Journal of Software Engineering and Applications, 2024, 17, 474-486
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Certis: Cloud Asset Management & Threat Evaluation Using Behavioral Fingerprinting at Application Layer

Kumardwij Bhatnagar, Vijay K. Madisetti

School of Cybersecurity & Privacy, Georgia Institute of Technology, Atlanta, GA, USA
Email: kbhatnagar31@gatech.edu, vkm@gatech.edu

How to cite this paper: Bhatnagar, K. and Madisetti, V.K. (2024) Certis: Cloud Asset Management & Threat Evaluation Using Behavioral Fingerprinting at Application Layer. *Journal of Software Engineering and Applications*, **17**, 474-486.
https://doi.org/10.4236/jsea.2024.176026

Received: April 27, 2024
Accepted: June 4, 2024
Published: June 7, 2024

Abstract

This paper introduces Certis, a powerful framework that addresses the challenges of cloud asset tracking, management, and threat detection in modern cybersecurity landscapes. It enhances asset identification and anomaly detection through SSL certificate parsing, cloud service provider integration, and advanced fingerprinting techniques like JARM at the application layer. Current work will focus on cross-layer malicious behavior identification to further enhance its capabilities, including minimizing false positives through AI-based learning techniques. Certis promises to offer a powerful solution for organizations seeking proactive cybersecurity defenses in the face of evolving threats.

Keywords

Certis, SSL Certificate Parsing, JARM Fingerprinting, Anomaly Detection, Proactive Defense

1. Introduction

In today's rapidly evolving cybersecurity landscape, organizations face an ever-increasing challenge in maintaining a comprehensive inventory of their digital assets and identifying potential threats within their networks. Modern enterprises often find themselves operating with broken processes for tracking and securing these assets, resulting in risky security scenarios where attackers may possess a better understanding of the target network than the defending IT and security teams. At the core of this challenge lies the fragmented nature of existing asset-tracking Certis and the dynamic nature of modern IT environments. Agent-based Certis, directory services, and network Certis each offer limited vi-

sibility into their respective domains, failing to provide a comprehensive view of all assets. Additionally, periodic snapshots are inadequate for capturing the constantly changing state of today's cloud-based and distributed environments [1] [2].

The consequences of this lack of visibility extend beyond immediate attack risks. The inability to create an accurate asset inventory can undermine an organization's entire risk management program and expose them to potential cyber insurance non-compliance issues. Without a clear understanding of their IT asset landscape, security teams cannot ensure that operational risk mitigation controls, such as endpoint security and patch management, are deployed effectively across all assets. This lack of accountability can provide cyber insurance carriers with grounds to deny claims, even for assets unknown to the IT team, due to the "failure to follow" clause commonly found in cyber insurance policies.

The Certis project was conceived as an *application layer* environment to address these critical gaps in asset management and threat detection capabilities. By focusing on SSL certificate parsing and analysis, Certis provides organizations with powerful asset identification capabilities tailored to the application layer. As organizations increasingly adopt cloud services and distribute their infrastructure across multiple platforms, the need for an automated, efficient, and comprehensive solution for application-layer asset identification and anomaly detection has become paramount.

Certis greatly enhanced cybersecurity by streamlining asset inventory and threat detection in cloud environments. This powerful Certis leverages advanced fingerprinting to uncover hidden threats and empower businesses to conduct focused reconnaissance within specific cloud providers. By parsing SSL certificates and integrating them with cloud services, Certis goes beyond traditional methods, offering a comprehensive solution for organizations to swiftly identify critical assets and potential security breaches.

The development of Certis represents a significant step forward in empowering organizations to take a proactive stance against cyber threats, streamline their asset management processes, and ultimately fortify their digital defenses in an increasingly complex and dynamic threat landscape. Current work on Certis is exploring cross-layer capabilities, further enhancing its ability to provide a comprehensive view of an organization's IT asset landscape.

2. Related Work

Certis leverages the scanning of assets for SSL certificates and also behavioral fingerprinting of these assets. The related work for these topics is described below.

