SD-WAN TLS Scanning

Sergey Gordeychik
Inception Institute of Artificial Intelligence
Abu-Dhabi, UAE
serg.gordey@gmail.com

Denis Kolegov
Tomsk State University
Tomsk, Russia
d.n.kolegov@gmail.com

Antony Nikolaev
Tomsk State University
Tomsk, Russia
antoniy.nikolaev@gmail.com

Introduction

In this work, we present a scan for known vulnerabilities in the TLS implementations of SD-WAN products deployed on the Internet. We used the TLS-Attacker framework as a TLS scanning engine and the Grinder framework, developed by us, as an orchestrator.

The idea to conduct this research was born after reading the "*Scalable Scanning and Automatic Classification of TLS Padding Oracle Vulnerabilities*" paper[1]. The last one evaluated the Alexa Top Million Websites for CBC padding oracle attack and revealed vulnerabilities in 1.83% of them. The vulnerability was found in the following network security products:

- Citrix Application Delivery Controller (ADC) and NetScaler Gateway. CVE-2019-6485.
- F5 BIG-IP. CVE-2019-6593. TMM TLS virtual server vulnerability CVE-2019-6593.
- SonicWall SonicOs. CVE-2019-7477.
- Oracle HTTP Server. Oracle Critical Patch Update Advisory - July 2019

We observed that the paper did not mention vulnerabilities to CBC Padding Oracle Attack in SD-WAN products. It can be explained by many reasons. For example, SD-WAN TLS interfaces are may not be vulnerable to this attack or the SD-WAN nodes are not in Alexa Top Million Websites. At the same time, the paper considers and encounters only vulnerabilities to CBC padding attack. We thought that it was also interesting to scan SD-WAN nodes related to all main TLS vulnerabilities.

Within our research, we considered only SD-WAN nodes (controllers, Web UI, edge routers, etc.) we had already collected and stored via enumeration and fingerprinting[2].

[1] https://github.com/RUB-NDS/TLS-Padding-Oracles
[2] https://github.com/sdnewhop/sdwannewhope/blob/master/docs/census.md

Approach and Methodology

The employed methodology can be defined as follows:

1. Craft signatures of the interfaces of an SD-WAN product.
2. Define and express the signatures within a search engine query language.
3. Discover and enumerate devices using the search engines (*Shodan*, *Censys*, etc.).
4. Use incremental save method to store the results in the database.
5. Run TLS scanning engines (e.g., *TLS-Attacker*, *SSL Labs Server Scan*, etc.) on the appropriate hosts and interfaces from the database.
6. If vulnerabilities are found, rescan the node two times to minimize false positives.
7. If the vulnerabilities are still present, check them again using PoC scripts in Python.
8. Save the confirmed results to the database.
9. If a new SD-WAN product or interface is discovered go to step 1.
10. Run the steps 3 - 7 regularly.

Responsible Disclosure

We responsibly reported findings to several vulnerable vendors. At this time, the notified vendors have been processing and fixing the vulnerabilities. If a vulnerability was found in an older version of a product but was absent in the new version of the considered product, we did not notify the vendor. We also reported several bugs found in *TLS-Attacker*.

Automation and Orchestration

Automation and orchestration are implemented by the *Grinder* framework[3]. *Grinder* is a security research intelligence framework that have been developing since 2018. It was created to automatically enumerate, fingerprint and scan hosts on the Internet using different specialised back-end systems: search engines (e.g., *Shodan* or *Censys*) for discovering, scanners (e.g., NMAP) for active fingerprinting and scanning, vulnerability databases (e.g., Vulners) for getting information related to vulnerabilities in the discovered software. The *Grinder* framework can be used in many different areas of security research, as a connected Python module to your project or as an independent ready to use from the box tool.

The main purpose of *Grinder* is to unify different sophisticated security tools, aggregate gathered information and help understanding collected information and exposed statistics related to the hosts in the research scope. *Grinder* incrementally saves all scans, results and statistics to its database to compare results over time and track the statistics changes.

[3] https://github.com/sdnewhop/grinder

To visualize gathered data, *Grinder* provides an interactive world map with all results. *Grinder* map backend is written in *Flask* and supports additional REST API methods to get more information about all scanned hosts or some particular host from the map. Also, it is possible to show some additional information about host interactively from the map. For example, currently open host will be automatically checked for availability with “ping” from the backend, also for every host many additional features are available: current host can be directly opened in *Shodan*, *Censys* and *ZoomEye* search engine web interfaces, host can be shown on Google Maps with all available information about geolocation, also it is possible to make an IP lookup, or open raw information in JSON directly in browser or from your application with provided API methods.

We used the *TLS-Attacker*[4] framework as a TLS core scanning engine and the *Grinder* framework as an orchestrator. In this case, the *TLS-Attacker* was used as a back-end within the Grinder framework. To get information about TLS configuration, bugs, and possible attacks, Grinder provides a wrapper module for *TLS-Scanner* and *TLS-Attacker* tools, that is used to handle all scanning options and analyze the results. TLS-Attacker is a Java-based framework for analyzing TLS libraries. It is able to send arbitrary protocol messages in an arbitrary order to the TLS peer, and define their modifications using a provided interface. Grinder uses related *TLS-Scanner* module to scan TLS configuration with *TLS-Attacker*.

Gathered data by *TLS-Scanner* and *TLS-Attacker* helps *Grinder* to count unique attacks and bugs, also it is possible to count different unique types of entities for every vendor, product, protocol, and many other types and categories. To get more accurate results, *Grinder* firstly checks availability of every host that was found in *Shodan* or *Censys*, after that *Grinder* tries to detect proper port and other options to successfully start scan with *TLS-Scanner*, and finally Grinder saves every result from every host in separate files that later can be parsed more accurately with additional parser and analyzer.

Results

We scanned about 7500 nodes and found the following:

- 1873 nodes were vulnerable to Sweet 32 attack
- 121 nodes were vulnerable to CBC Padding Oracle attack
- 30 nodes were vulnerable to CRIME attack
- 29 nodes were vulnerable to Logjam attack
- 28 nodes were vulnerable to CVE-2016-2107
- 14 nodes were vulnerable to DROWN attack
- 6 nodes were vulnerable to ROBOT attack
- 1 node was vulnerable to Heartbleed

Vulnerabilities to the ROBOT attack were found in Viprinet product only.

[4] https://github.com/RUB-NDS/TLS-Attacker

We found one Riverbed SteelHead node vulnerable to Heartbleed attack. The software was released in 2013.
Riverbed SteelHead nodes released prior 2014 and Viprinet VPN Virtual Hub nodes were vulnerable to CBC Padding Oracle attack.