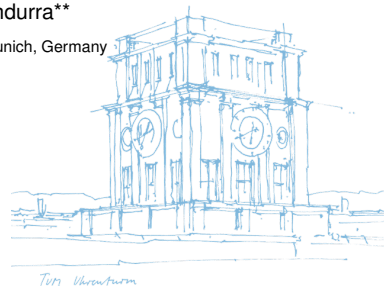
Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Active TLS Stack Fingerprinting: Characterizing TLS Server Deployments at Scale

Markus Sosnowski*, Johannes Zirngibl*, Patrick Sattler*,Georg Carle*,
Claas Grohnfeld**, Michele Russo**, Daniele Sgandurra**

*Chair of Network Architectures and Services, Technical University of Munich, Germany
*AI4Sec, Huawei Technologies Munich, Germany



Introduction

Three facts about the TLS:

1. It is currently the de facto standard for encrypted communication on the Internet[1]
2. It is old, and it has grown to a complex ecosystem due to its continuous development[2]
3. Thus, during the handshake the client and server capabilities must be exchanged.

→ This meta-data allows to fingerprint the TLS stack (config, implementations, and hardware)

[1] C. Labovitz, „Internet Traffic 2009-2019," in Proc. Asia Pacific Regional Internet Conf. Operational Technologies, 2019.

[2] P. Kotzias, A. Razaghpanah, J. Amann u. a., „Coming of Age: A Longitudinal Study of TLS Deployment," in Proc. ACM Int. Measurement Conference (IMC), 2018.

Background

TLS Fingerprinting

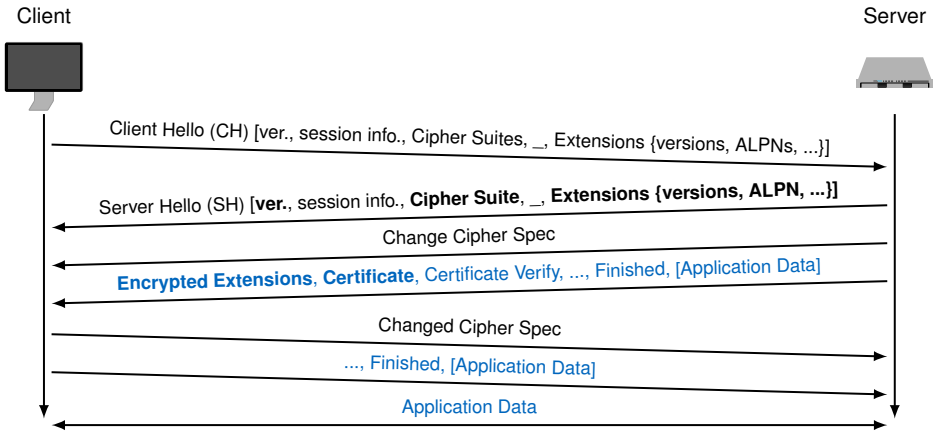
- Collecting TLS characteristics ($\Rightarrow$ represented as fingerprint)
- Build a database mapping fingerprints with not directly related data, e.g.:

Fingerprint	Indicates
`771_1301_...`	TUM Apache webserver
`771_1302_...`	Nginx docker image
`770_cf_...`	TrickBot Command and Control (CnC) server

$\rightarrow$ Effective TLS fingerprinting groups similar deployments and distinguishes the rest

Background

Example TLS 1.3 handshake



Legend: **fingerprintable server data**, encrypted data

Motivation

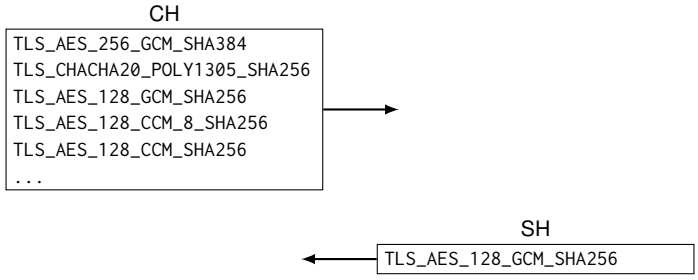
Future Applications:

1. **Enhance existing Intrusion Detection Systems**
   Servers from network flows are fingerprinted on-demand and results compared with known malicious ones
2. **Internet-wide measurements**
   Security researchers use fingerprinting to find previously unknown threats
3. **Monitoring of own Servers**
   Deviations from a fingerprints baseline can indicate an unintended software change or a malware infection

Goals

Problem: Early fingerprints with default CHs were not distinctive enough

This was due to the question-answer design of TLS, e.g.:



→ use unusual CHs that trigger distinguishable behaviors

→ find multiple CHs to increase the learned data

→ find a trade–off between learned data and scan costs (time and impact)

Introduction

Research Questions

1. How can we relate similar TLS server deployments?
2. How can we improve the effectiveness of our CHs?
3. What is the performance of actual fingerprinting use-cases?

Methodology

Research Questions

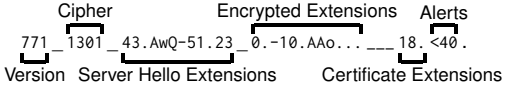
1. **How can we relate similar TLS server deployments?**
2. How can we improve the effectiveness of our CHs?
3. What is the performance of actual fingerprinting use-cases?