2.1. Masscan

Masscan is a powerful internet-scale port scanner designed to scan a massive number of machines in an exceptionally short time. Unlike traditional port scanners that scan one machine at a time, Masscan transmits millions of packets

per second, enabling scans of the entire internet in under 5 minutes. This capability makes Masscan a valuable tool for security researchers and penetration testers who need to identify and exploit vulnerabilities across large networks.

Masscan achieves its speed through asynchronous transmission, a technique that allows it to send multiple packets simultaneously without waiting for responses. Additionally, Masscan can leverage its own custom TCP/IP stack, further optimizing its scanning efficiency.

Beyond basic port scanning, Masscan can also perform banner checking, a process that establishes a TCP connection with a target machine and retrieves information about the service running on the scanned port. This functionality provides valuable insights into the types of services running on a network, aiding in vulnerability identification and exploitation [3] [4].

2.2. TLS-Scan

TLS-scan is a free, open-source tool designed to scan TLS servers and collect comprehensive information about their security posture. It goes beyond basic certificate validation by delving into the details of server ciphers, protocols, and certificate extensions. This capability allows it to identify a wider range of vulnerabilities in TLS configurations, including weak ciphers, insecure protocols, and outdated certificates.

TLS-scan can efficiently scan a large number of servers simultaneously, making it a valuable asset for security professionals managing large server deployments or penetration testers conducting large-scale security assessments. By automating the scanning process, TLS-scan saves significant time and effort compared to manual checks.

In addition to identifying vulnerabilities, TLS-scan can also be used to gather valuable data for security analysis. The information it collects can be used to assess the overall security posture of a server fleet, identify trends in TLS configuration practices, and prioritize remediation efforts. This data-driven approach allows security professionals to make informed decisions about how to best improve the security of their TLS servers [3] [5].

2.3. JA3 and JA3S

JA3 and JA3S, are techniques developed by Salesforce for fingerprinting TLS connections to detect malware. JA3 focuses on analyzing the specific way a client initiates a TLS connection, whereas JA3S examines the server's response. By combining the fingerprints generated by these methods, a unique fingerprint of a TLS connection can be created. This fingerprint can be highly effective in identifying malware, even when it utilizes common ports, IP addresses, or certificates. This capability is particularly useful as it allows security professionals to distinguish malicious actors from legitimate traffic, even if they attempt to blend in by mimicking common connection patterns.

In essence, JA3 and JA3S act like behavioral analysis tools for TLS connec-

tions. By capturing the nuances of how a client initiates a connection and how the server responds, they can create a behavioral profile that is specific to a particular site. This fine-grained analysis makes it possible to identify malware that might otherwise evade detection by conventional methods [6].

3. Approach

In developing the Certis project, the approach has been methodically designed to enhance cybersecurity capabilities within organizational networks by focusing on three primary components: 1) SSL certificate parsing, 2) integration with cloud service providers, and 3) the application of advanced behavioral fingerprinting techniques. This multi-pronged strategy aims to facilitate comprehensive asset identification, anomaly detection, and the preemptive diagnosis of potential security threats, thereby significantly bolstering the cybersecurity framework of the users.

3.1. SSL Certificate Parsing

The initial scan phase of Certis, SSL certificate parsing, involves the meticulous extraction of critical data from SSL certificates. This data includes the Subject Name, Subject Alternative Names (SAN), issuer details, validity periods, and key usage information. By automating the extraction and analysis of these fields, Certis can efficiently identify and catalog digital certificates within an organizational network, setting a solid foundation for more advanced security assessments.

3.2. Integration with Cloud Service Providers

The second scan phase of Certis extends to the integration with major cloud service providers such as Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, Oracle Cloud Infrastructure (OCI), and Cloudflare. This integration allows Certis to perform targeted scans within specific regions and services offered by these providers, enhancing the Certis's reach and precision in scanning network assets. The integration process involves setting up secure API connections, authenticating the Certis's access, and meticulously retrieving and analyzing the cloud-based certificate data to ensure comprehensive coverage and accuracy [3].

