Unsupervised Clustering Under Temporal Feature Volatility in Network Stack Fingerprinting

Zain Shamsi and Dmitri Loguinov

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

June 16, 2016

Agenda

- Introduction
- The Plata Algorithm
- OS Fingerprinting Database
- Internet Scan
- Nmap Comparison

Introduction

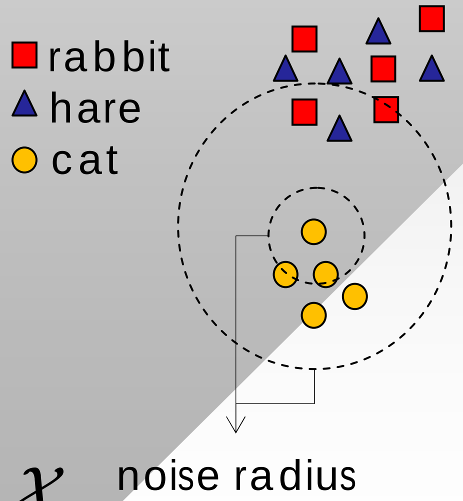
- Classification of large networking datasets is an important topic
- This requires a database of known signatures for a classifier to match against
  - Made of different specimens found in the wild
- Generally, these databases are created manually and must be kept up-to-date
  - Slow process that usually lags behind the discovery of a new specimen
  - Is prone to error, heuristic decisions and poor repeatability

Introduction

- The performance of a classifier depends heavily on the makeup of the underlying database
- E.g., imagine two databases of animal images
  - Database A contains two pictures, 1 rabbit and 1 cat
  - Database B contains 5 rabbits, 5 hares and 5 cats
- Then a classifier trying to identify animals could have:
  - 99% accuracy and a quick runtime using A
  - 40% accuracy and slower performance using B since rabbits/hares are similar, and more comparisons are needed
- Databases should only keep classes which can be differentiated, and drop duplicate specimens

Introduction

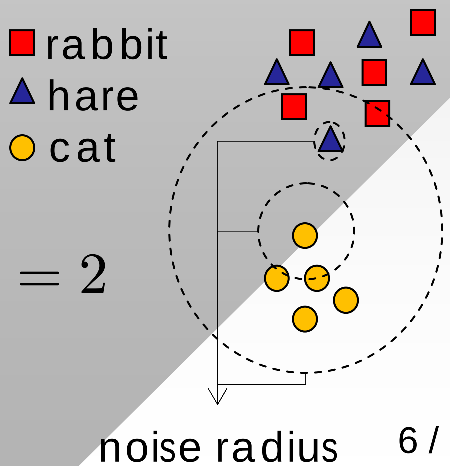
- However, data gathered in the wild has more complex elements in reality
  - Each image captured could additionally be disturbed by noise such as lens distortion or motion blur
  - Lets say this is given by some noise model $\mathcal{X}$
- We now plot the features of animals in database B
  - Under a small noise radius, we cannot tell hares and rabbits apart
  - With larger noise, we may not be able to distinguish cats either
- We want a database that can classify specimens correctly under given noise $\mathcal{X}$



Computer Science, Texas A&M University

Introduction

- We introduce a framework to describe a classifier and its database – its *dimension* $d(1-\epsilon, \mathcal{X})$
  - This means that the classifier can differentiate between d signatures with probability $1-\epsilon$ under noise model $\mathcal{X}$
  - The database for this classifier must then contain exactly d signatures
- We can use this metric to build databases as well as compare the power of classifiers
  - In our example, if we had almost no noise, our dimension $d = 15$
  - Under small noise radius our dimension $d = 2$
  - Under large noise radius it would be reduced to $d = 1$



Introduction

- We define features to be *volatile* if $\mathcal{X}$ can distort them, or *deterministic* otherwise
- Our interest is in the problem of specimen *separation*
  - Deciding whether two observations are too similar for database inclusion
- Separation for deterministic features can be simple
  - E.g. pick every unique combination as a different signature
- For volatile features, we require a more sophisticated solution
  - Our goal is to solve this problem
  - We apply our method to OS fingerprinting

Agenda

- Introduction
- The Plata Algorithm
- OS Fingerprinting Database
- Internet Scan
- Nmap Comparison

