservable Censorship Circum-

vention. In *The 20th Annual Network and Distributed System Security Symposium (NDSS)*, 2013.

[56] Docker Inc. https://www.docker.com/.

[57] A. Janc and L. Olejnik. Feasibility and Real-World Implications of Web Browser History Detection. In *IEEE Workshop on Web 2.0 Security and Privacy*, 2010.

[58] Sam Jones. Muslim vigilantes jailed for 'sharia law' attacks in London . *The Washington Post*, December 2013. https://www.theguardian.com/uk-news/2013/dec/06/muslim-vigilantes-jailed-sharia-law-attacks-london.

[59] Samy Kamkar. Evercookie — Virtually Irrevocable Persistent Cookies. http://samy.pl/evercookie/.

[60] Brian Kennish. Disconnect. https://disconnect.me/.

[61] Sheharbano Khattak, Laurent Simon, and Steven J Murdoch. Systemization of Pluggable Transports for Censorship Resistance. *arXiv preprint arXiv:1412.7448*, 2014.

[62] Jim Killock. Sleepwalking Into Censorship. *Open Rights Group*, July 2013.

[63] Ben Kneen. SSP to DSP Cookie Syncing Explained. http://www.adopsinsider.com/ad-exchanges/cookie-syncing/.

[64] Ben Kneen. Data Management Platform: What is a DMP? 2011. http://www.adopsinsider.com/data-management-platform/what-is-a-data-management-platform/.

[65] Aleksandra Korolova. Privacy Violations Using Microtargeted Ads: A Case Study. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 474–482. IEEE, 2010.

[66] Balachander Krishnamurthy and Craig E. Wills. Generating a Privacy Footprint on the Internet. In *Proceedings of the ACM Internet Measurement Conference*, 2006.

[67] D. Kristol and L. Montulli. RFC 2965: HTTP State Management Mechanism.

[68] Adrienne Lafrance. The Convience-Surveillance Tradeoff. *The Atlantic*, 2016. https://www.theatlantic.com/technology/archive/2016/01/the-convenience-surveillance-tradeoff/423891/.

[69] Mathias Lécuyer, Guillaume Ducoffe, Francis Lan, Andrei Papancea, Theofilos Petsios, Riley Spahn, Augustin Chaintreau, and Roxana Geambasu. XRay: Enhancing the Web's Transparency with Differential Correlation. In *23rd USENIX Security Symposium*, 2014.

[70] Mathias Lécuyer, Riley Spahn, Yannis Spiliopolous, Augustin Chaintreau, Roxana Geambasu, and Daniel Hsu. Sunlight: Fine-grained Targeting Detection at Scale with Statistical Confidence. In *22nd ACM Conference on Computer and Communications Security*, 2015.

[71] Pedro G. Leon, Blase Ur, Yang Wang, Manya Sleeper, Rebecca Balebako, Richard Shay, Lujo Bauer, Mihai Christodorescu, and Lorrie Faith Cranor. What Matters to Users? Factors that Affect Users' Willingness to Share Information with Online Advertisers. In *Symposium on Usable Privacy and Security*, 2013.

[72] Shuai Li, Mike Schliep, and Nick Hopper. Facet: Streaming over videoconferencing for censorship circumvention. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 163–172. ACM, 2014.

[73] Zhou Li, Kehuan Zhang, Yinglian Xie, Fang Yu, and XiaoFeng Wang. Knowing Your Enemy: Understanding and Detecting Malicious Web Advertising. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 674–686. ACM, 2012.

[74] Nicolas Lidzborski. Staying at the Forefront of Email Security and Reliability: HTTPS-Only and 99.978 Percent Availability , 2014. `https://googleblog.blogspot.co.uk/2014/03/staying-at-forefront-of-email-security.html`.

[75] Yehuda Lindell and Eran Omri. A Practical Application of Differential Privacy to Personalized Online Advertising. *IACR Cryptology EPrint Archive*, 2011:152, 2011.