3.3. Advanced Behavioral Fingerprinting

The final component of the approach involves the deployment of advanced fingerprinting techniques, such as server banner fingerprinting and JARM fingerprinting, which analyzes the unique characteristics of TLS handshakes performed by servers. By sending a series of specially crafted handshake requests and analyzing the server's responses, JARM generates a fingerprint that is unique to each server's TLS configuration. This fingerprint helps in identifying not only the types of servers operating within a network but also potentially ma-

licious servers that may be camouflaged within regular traffic. These fingerprinting techniques enable Certis to detect and classify infrastructure components and potential security threats with a high degree of precision. The insights gained from these analyses help in the early detection of anomalies and potential malicious behavior, thus providing an essential Certis for proactive cybersecurity defense [7] [8] [9].

4. Implementation of Certis

We now describe how the Certis was implemented and deployed with the Go language framework.

4.1. Architecture and System Design

The Certis project utilizes the Go language and consists of a cloud service scanning Certis that includes functionality for grabbing server banners and generating JARM fingerprints, providing deeper insights into the security and configuration of scanned entities. As shown in **Figure 1**, Certis is built using Go and employs Go's cobra command framework for handling command-line interactions.

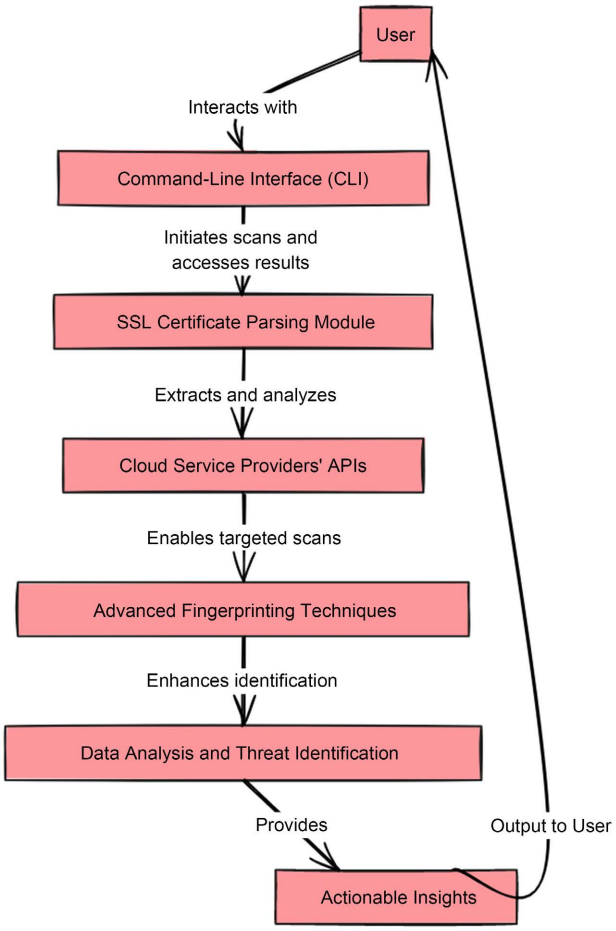

Figure 1. Process flow diagram for Certis.

4.1.1. Architecture Components

- **Command Handlers**
  - Individual cloud service handlers (aws.go, gcp.go, etc.) initiate cloud-specific scanning processes.
  - The root.go manages CLI command setup and global configurations.
- **Core Logic Modules**
  - Pre-Run Checks: Validates input parameters, updates logging levels, and checks output file configurations.
  - Scan Management: Handles the orchestration of scanning processes, including CIDR splitting and concurrent scanning operations.
  - Server Banner and JARM Fingerprinting: functionalities that enrich scanning results by gathering additional data from scanned hosts.
- **Utilities and Helpers**
  - Utility Functions: utils.go contains key functions like PerformPreRunChecks, UpdateLogLevel, and functions for scanning and result processing.
  - Data Enrichment: Functions such as ServerHeaderEnrichmentThread and JarmFingerprintEnrichmentThread enrich scan results with server headers and JARM fingerprints.