Methodology

Constructing Fingerprints

Extract data such that similar deployments have the same fingerprint

- A handshake is represented textually, e.g.,



- We include the content of 18 TLS extensions as feature (e.g., ALPN, Supported Groups, etc.)
- The final TLS fingerprint is a combination of multiple handshakes, e.g.,

```
771_1301..., 771_1302..., 770_fa. . . , ...
```

Effective Scanning Configurations

Research Questions

1. How can we relate similar TLS server deployments?
2. **How can we improve the effectiveness of our CHs?**
3. What is the performance of actual fingerprinting use-cases?

Effective Scanning Configurations

Challenge: Without knowledge about the implementation of every TLS server, it is impossible to select the ideal CHs for fingerprinting.

→ However, we can optimize the effectiveness of the CHs

We propose and empiric design of CHs:

1. Measure the effectiveness with a metric
2. Maximize the metric on a large pool of randomly generated candidates

Effective Scanning Configurations

Experimental Approach:

1. Randomly generate CHs (10 000) from the parameters space listed by IANA[3]
2. Alexa Top 1 Million measurement to pre-filter functional and good-performing CHs
3. Second measurement and scan every server with the top CHs (50)
4. Pick the (10) best combination of CHs maximizing the metric (distinct fingerprints)

This way we generated the **10 general-purpose CHs** used in the paper.

[3] IANA, „Transport Layer Security (TLS) Parameters," (2022). Adresse: https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml.

Fingerprinting Use Cases

Research Questions

1. How can we relate similar TLS server deployments?
2. How can we improve the effectiveness of our CHs?
3. **What is the performance of actual fingerprinting use-cases?**

Fingerprinting Use Cases

Overview

Long-running study over 30 weeks with weekly measurements:

Source	Total Scans	Unique Domains
Alexa Top 1 Million	81 M	12 M
The Majestic Million	37 M	2 M
abuse.ch Feodo Tracker	8 k	
abuse.ch SSL Blacklist	951	
Unique	104 M	12 M

Fingerprinting Use Cases

Content Delivery Network (CDN) Server Detection

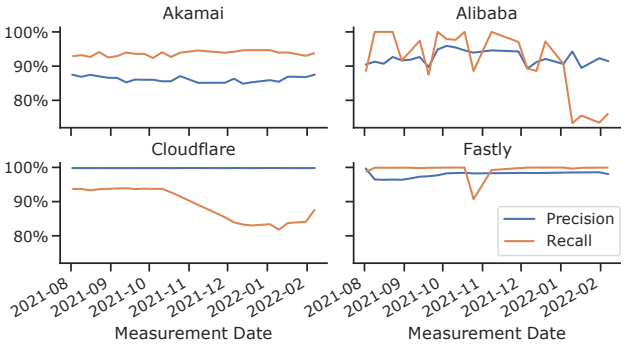
CDNs enable us to evaluate the approach because

- they are a single actor deploying TLS servers on a large-scale (large amount of data samples)
- their servers can be verified (AS, HTTP headers, and x509 certificates) to get a ground-truth

→ Evaluated a classification on week $n + 1$ based on fingerprints from weeks [1..n]

Metrics:

- Precision $\left(\frac{TP}{TP+FP}\right)$: „correct classifications“
- Recall $\left(\frac{TP}{TP+FN}\right)$: „detected CDN servers“



Note: The approach enabled us to detect off-net CDN servers in sometimes unexpected ASs

Fingerprinting Use Cases

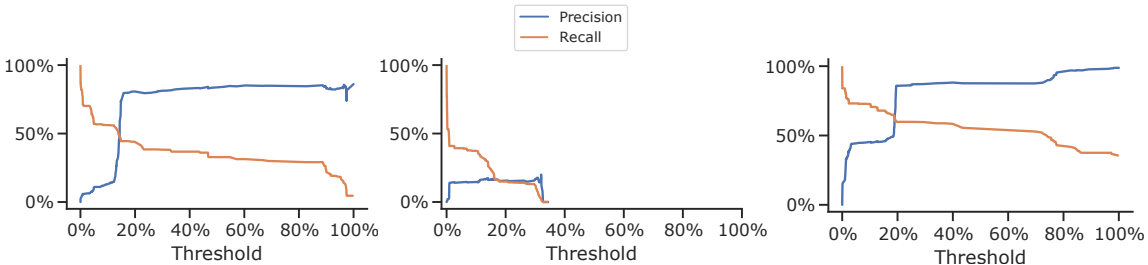
CnC Server Detection

Fingerprinting allows to detect potentially malicious servers

- We analyzed new targets on week $n + 1$ based on weeks $[1..n]$
- We considered how often a fingerprint is observed from CnC servers vs. from toplist servers
- This results in a score $\in [0, 1]$ how certain we are that we found a CnC server
- If the score was above a tune-able threshold, the server is classified as CnC server

Fingerprinting Use Cases

CnC Server Detection



→ Using the fingerprints together with additional data improves the detection

Conclusion

- Proposed a selection of TLS handshake features and their encoding as fingerprint to relate servers
- Provided a methodology for finding effective CHs and 10 general-purpose CHs that maximize information extraction from servers
- Demonstrated the potential of Active TLS Stack Fingerprinting based on detecting CDN and CnC servers
- The approach resulted in more effective fingerprinting compared to related work JARM[4]
- Open-sourced our data and code

https://active-tls-fingerprinting.github.io

4 J. Althouse, A. Smart, R. Nunnally Jr. u. a., Easily Identify Malicious Servers on the Internet with JARM, 17. Nov. 2020. Adresse: https://engineering.salesforce.com/easily-identify-malicious-servers-on-the-internet-with-jarm-e095edac525a.