[76] David Llamas, C Allison, and A Miller. Covert Channels in Internet Protocols: A Survey. In *Proceedings of the 6th Annual Postgraduate Symposium about the Convergence of Telecommunications, Networking and Broadcasting, PGNET*, volume 2005, 2005.

[77] Natasha Lomas. UK Surveillance Law Still Fuzzy on Decryption Rules for

Comms Providers. *Tech Crunch*, May 2017. https://techcrunch.com/2017/05/05/uk-surveillance-law-still-fuzzy-on-decryption-rules-for-comms-providers/.

[78] Norka B Lucena, James Pease, Payman Yadollahpour, and Steve J Chapin. Syntax and Semantics-Preserving Application-Layer Protocol Steganography. In *Information Hiding*, pages 164–179. Springer, 2005.

[79] Jianming Lv, Tieying Zhang, Zhenhua Li, and Xueqi Cheng. PACOM: Parasitic Anonymous Communication in the BitTorrent Network. *Computer Networks*, 2014.

[80] Steve Mansfield-Devine. When Advertising Turns Nasty. *Network Security*, 2015(11):5–8, 2015.

[81] Jonathan Mayer. Tracking the Trackers: Self Help Tools, September 2011. http://cyberlaw.stanford.edu/blog/2011/09/tracking-trackers-self-help-tools.

[82] Jonathan Mayer and Arvind Narayanan. Do Not Track. http://donottrack.us/.

[83] Jonathan R. Mayer and John C. Mitchell. Third-Party Web Tracking: Policy and Technology. In *IEEE Symposium on Security and Privacy*, 2012.

[84] Wojciech Mazurczyk and Krzysztof Szczypiorski. Steganography of VoIP streams. In *On the Move to Meaningful Internet Systems: OTM 2008*, pages 1001–1018. Springer, 2008.

[85] Aleecia M. McDonald and Lorrie Faith Cranor. Americans' Attitudes about Internet Behavioral Advertising Practices. In *Proceedings of the Workshop on Privacy in the Electronic Society*, 2010.

[86] Jeffrey Meisner. Advancing Our Encryption and Transparency Efforts, 2014. https://blogs.technet.microsoft.com/microsoft_on_the_issues/2014/07/01/advancing-our-encryption-and-transparency-efforts/.

[87] Inc. Merriam-Webster, 2017. https://www.merriam-webster.com/dictionary/intelligence.

[88] Georg Merzdovnik, Markus Huber, Damjan Buhov, Nick Nikiforakis, Sebastian Neuner, Martin Schmiedecker, and Edgar Weippl. Block Me If You Can: A Large-Scale Study of Tracker-Blocking Tools. In *Proceedings of the 2nd IEEE European Symposium*

on Security and Privacy, 2017.

[89] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. SkypeMorph: Protocol Obfuscation for Tor Bridges. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 97–108. ACM, 2012.

[90] Keaton Mowery and Hovav Shacham. Pixel Perfect: Fingerprinting Canvas in HTML5. *Proceedings of W2SP*, 2012.

[91] Martin Mulazzani, Philipp Reschl, Markus Huber, Manuel Leithner, Sebastian Schrittwieser, Edgar Weippl, and FC Wien. Fast and Reliable Browser Identification With Javascript Engine Fingerprinting. In *Web 2.0 Workshop on Security and Privacy (W2SP)*, volume 5, 2013.

[92] Steven J Murdoch and Stephen Lewis. Embedding Covert Channels Into TCP/IP. In *Information Hiding*, pages 247–261. Springer, 2005.

[93] Allan Nienhuis. Blocked 3rd Party Session Cookies in Iframes. `http://www.allannienhuis.com/archives/2013/11/03/blocked-3rd-party-session-cookies-in-iframes/`.

[94] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. Cookieless Monster: Exploring the Ecosystem of Web-based Device Fingerprinting. In *IEEE Symposium on Security and Privacy*, 2013.

