Hershel: Single-Packet OS Fingerprinting

Zain Shamsi, Ankur Nandwani, Derek Leonard, and Dmitri Loguinov

Internet Research Lab
Department of Computer Science and Engineering
Texas A&M University

June 18, 2014

Agenda

- Introduction
- Background
- Building Hershel
- Simulations
- Internet Scan

Introduction

- The goal of OS fingerprinting is to determine OS of a remote host based on its network behavior
- Stack differentiation is possible due to:
  - Unclear language and lack of response standardization in IETF RFCs
  - No mandated behavior for malformed requests
  - Broken (non-compliant) implementations
- Network administrators and industry analysts have used OS fingerprinting as a tool
  - Identify and secure devices in own network
  - Market analysis of OS usage

Introduction

- Internet measurement studies are important to researchers
  - Detect vulnerabilities
  - Show deployment of new software and protocols
- Scans have become progressively faster
  - 30 days, 1K pps [Heidemann 2008]
  - 24 hours, 24K pps [Leonard 2010]
  - 45 minutes, 1.4M pps [Durumeric 2013]
- Large-scale measurement tools need to be fast, low overhead, and accurate
  - OS fingerprinting at large scale has not been explored before, which is our topic here

Agenda

- Introduction
- Background
- Building Hershel
- Simulations
- Internet Scan

Background

- Active OS fingerprinting typically requires open port
- Rooted in banner grabbing, which has many drawbacks
  - Protocol must be known
  - High overhead
  - Defeated by generic software (e.g., Apache)
  - Admins can also remove/obfuscate OS-identifying strings