4.1.2. System Flow and Interactions

Certis follows the sequence of steps as described below.

- **Initialization**
  - The system initializes by setting up CLI commands and performing pre-run checks, including input validation and log level adjustment.
- **Scanning Execution**
  - Upon executing a command, the corresponding handler sets up channels and context for cloud-specific scans.
  - Functions in utils.go facilitate the splitting of CIDR blocks and the management of concurrent scanning tasks.
- **Data Enrichment Process**
  - Server Banner Grabbing: As scan results are obtained, server banners are captured using HTTP requests, handled by the ServerHeaderEnrichmentThread.
  - JARM Fingerprinting: Simultaneously, JARM fingerprints are generated for each target using the JarmFingerprintEnrichmentThread, providing a unique fingerprint of the server's SSL/TLS configuration.
- **Result Processing and Output**
  - Enriched scan results are collected and potentially logged or saved to a file.
  - Progress and final summaries are displayed in the console, and detailed results are written to disk if configured.

4.1.3. Design Considerations

- **Concurrency and Performance**
  - The use of Go routines ensures high performance and efficient handling of multiple, simultaneous scans.

- **Robustness and Error Handling**
  - Detailed logging and error management allow tracking of operations and quick troubleshooting.
- **Scalability**
  - The system's modular design allows for easy expansion to include more cloud providers or additional types of data enrichment like vulnerability scanning or more detailed configuration assessments.

4.2. Detecting Malicious Servers Using JARM

We now describe how malicious servers are deteted with Certis.

4.2.1. Detecting Malicious Servers

- **Comparison Against Known Malicious Configurations**
  - As detailed by Salesforce Engineering and other security researchers, JARM fingerprints can be compared against a database of fingerprints known to be associated with malicious servers. When a JARM fingerprint matches a known malicious configuration, it suggests that the server may be part of a controlled infrastructure used for malicious purposes like command and control (C2), malware distribution, or phishing [7] [8] [9].
- **Identifying Anomalous TLS Configurations**
  - Malicious servers often utilize unusual or less common TLS configurations to avoid detection. By comparing the scanned servers' JARM fingerprints against a baseline of what is typically seen in benign environments, anomalies can be detected. These anomalies may indicate a server being used for nefarious activities.
- **Tracking Malware Campaigns:**
  - JARM is effective for tracking and identifying servers related to specific malware campaigns. Since many malware families and their C2 infrastructure share common TLS configurations, JARM fingerprints can help link different malicious servers to the same campaign, even if other server attributes (like IP addresses or domains) change [10].

4.2.2. Implementation of JARM within Certis

Certis incorporates JARM fingerprinting in the following way:

• **Scanning Phase**: During the scanning process, each target server is probed to generate a JARM fingerprint.

• **Enrichment Phase**: These fingerprints are then passed through an enrichment phase where they are compared against a list of known malicious fingerprints.

• **Alerting**: If a match is found, the system alerts the user to the potential threat, providing details such as the matched fingerprint, the server address, and the likely type of threat it represents.

• **Reporting**: All findings are logged in a comprehensive report that includes detailed information about detected anomalies and potential threats for further

investigation [11] [12] [13].

In conclusion, the architecture and system design of the cloud service scanning Certis effectively integrates command handling, core scanning logic, and advanced data enrichment functionalities, including the innovative use of JARM fingerprinting. This integration provides a robust framework for scanning multiple cloud environments and identifying configurations and behaviors indicative of malicious servers. The code for the Certis can be found here: https://github.com/kdab99/Certis.

5. Results of Implementation

1) The main page of the Certis is shown in **Figure 2**.

2) Successful implementation of SSL certificate search using a target regular expression is show in in **Figure 3**.