[95] Rishab Nithyanand, Sheharbano Khattak, Mobin Javed, Narseo Vallina-Rodriguez, Marjan Falahrastegar, Julia E Powles, Emiliano De Cristofaro, Hamed Haddadi, and Steven J Murdoch. Ad-Blocking and Counter Blocking: A Slice of the Arms Race. *arXiv preprint arXiv:1605.05077*, 2016.

[96] Rick Noack. Could Using Gay Dating App Grindr Get You Arrested in Egypt? 2014. `https://www.washingtonpost.com/news/worldviews/wp/2014/09/12/could-using-gay-dating-app-grindr-get-you-arrested-in-egypt/`.

[97] NPR-Staff. With Ad Blocking Use On The Rise, What Happens To Online Publishers. *National Public Radio*, 2015.

[98] Wladimir Palant. `https://adblockplus.org/`.

[99] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[100] plvines. Rook Github Repository. `https://github.com/plvines/rook`.

[101] Enric Pujol, Oliver Hohlfeld, and Anja Feldmann. Annoyed Users: Ads and Ad-Block Usage in the Wild. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, pages 93–106. ACM, 2015.

[102] Björgvin Ragnarsson and Pieter Westein. Using Git to Circumvent Censorship of Access to the Tor Network. 2013.

[103] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and Defending Against Third-Party Tracking on the Web. In *USENIX Symposium on Networked Systems Design and Implementation*, 2012.

[104] Franziska Roesner, Chris Rovillos, Alisha Saxena, and Tadayoshi Kohno. TrackingObserver: A Browser-Based Web Tracking Detection Platform. `http://trackingobserver.cs.washington.edu/`.

[105] Matthew Ruffell, Jin B Hong, and Dong Seong Kim. Analyzing the Effectiveness of Privacy Related Add-ons Employed to Thwart Web Based Tracking. In *Dependable Computing (PRDC), 2015 IEEE 21st Pacific Rim International Symposium on*, pages 264–272. IEEE, 2015.

[106] Rowan Scarborough. ISIS Hit List Targets 700 U.S. Army Soldiers: 'Kill the Dogs'. *The Washington Times*, 2016. `http://www.washingtontimes.com/news/2016/aug/2/isis-hit-list-targets-700-us-army-soldiers/`.

[107] Taeshik Sohn, Jongsub Moon, Sangjin Lee, Dong Hoon Lee, and Jongin Lim. Covert Channel Detection in the ICMP Payload Using Support Vector Machine. In *Computer and Information Sciences-ISCIS 2003*, pages 828–835. Springer, 2003.

[108] Taeshik Sohn, JungTaek Seo, and Jongsub Moon. A Study on the Covert Channel Detection of TCP/IP Header Using Support Vector Machine. In *Information and*

Communications Security, pages 313–324. Springer, 2003.

[109] Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagle. Flash Cookies and Privacy. *Social Science Research Network Working Paper Series*, August 2009.

[110] Sooel Son, Daehyeok Kim, and Vitaly Shmatikov. What Mobile Ads Know About Mobile Users. In *Proc. 23rd Annual Network and Distributed System Security Symposium (2016)*, 2016.

[111] Mark Joseph Stern. This Daily Beast Grindr Stunt Is Sleazy, Dangerous, and Wildly Unethical. 2016. `http://www.slate.com/blogs/future_tense/2016/08/11/the_daily_beast_s_olympics_grindr_stunt_is_dangerous_and_unethical.html`.

[112] Ryan Stevens, Clint Gibler, Jon Crussell, Jeremy Erickson, and Hao Chen. Investigating User Privacy in Android Ad Libraries. In *Workshop on Mobile Security Technologies (MoST)*, page 10, 2012.

[113] Philip H Swain and Hans Hauska. The Decision Tree Classifier: Design and Potential. *IEEE Transactions on Geoscience Electronics*, 15(3):142–147, 1977.

[114] Open Whisper Systems. Signal Private Messenger, 2017. `https://whispersystems.org/`.

[115] Blase Ur, Pedro Giovanni Leon, Lorrie Faith Cranor, Richard Shay, and Yang Wang. Smart, Useful, Scary, Creepy: Perceptions of Online Behavioral Advertising. In *8th Symposium on Usable Privacy and Security*, 2012.