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html;
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: 15 Jun 2014 20:06:22
Connection: close
Content-Length: 20559
```

- Nmap is the current state of the art
  - Database of over 4K different OSes
  - Default 1032 probes per target, but no less than 38 in the least-verbose mode

Background

- Why not use Nmap?
  - Not a polite tool, generates complaints
  - Sends malformed probes, performs vertical port scans
  - Slow, infeasible for large scale
  - Packets easily blocked by IDS such as snort
- Therefore, a more subtle approach is needed
  - p0f, RING, Snacktime are single-packet tools
  - Use header fields and timing of SYN-ACKs
  - Have small OS fingerprint databases (~20 different stacks)
  - Inaccurate when features change (e.g., packet loss)
- As a result, the issue of low-overhead and accurate fingerprinting remains open

Agenda

- Introduction
- Background
- Building Hershel
- Simulations
- Internet Scan

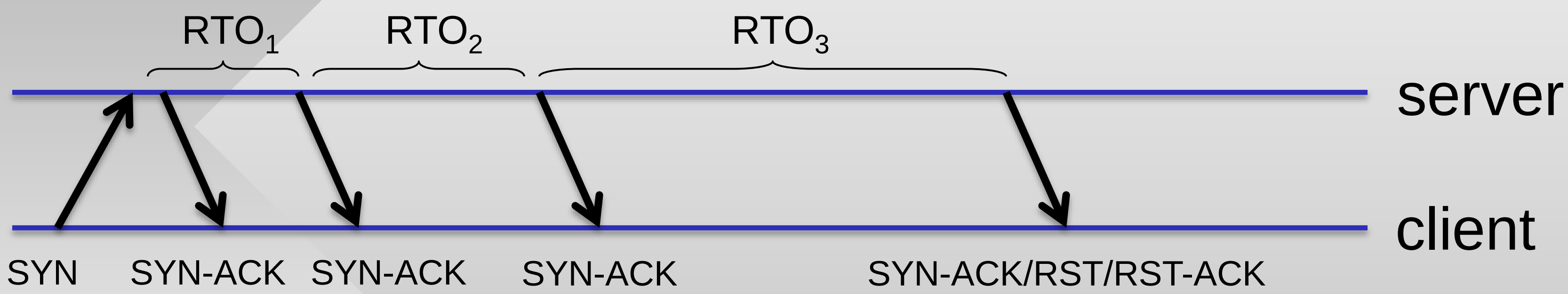
Computer Science, Texas A&M University

Building Hershel

- Our aim is to build a single-packet tool that is robust to network and user modification
  - "Single-packet" means one outbound probe, but multiple responses from the remote OS are allowed
- Assume remote host responds to TCP SYN
  - Specific port/protocol does not matter
  - A SYN probe provides minimal intrusiveness, along with non-malicious operation
- Suppose each OS j can be described by some fingerprint vector y_j
  - Consists of two types of features – network and user

Building Hershel

- Network features are SYN-ACK RTOs



- Examples:

OS	SYN-ACK RTO	Reset RTO
Windows 7	3 6	12
Mac OSX 10.3	2.92 6 12 24	30
NetBSD 4.0	2.92 6 12 24	-
Juniper Netscreen	1.67 2 2 2 2 2 2 2	2
Huawei Embedded	0.7 1 1.2 3 4 5	-

Building Hershel

- User features are values taken from packet header fields

Receiver window (TCP)

Time-to-live field (IP)

Do-not-fragment flag (IP)

Options (TCP)

Maximum segment size (TCP)

vector (RST present, RST ACK, RST Seq, RST Win)

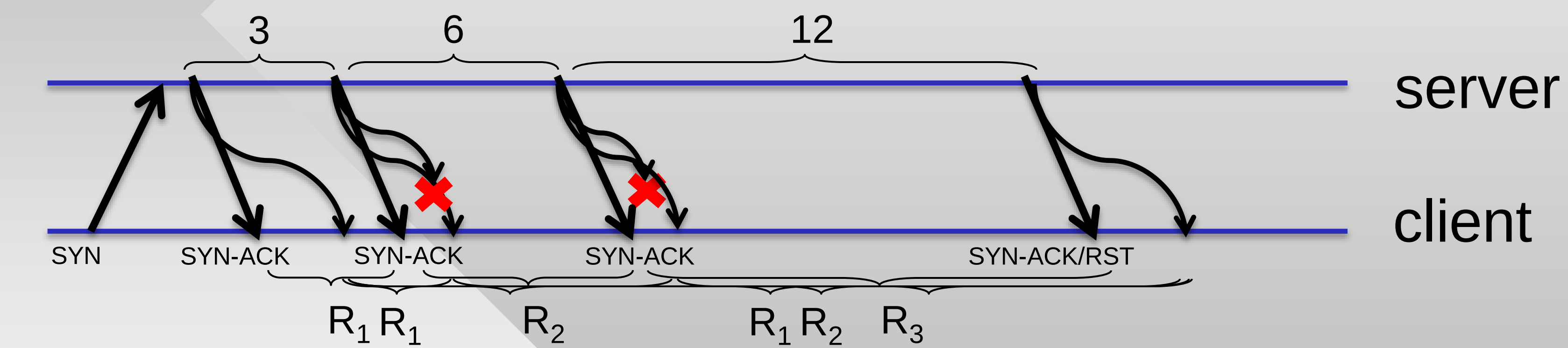
OS	Win	TTL	DF	OPT	MSS	RST
Windows 7	8192	128	1	MNWST	1460	1,0,1,0
Mac OSX 10.3	33304	64	1	MNWNNT	1460	1,1,1,32768
NetBSD 4.0	32768	64	1	MNWNNTSNN	1460	0,-,-,-
Juniper Netscreen	8192	64	0	M	1380	1,0,0,8192
Huawei Embedded	1536	255	0	M	768	0,-,-,-

M = MSS, **N** = NOP, **W** = Window Scale, **S** = Selective ACK, **T** = Timestamp

never used before

Building Hershel

- Challenges
  - One-way delay (OWD) jitter (usually zero-mean)
  - Packet loss



With OWD	1 packet lost	2 packets lost	3 packets lost
(2.8,6.4,12.1)	(9.2, 12.1)	(21.3)	empty
	(2.8, 18.5)	(6.4)	
	(2.8, 6.4)	(18.5)	
	(6.4, 12.1)	(9.2)	
		(12.1)	
		(2.8)	

Not just many possibilities, but also drastically different values!

Building Hershel

- Challenges (cont'd)
  - User modification of default TCP/IP parameters (e.g., OS tuning software, fingerprint scrubbers, NAT, IDS)
  - Unlike OWD, these result in arbitrary value fluctuations
  - Example: Window size is more likely to jump from 8,192 to 65,535 than to 8,193
- Treating all features as volatile, an observed sample can match pretty much any OS

Fingerprint	Win	TTL	DF	OPT	MSS	RST	RTO
Observed	65535	64	1	MNW	1460	1,1,0,0	2.8 6.4
Windows 7	8192	128	1	MNWST	1460	1,0,1,0	3 6 12
Mac OSX	33304	64	1	MNWNNT	1460	1,1,1,32768	2.9 6 12 24 30

Building Hershel