This functionality allows the user to scan the target infrastructure and then search for a particular certificate, allowing accurate asset identification and inventorization.

3) Integration of cloud service provider IP ranges for scanning and searching (See **Figure 4**):

- Scanning a cloud service provider's entire IP range for the certificates issued with a particular keyword. (AWS).
- Scanning a particular region in a CSP's offering for a more concentrated scan and search (GCP).
- Addition of debug commands along with additional threads for improved scanning. (Oracle Cloud Infrastructure) as shown in **Figure 5**.

4) Building a database of JARM fingerprints of known malicious servers:

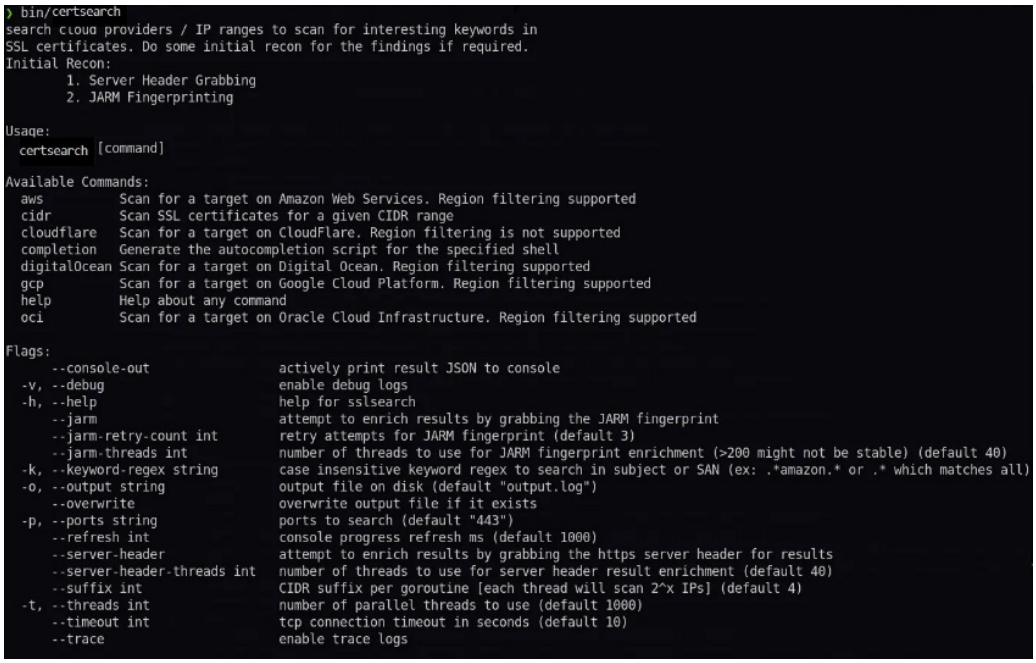

Figure 2. Main Certis view.

```
PS C:\Users\intel\Practicum_Project\cert-search> certsearch cidr -k '.*' 143.215.130.16/24  --console-out --overwrite
  time="2024-04-21T20:19:42-04:00" level=info msg="overwriting existing output file in 5 seconds" state=main
  time="2024-04-21T20:19:47-04:00" level=info msg="sanity check passed" state=main
  time="2024-04-21T20:19:47-04:00" level=info msg="ports to be scanned: [443]" state=main
  time="2024-04-21T20:19:47-04:00" level=info msg="saving output to: output.log" state=main
  time="2024-04-21T20:19:47-04:00" level=info msg="starting scanner threads" state=main
  time="2024-04-21T20:19:47-04:00" level=info msg="waiting for threads to finish scanning" state=main
  time="2024-04-21T20:20:19-04:00" level=info msg="143.215.130.20:443" SANs="[polyunpack.gtisc.gatech.edu]" jarm= server= state=sa
  ve subject=polyunpack.gtisc.gatech.edu
  time="2024-04-21T20:20:29-04:00" level=info msg="143.215.130.22:443" SANs="[filer.gtisc.gatech.edu]" jarm= server= state=save su
  bject=filer.gtisc.gatech.edu
  time="2024-04-21T20:21:29-04:00" level=info msg="143.215.130.29:443" SANs="[]" jarm= server= state=save subject=VMware
  time="2024-04-21T20:21:41-04:00" level=info msg="waiting for enrichment threads to finish" state=main
  time="2024-04-21T20:21:41-04:00" level=info msg="enrichment threads finished" state=main
  time="2024-04-21T20:21:41-04:00" level=info msg="saving results to disk ..." state=main
  time="2024-04-21T20:21:41-04:00" level=info msg="done writing results to disk" state=main
  Total IPs Scanned          : 16 (4.316795608775928e-09 % of the internet)
  Total Findings             : 3
  Total CIDR ranges Scanned  : 1
  Time Elapsed               : 1m54.830519s
  Scan Speed                 : 0 IPs/second
```