The Plata Algorithm

- Consider a measurement of production systems $S_1, \ldots, S_N$ in the wild
  - This produces a set of feature vectors
- Initially, all vectors are added into one large database
  - May contain several duplicates
- We want to determine the separability of the specimens in this database according to noise $\mathcal{X}$
- We introduce an algorithm called Plata, which refines this database and determines its dimension
  - La Plata, Argentina was the first city to use fingerprint databases in 1892

The Plata Algorithm

- Assume Δ is a vector of features and Δ' is feature vector in the database
- Any given classifier will have a function $p(\Delta|\Delta', \mathcal{X})$
  - Produces classification probability for Δ' becoming Δ under $\mathcal{X}$
- Plata uses the classifier to compare all database signatures to each other under simulated noise
- This constructs a confusion matrix M, where each cell is calculated as: $M_{ij} = E[p(\Delta_i + \theta|\Delta_j, \mathcal{X})]$
  - θ is random observation noise driven by model $\mathcal{X}$
  - Generally, Monte-Carlo simulations can be used to determine M_{ij}

The Plata Algorithm

- Note that the diagonal of M signifies the probability of self classification
  - We want signatures that can be matched back to themselves with probability $1-\epsilon$, i.e.

$$E[p(\Delta_i + \theta | \Delta_i, \mathcal{X})] \geq 1 - \epsilon$$

- Plata iteratively eliminates signatures from M starting with the lowest diagonal value
  - Keeps going until all diagonal values are $\geq 1-\epsilon$

0.45	0.19	0.36	0.00
0.05	0.49	0.44	0.01
0.12	0.12	0.72	0.04
0.07	0.05	0.00	0.88

$\longrightarrow$

0.52	0.47	0.01
0.13	0.82	0.04
0.05	0.00	0.95

$\longrightarrow$

0.95	0.05
0.00	1.00

The Plata Algorithm

- Instead of re-running expensive Monte-Carlo simulations after each removal, infer the next matrix
  - Removing system k distributes its classifications proportionally amongst the remaining candidates

$$M_{ij} = M_{ij} + \frac{M_{ij}}{1 - M_{ik}} M_{ik}$$

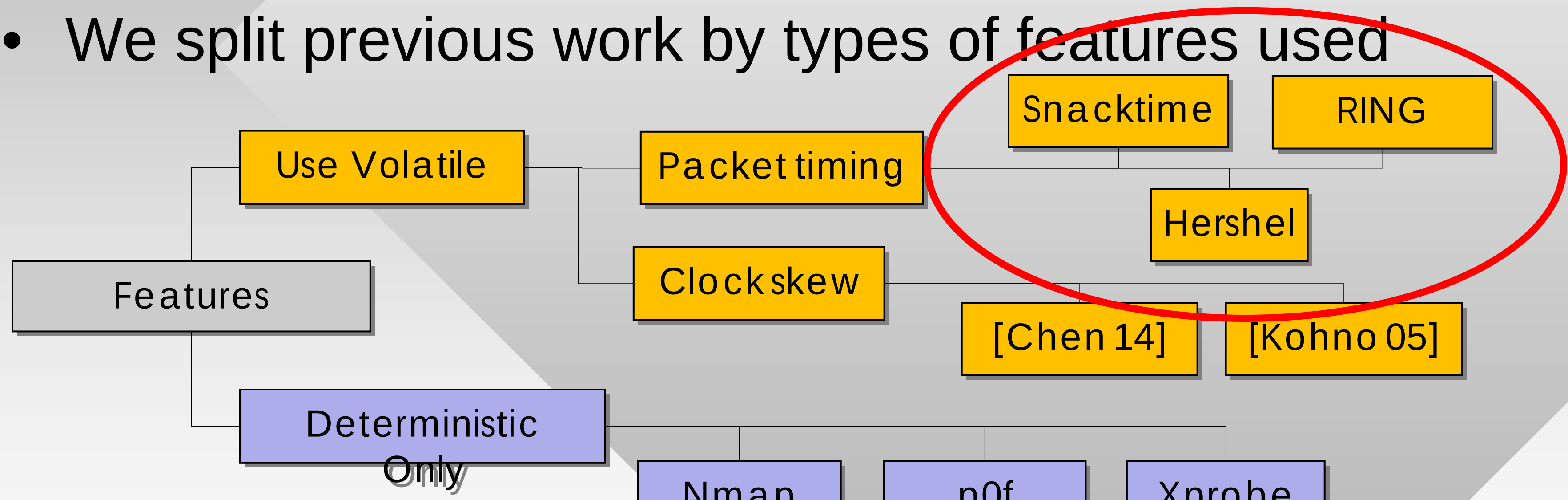
- $1{-}\epsilon$ can be a tuning parameter
  - Higher means smaller database, more uniqueness
  - Lower means larger database, higher risk of duplicates
- For labeling, we only need to know labels for a subset
  - Attach labels to the final clusters based on membership
  - The paper gives more details