- Thus, any observation x can be viewed as a distortion of each original fingerprint y_j from underlying OS j

- Given a sample x, our goal is to determine the most probable y_j that could have produced it:

$$s(x) = \arg\max p(y_j|x)$$

probability that observation x comes from OS j

- Which is equivalent to:

$$s(x) = \arg\max p(x|y_j)p(y_j)$$

probability that y_j became distored into x

fraction of hosts running OS j

Building Hershel

- To obtain these probabilities, we need a new model
  - Machine learning techniques don't work due to lossy features
- We develop a stochastic theory of single-packet fingerprinting to account for these random effects
  - See paper for details
- We then build a classifier called Hershel, which can additionally handle OSes with random feature vectors, and construct a database of 116 OSes
- Can distinguish not only between OS families (Windows, Linux, FreeBSD, embedded devices), but also patch levels (SP1 vs SP2)

Agenda

- Introduction
- Background
- Building Hershel
- Simulations
- Internet Scan

Simulations

- Emulate a FIFO queue between server and client
  - Run simulations to classify 2^{18} IP samples with random network/user modifications
  - Vary packet loss and user feature modification from 0 to 50%
- First, we perform comparison with Snacktime, which is the most accurate previous single-packet tool
  - Uses only RTO and Win/TTL (Pareto OWD, mean 0.5 sec)

		RTO only accuracy		+Win/TTL accuracy	
Loss	**Feature mod**	**Snacktime**	**Hershel**	**Snacktime**	**Hershel**
0%	0%	12%	22%	58%	86%
3.8%	10%	10%	21%	44%	78%
10%	10%	7%	20%	33%	76%
50%	50%	0.8%	10%	2%	28%

Simulations

- Hershel's RTO classifier doubles Snacktime accuracy at low loss, triples at 10%, and improves an order of magnitude at 50% loss
  - However, Hershel works even better with new features

Hershel accuracy, using Pareto OWD (mean 0.5 sec)							
Loss	**Feature mod**	**RTO Only**	**+Win/TTL**	**+DF**	**+TCP OPT**	**+MSS**	**+RST**
0%	0%	22%	86%	89%	96%	99%	99.9%
3.8%	10%	21%	77%	79%	91%	94%	95%
10%	10%	20%	76%	77%	91%	94%	95%
50%	50%	10%	28%	35%	54%	57%	60%

- Numerous other scenarios and delay distributions omitted here, but shown in the paper

Agenda

- Introduction
- Background
- Building Hershel
- Simulations
- Internet Scan

Internet Scan

RTOs	Hosts	Database
3	9.6M	27
2	9.0M	16
5	7.8M	23
4	5.0M	16
1	2.6M	1

- Port-80 SYN scan of the Internet
  - 2.1B IPs in 24 hours, 37.8M responses, 94% with at least one RTO
- Extensive sanity verification of the dataset
  - Not enough room to show here, see the paper
- We see a lot more values for each header field than we have in our dataset
  - Emphasizes the importance of probabilistic matching
- Run Hershel on all hosts and obtain a non-zero matching probability on 37.4M devices

Internet Scan

- Classification results – top 5 OSes and families

OS	Hosts
Linux 2.6 / 2.4	9.6 M
VxWorks Embedded	4.1 M
Windows Server 2003 SP1 SP2	2.3 M
VxWorks 5.4 / Xerox Embedded	1.8 M
Linux 2.6 / Debian / CentOS	1.1 M

Family	Count
Linux	13.8 M
Embedded	13.5 M
Windows	7.5 M
Other (Mac, BSD, Novell, etc)	2.3 M

- Compared to previous application of Snacktime to this dataset [Leonard10], 9M more embedded devices
- Manual verification vs. Snacktime
  - We pick 1000 random hosts to compare classifications
  - When Hershel and Snacktime disagree, 97% of the time Hershel is correct, 1.8% Snacktime, and 1.2% neither

Thank you!

Questions?