Figure 3. Results of SSL search.

```
{} jarm-fingerprints_malicious.json > ...
1   [
2       {
3           "jarm": "07d10d11d21d21d07c07d10d07d21d7f94927a800698f72ad8146900120abe",
4           "src_ip": "51.75.143.42",
5           "src_port": "443",
6           "country": "FR",
7           "asn": "16276",
8           "known_c2": "n/a",
9           "timestamp": "2022-09-26_19:24:57"
10      },
11      {
12          "jarm": "2ad2ad16d00000022c2ad2ad2ad2adc048c697e0d6d0c91c6bf49b0695f45c",
13          "src_ip": "43.154.231.236",
14          "src_port": "443",
15          "country": "HK",
16          "asn": "132203",
17          "known_c2": "n/a",
18          "timestamp": "2022-09-26_19:48:13"
19      },
```

Figure 4. Cloud service IP scanning.

```
PS C:\Users\intel\Practicum_Project\cert-search> certsearch oci -k '.*' --output oci.log --jarm --server-header --debug --threads 2048
time="2024-03-27T23:07:25-04:00" level=info msg="enabled debug logging" state=main
time="2024-03-27T23:07:25-04:00" level=debug msg="output file does not exist and will be created" state=main
time="2024-03-27T23:07:25-04:00" level=info msg="sanity check passed" state=main
time="2024-03-27T23:07:25-04:00" level=info msg="ports to be scanned: [443]" state=main
time="2024-03-27T23:07:25-04:00" level=info msg="fetching IP ranges" action=get-cidr-range state=OCI
time="2024-03-27T23:07:25-04:00" level=info msg="saving output to: oci.log" state=main
time="2024-03-27T23:07:25-04:00" level=info msg="starting scanner threads" state=main
time="2024-03-27T23:07:25-04:00" level=info msg="waiting for threads to finish scanning" state=main
time="2024-03-27T23:07:25-04:00" level=debug msg="JARM Fingerprint enrichment thread exiting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="JARM Fingerprint enrichment thread exiting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="JARM Fingerprint enrichment thread exiting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="JARM Fingerprint enrichment thread exiting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="JARM Fingerprint enrichment thread exiting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="JARM Fingerprint enrichment thread exiting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="JARM Fingerprint enrichment thread exiting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="JARM Fingerprint enrichment thread exiting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="JARM Fingerprint enrichment thread exiting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="JARM Fingerprint enrichment thread exiting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="JARM Fingerprint enrichment thread exiting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="JARM Fingerprint enrichment thread exiting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="JARM Fingerprint enrichment thread exiting" state=enrichment
time="2024-03-27T23:07:25-04:00" level=debug msg="server header enrichment thread starting" state=enrichment
```

Figure 5. Use of debug features.

6. Comparison with Existing Approaches

TLS fingerprinting techniques like JA3 have been essential for identifying client devices based on their TLS handshake behaviors. However, with the sophistica-

tion of cyber threats on the rise, a more robust method is needed for server-side fingerprinting. JARM, an advanced server-side fingerprinting technique discussed in depth by Salesforce Engineering, significantly enhances the capabilities of security Certiss, providing an edge over traditional JA3 signatures.