Agenda

- Introduction
- The Plata Algorithm
- OS Fingerprinting Database
- Internet Scan
- Nmap Comparison

OS Fingerprinting Database

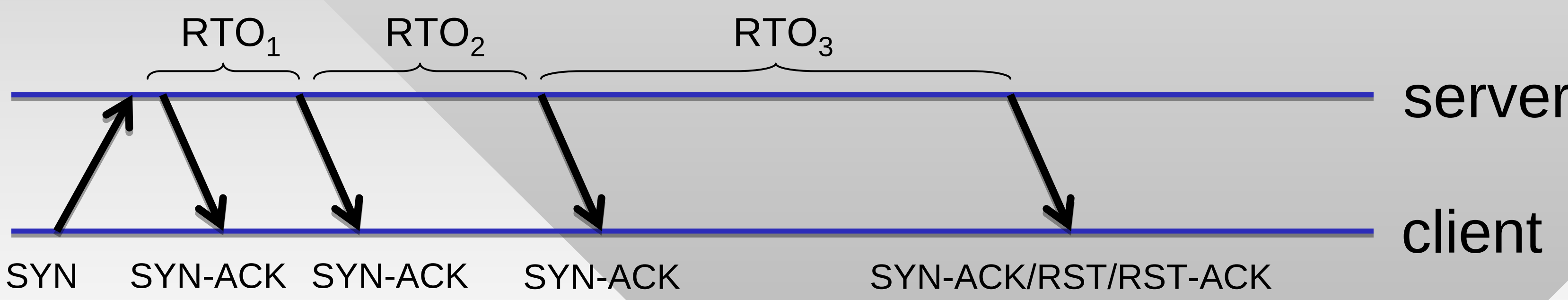
- OS Fingerprinting is a technique to determine the OS of a remote host
- We split previous work by types of features used



- Packet timing techniques
  - Use volatile and deterministic features
  - Low overhead and not intrusive, suitable for large scans

OS Fingerprinting Database

- Deterministic features are values from packet headers
  - TCP Window, IP TTL, TCP Options etc.
- Volatile features are SYN-ACK Retransmission Timeouts (RTOs)



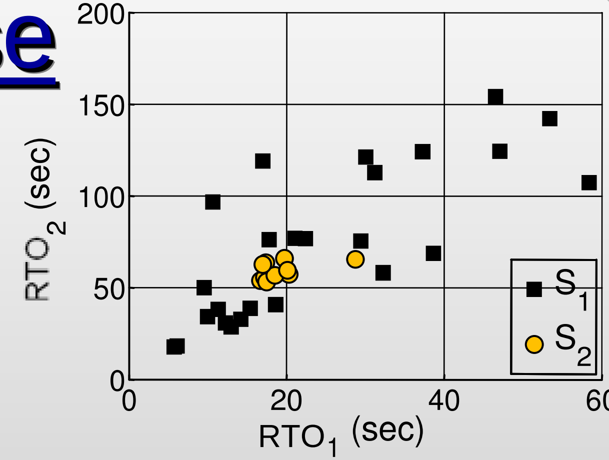
  - Volatile due to network queuing delays

Label	Win	TTL	DF Flag	TCP Options	MSS	SYN-ACK RTOs
Windows 7	8192	128	1	MNWST	1460	3 6 12
Ubuntu	5792	64	1	MSTNW	1460	4.3 6 12 24 48