[116] Valve. Steam and Game Stats.

[117] Valve. Team Fortress 2.

[118] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable Private Messaging Resistant to Traffic Analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152. ACM, 2015.

[119] Ratko Vidakovic. The Mechanics of Real-Time Bidding, 2013. `http://marketingland.com/the-mechanics-of-real-time-bidding-31622`.

[120] Paul Vines and Tadayoshi Kohno. Rook: Using Video Games as a Low-Bandwidth

Censorship Resistant Communication Platform. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society*, pages 75–84. ACM, 2015.

[121] Curtis Waltman. Rochester Police Release Unredacted List of Harris Corp StingRay and KingFish Products. *Muckrock*, 2017. https://www.muckrock.com/news/archives/2016/dec/07/rochester-police-release-unredacted-list-harris-co/.

[122] Qiyan Wang, Xun Gong, Giang TK Nguyen, Amir Houmansadr, and Nikita Borisov. Censorspoofer: asymmetric communication using ip spoofing for censorship-resistant web browsing. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 121–132. ACM, 2012.

[123] Barney Warf and Peter Vincent. Multiple Geographies of the Arab Internet. *Area*, 39(1):83–96, 2007.

[124] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. Stegotorus: a camouflage proxy for the tor anonymity system. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 109–120. ACM, 2012.

[125] Craig E Wills and Doruk C Uzunoglu. What Ad Blockers Are (and Are Not) Doing. In *Hot Topics in Web Systems and Technologies (HotWeb), 2016 Fourth IEEE Workshop on*, pages 72–77. IEEE, 2016.

[126] Craig E Wills and Doruk C Uzunoglu. What Ad Blockers Are (and Are Not) Doing. In *Hot Topics in Web Systems and Technologies (HotWeb), 2016 Fourth IEEE Workshop on*, pages 72–77. IEEE, 2016.

[127] Philipp Winter and Stefan Lindskog. How the great firewall of china is blocking tor. In *FOCI*, 2012.

[128] Charles V Wright, Lucas Ballard, Fabian Monrose, and Gerald M Masson. Language Identification of Encrypted VoIP Traffic: Alejandra Y Roberto or Alice and Bob? In *USENIX Security*, volume 3, page 3, 2007.

[129] Xueyang Xu, Z Morley Mao, and J Alex Halderman. Internet censorship in china:

Where does the filtering occur? In *International Conference on Passive and Active Network Measurement*, pages 133–142. Springer, 2011.

[130] Tom Yeh, Tsung-Hsiang Chang, and Robert C Miller. Sikuli: using gui screenshots for search and automation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 183–192. ACM, 2009.

[131] Ting-Fang Yen, Yinglian Xie, Fang Yu, Roger Peng Yu, and Martin Abadi. Host fingerprinting and tracking on the web: Privacy and security implications. In *NDSS*, 2012.

[132] Zhonghao Yu, Sam Macbeth, Konark Modi, and Josep M Pujol. Tracking the Trackers. In *Proceedings of the 25th International Conference on World Wide Web*, pages 121–132. International World Wide Web Conferences Steering Committee, 2016.

[133] Apostolis Zarras, Alexandros Kapravelos, Gianluca Stringhini, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. The dark alleys of madison avenue: Understanding malicious advertisements. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 373–380. ACM, 2014.

[134] Tiliang Zhang, Hua Zhang, and Fei Gao. A malicious advertising detection scheme based on the depth of url strategy. In *Computational Intelligence and Design (ISCID), 2013 Sixth International Symposium on*, volume 2, pages 57–60. IEEE, 2013.

[135] Wenxuan Zhou, Amir Houmansadr, Matthew Caesar, and Nikita Borisov. SWEET: Serving the Web by Exploiting Email Tunnels. *HotPETS. Springer*, 2013.

[136] Peter Thomas Zimmerman. Measuring Privacy, Security, and Censorship Through the Utilization of Online Advertising Exchanges. Technical report, Princeton University, 2015.