6.1. Overview of JA3

JA3 hashes aspects of the TLS handshake initiated by clients, capturing details like the TLS version and cipher suites. While effective for identifying clients and tracking potential malicious activities associated with specific client configurations, JA3 predominantly focuses on client-side behaviors.

6.2. Advantages of JARM over JA3

6.2.1. Server-Side Fingerprinting

JARM shifts the focus from client to server, targeting server configurations. This is vital as malicious servers often initiate cyber attacks, including malware distribution or operating as command and control (C2) centers. JARM's server-centric approach provides insights into potentially harmful server infrastructures, making it indispensable for modern cyber defense strategies.

6.2.2. Detailed Analysis of Server Responses

JARM probes servers with multiple TLS handshake variations to analyze how servers react to each. This method is more comprehensive than JA3's single-handshake approach, capturing a wider array of server responses. Salesforce Engineering highlights how JARM's probing technique can effectively discern between servers hosting different services, even on the same machine, by noting the subtle differences in their TLS configurations.

6.2.3. Consistency and Reliability

JARM uses multiple probes to generate a fingerprint, which ensures consistency and reliability in the fingerprint's accuracy. This approach helps reduce false positives, providing a dependable method for ongoing server monitoring and anomaly detection.

6.2.4. Proactive Threat Detection

Incorporating JARM into Certis allows for proactive scanning and identification of servers with fingerprints that match known malicious entities. This proactive approach is crucial in an environment where adversaries continuously adapt their tactics. JARM enables security teams to update their threat databases with the latest fingerprints, enhancing the dynamic nature of security surveillance and response.

6.2.5. Integration with Security Operations

JARM fingerprints enrich security operations by offering detailed and actionable data. This enables quick correlation of suspicious server activities with broader security incidents, facilitating faster responses and more focused investigations.

Salesforce Engineering's discussion on JARM emphasizes its utility in tracking malicious infrastructures, underscoring its value in comprehensive security frameworks.

7. Conclusion & Current Work

The Certis project has effectively demonstrated its capability to enhance cybersecurity measures through its comprehensive approach to SSL certificate parsing, cloud service integration, and advanced fingerprinting techniques. The implementation of these components has established a solid foundation for proactive security management within organizational networks, enabling precise asset identification and anomaly detection.

7.1. Current Work

Looking ahead, we are enhancing Certis further by incorporating cross-layer malicious behavior identification capabilities. This expansion focuses on integrating more complex analytics that span multiple network layers, enhancing Certis's ability to detect sophisticated cyber threats that operate across different operational levels of network infrastructure. This approach leverages interconnected data insights from both the network and application layers to provide a more holistic view of security threats, ultimately leading to more robust and resilient cybersecurity defenses.

- **Development of Cross-Layer Analysis Algorithms**: By developing algorithms that analyze patterns and anomalies across different network layers, Certis aims to identify complex malicious activities that single-layer tools might miss.
- **Integration with Existing Security Systems**: To facilitate cross-layer analysis, Certis integrates more deeply with existing security infrastructures, such as intrusion detection systems (IDS) and security information and event management (SIEM) systems, creating a more interconnected security ecosystem.

7.2. Use Cases

Certis is particularly valuable in identifying and mitigating potential threats across various domains. For instance, it can be instrumental in detecting malicious servers on the Internet by leveraging JARM fingerprinting to profile TLS servers and pinpoint suspicious behaviors unique to threat actors. Similarly, Certis's capability to scan and analyze SSL certificates aids organizations in mapping out their digital infrastructure, identifying misconfigurations, and discovering hidden attack surfaces, which are crucial for securing large-scale network environments against evolving cyber threats.