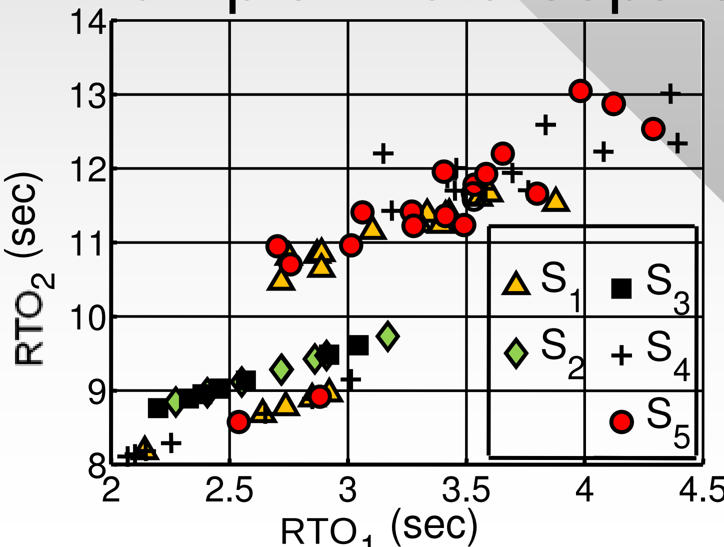
OS Fingerprinting Database

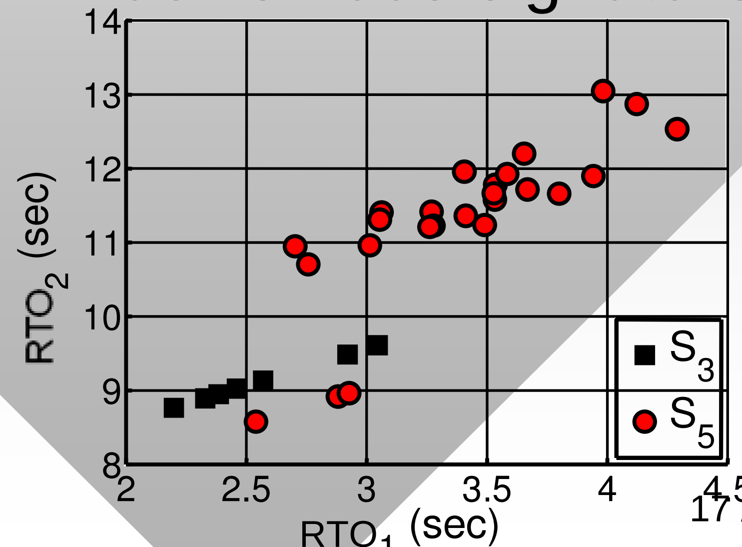
- We scan our campus network on port 80 and gather feature vectors from ~10K systems
- We build databases using three previous packet timing classifiers: RING, Snacktime, and Hershel
- Our data is initially separated using the deterministic features of each classifier
  - Forms several clusters that need to be further separated
- Plata then separates each cluster's volatile features
  - The noise model $\mathcal{X}$ is simulated as a packet queue that adds exponential random one way delay to each packet
  - We also set $1\text{-}\epsilon = 0.8$ to ensure sufficient duplicate elimination

OS Fingerprinting Database



- RTOs are not only volatile, but also random due to OS timers
  - Example shows two Xerox printers
- Most OS signatures have 3-5 RTOs, some with 20
  - Doing this separation manually is practically impossible
- Example: Plata separates 2 Windows 2003 signatures





OS Fingerprinting Database

- We obtain these dimensions for the previous methods:

Grouping	RING	Snacktime	Hershel
Deterministic Features Only	28	209	344
Volatile Features Only	23	52	117
Both	39	260	398

- This allows us to directly compare the power of each classifier in separating the same dataset
  - Hershel is clearly the most powerful method
- For labeling, we use banner grabbing
  - Use simhash to form cluster all hosts of similar OS
  - Match the label clusters to database, see paper for details

OS Fingerprinting Database

- Running Monte-Carlo simulations to determine $E[p(\Delta_i + \theta \mid \Delta_j, \mathcal{X})]$ for each cell can be time consuming
  - Our dataset with 10K hosts takes over 24 hours
  - We want easy repeatability and scalability to larger networks
- We can optimize Plata when the noise model allows the expectation to be calculated directly
- Since Hershel calculates probabilities, we can derive the expected similarity between Δ_i and Δ_j under θ
  - This gives us a closed form for each cell of the Plata matrix
  - The paper develops the model, omitted here

OS Fingerprinting Database

- We also improve on Hershel's classifier, calling the new method Hershel+
  - We switch from using jitter to one-way delay, see paper for a detailed explanation
  - Simulations show it performs up to 10% better than Hershel in RTO classification
- Hershel+ now produces 420 signatures (as opposed to Hershel's 398) for our OS database
  - Improved dimension also confirms its superiority
- Plata's closed form matches Monte-Carlo results, reduces runtime from 24+ hours to 12 minutes

Agenda

- Introduction
- The Plata Algorithm
- OS Fingerprinting Database
- Internet Scan
- Nmap Comparison