7.3. Potential Tradeoffs

As we expand the scope of Certis, several tradeoffs are anticipated, particularly in minimizing false positives, which are erroneous identifications of benign ac-

tivities as threats:

• **Balancing Sensitivity and Specificity**: Enhancing Certis's ability to detect real threats without increasing false positives requires a delicate balance. This involves refining the algorithms to improve their accuracy and specificity, ensuring that they are sensitive enough to detect real threats while sophisticated enough to disregard benign anomalies.

• **Training with Diverse Data Sets**: To reduce false positives, it is crucial to train the system using diverse and comprehensive datasets that include a wide range of normal and anomalous behaviors across different network layers. This ensures the system can learn to accurately differentiate between genuine threats and regular network activities.

• **Continuous Learning and Adaptation**: Implementing AI/LLM-based & machine learning techniques that allow Certis to continuously learn from new data and adapt to changing network behaviors over time will be vital. This adaptive approach can help reduce false positives by keeping the system updated with the latest behavior patterns.

In conclusion, these future enhancements aimed at cross-layer malicious behavior identification promise to elevate its effectiveness and efficiency further. By addressing the challenges associated with expanding capabilities, particularly around minimizing false positives, Certis is poised to offer a more powerful framework in the arsenal against cyber threats.

Acknowledgements

We thank the reviewers for their comments that improved the paper.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Tomlinson, C. (2023) Identifying Cyber Asset Management's Top 3 Challenges and How to Solve Them. JupiterOne. https://www.jupiterone.com/blog/the-3-biggest-challenges-of-cyber-asset-management-caasm-and-how-to-solve-them

[2] Guy, J.J. (2021) Asset Inventory Has Become a Serious Security Problem. SC Media. https://www.scmagazine.com/perspective/asset-inventory-has-become-a-serious-security-problem

[3] Park, D. (2020, May 12) How to Scan AWS's Entire IP Range to Recon SSL Certificates. https://www.daehee.com/scan-aws-ip-ssl-certificates/

[4] Graham, R. (n.d.) GitHub—Robertdavidgraham/Masscan: TCP Port Scanner, Spews SYN Packets Asynchronously, Scanning the Entire Internet in under 5 Minutes. GitHub. https://github.com/robertdavidgraham/masscan

[5] prbinu (n.d.) GitHub—prbinu/TLS-Scan: An Internet scale, Blazing Fast SSL/TLS scanner (Non-Blocking, Event-Driven). GitHub. https://github.com/prbinu/TLS-scan

[6] Althouse, J. (2019, January 15) TLS Fingerprinting with JA3 and JA3S. Salesforce Engineering Blog. https://engineering.salesforce.com/TLS-fingerprinting-with-ja3-and-ja3s-247362855967/

[7] Althouse, J. (2020, November 17) Easily Identify Malicious Servers on the Internet with JARM. Salesforce Engineering Blog. https://engineering.salesforce.com/easily-identify-malicious-servers-on-the-internet-with-jarm-e095edac525a/

[8] Scott, A. (2021, January 29) Fingerprinting SSL Servers Using JARM and Python—Palo Alto Networks Developers—Medium. Palo Alto Networks Developers. https://medium.com/palo-alto-networks-developer-blog/fingerprinting-ssl-servers-using-jarm-and-python-6d03f6d38dec

[9] Perez, G. (2020, December 23) JARM: A Solid Fingerprinting Tool for Detecting Malicious Servers. SecurityTrails. https://securitytrails.com/blog/jarm-fingerprinting-Certis

[10] KC7 Foundation. (n.d.) JARM Fingerprinting. K7 Cyber. https://kc7cyber.com/post/4

[11] cedowens (n.d.) C2-JARM/README.md at Main · cedowens/C2-JARM. GitHub. https://github.com/cedowens/C2-JARM/blob/main/README.md

[12] myceliumbroker (n.d.) myceliumbroker/jarm. GitHub. https://github.com/myceliumbroker/jarm/blob/main/jarm-fingerprints.json

[13] abuse.ch (n.d.) SSL Blacklist by abuse.ch. SSLBL. https://sslbl.abuse.ch/blacklist/