Internet Scan

- We performed a port-80 SYN scan of the Internet in July 2015
  - 2.7B IPs in 6 hours, 125K packets / sec
  - 66.4 million hosts responded, almost double the last study
- On the Internet, observed signatures may undergo several changes
  - Deterministic features might be changed by users, firewalls
  - Packet loss and network delays may change RTOs
- We classify all hosts using Hershel+ as the classifier and our database of 420 signatures built using Plata
  - This is the largest study performed with so many signatures and the first with an automated database

Internet Scan

- Linux most popular, for webservers, also a lot of embedded devices
- Comparing with previous results
  - Linux/embedded have doubled
  - Windows has stayed almost the same
  - Misc has lost 69% of its membership
- Paper shows additional results and distributions

Family	OS	Count
Linux 25.67 M (13.88 M)	Ubuntu/Redhat /CentOS	14.55 M
	Ubuntu/Redhat/SUSE	2.62 M
	Ubuntu/Debian/Redhat	2.38 M
Embedded 24.44 M (13.59 M)	3Com Routers	2.66 M
	Dell Laser/Xerox Printers	1.98 M
	Cisco Embedded	1.86 M
Windows 7.12 M (7.56 M)	Windows 7/2008/2012	2.18 M
	Windows XP / 2003	822 K
	Windows 2000 / 2003	791 K
Misc 752 K (2.39 M)	FreeBSD	480 K
	FreeBSD	107 K
	Novell Netware	37 K

Agenda

- Introduction
- The Plata Algorithm
- OS Fingerprinting Database
- Optimization
- Internet Scan
- Nmap Comparison

Nmap Comparison

- We Nmap 1% of responsive hosts from a separate machine at the same time as our scan
- Nmap sends 10 types of probes to the target host
  - Its features are based on existence of responses and values from header fields of each response
- Hershel+ and Nmap find agreement in the OS family in the majority of cases
  - Comparison on the exact OS/device is made difficult due to a large variety of OS names, especially for embedded devices
- However, some cases have glaring disagreements
  - We focus on four such cases between the two classifiers

Nmap Comparison

- Sampled hosts (S1-S4) and their Hershel+ matches:

System / Signature	Win	TTL	TCP Options	MSS	SYN-ACK arrival timestamp
S1	8192	128	MNWST	1464	0.22 3.22 9.22 21.22
S2	8192	64	MNWST	1460	0.18 3.17 9.17
Windows 7/2008 R2	8192	128	MNWST	1464	0.00 2.99 9.00 21.00
S3	16384	128	MNWNNTNNS	1460	0.21 2.67 9.22
S4	16384	128	MNWNNTNNS	1370	0.21 3.07 9.63
Windows 2000/2003	16384	128	MNWNNTNNS	1380	0.00 2.65 9.21

- Hershel+ overcomes packet loss and changes in the TTL/MSS value

- Classifications make sense and are not in doubt

Nmap Comparison

- Nmap signatures of the same four systems:

System / Signature	Win	TTL	TCP Options	Response TCP open port	Response TCP closed port	Response UDP / ICMP
S1	8192	128	MNWST	0000	000	00
Tomato 1.28 (Router firmware)	∅	∅	∅	0000	111	01
S2	8192	64	MNWST	1000	100	11
OpenBSD 4.3 (8 years old)	∅	64	MNNSNWNNT	1000	100	11
S3	16384	128	MNWNNTNNS	0000	000	01
S4	16384	128	MNWNNTNNS	0000	111	01
D-Link DWL624 (2003 54g router)	∅	64	∅	0000	111	01

- Nmap allows null features that match everything, weighs heavily on whether response was received

Nmap Comparison

- Nmap has no provisions for feature volatility, especially in its response vector
  - Becomes an issue when middleboxes block its packets
- The Tomato signature was matched to 21% of hosts in the entire Nmap data
  - Very doubtful for so many hosts to run this firmware
- Nmap's results are questionable on public networks where IDS, packet filters and firewalls are abundant
- Future work will determine the dimension of Nmap

Conclusion

- Introduced Plata – an algorithm for separating observed samples under feature volatility
- Applied Plata to OS classification and automatically built a database of 420 OS signatures
- Developed an improved state-of-the-art classifier, Hershel+, and used it to classify every webserver on the Internet
- Compared our results with Nmap, showing that disagreements tend to favor the Hershel+ result

Thank you!