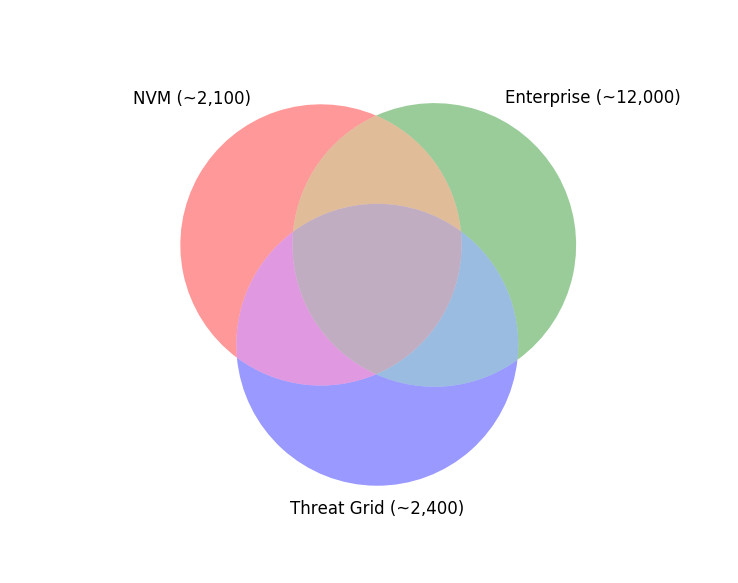
To protect your data, you must understand the traffic on your network.  This task has become even more challenging with widespread use of the Transport Layer Security (TLS) protocol, which inhibits traditional network security monitoring techniques.  The good news is that TLS fingerprinting can help you understand your traffic without interfering with any of the security benefits TLS brings to applications and complements current solutions like Encrypted Traffic Analytics [9].   To help our customers better understand the benefits of the approach, and to help drive the development and adoption of defensive uses of traffic analysis, the Advanced Security Research team of Cisco’s Security and Trust Organization has published a large set of fingerprints [1] with the support of the Cisco Technology Fund.

Transport Layer Security (TLS) fingerprinting is a technique that associates an application and/or TLS library with parameters extracted from a TLS ClientHello by using a database of curated fingerprints, and it can be used to identify malware and vulnerable applications and for general network visibility. These techniques gained attention in 2009 with mod\_sslhaf [2], in 2012 with SSL fingerprinting for p0f [3], in 2015 with FingerprinTLS [4], and most recently with JA3 [5].  We have been using this approach at Cisco since 2016 [6].  The attention given to TLS fingerprinting has been warranted because it is a proven method that provides network administrators with valuable intelligence to protect their networks. And while more of the TLS handshake goes dark with TLS 1.3, client fingerprinting still provides a reliable way to identify the TLS client. In fact, TLS 1.3 has *increased* the parameter space of TLS fingerprinting due to the added data features in the ClientHello. While there are currently only five cipher suites defined for TLS 1.3, most TLS clients released in the foreseeable future will be backwards compatible with TLS 1.2 and will therefore offer many “legacy” cipher suites. In addition to the five TLS 1.3-specific cipher suites, there are several new extensions, such as supported versions, that allows us to differentiate between clients that supported earlier draft implementations of TLS 1.3.

Why is our approach different?

But here’s the catch: the visibility gained by TLS fingerprinting is only as good as the underlying fingerprint database, and until now, generating this database was a manual process that was slow to update and was not reflective of real-world TLS usage. Building on work we first publicly released in January 2016 [6], we solved this problem by creating a continuous process that fuses network telemetry with endpoint telemetry to build fingerprint databases automatically. This allows us to leverage data from managed endpoints to generate TLS fingerprints that give us visibility into the (much larger) set of unmanaged endpoints and do so in a way that can quickly incorporate information about newly released applications. By automatically fusing process and OS data gathered by Cisco® AnyConnect® Network Visibility Module (NVM) [7] with network data gathered by Joy [1], our system generates fingerprint databases that are representative of how a diverse set of real-world applications and operating systems use network protocols such as TLS. We also apply this process to data generated from Cisco Threat Grid [8], an automated malware analysis sandbox, to ensure that our system captures the most recent trends in malware. With ground truth from multiple sources like real-world networks and malware sandboxes, we can more easily differentiate fingerprints that are uniquely associated with malware versus fingerprints that need additional context for a confident conviction.

Our internal database has process and operating system attribution information for more than 4,000 TLS fingerprints (and counting) obtained from real-world networks (NVM) and a malware analysis sandbox (Threat Grid). The database also has observational information such as counts, destinations, and dates observed for more than 12,000 TLS fingerprints from a set of enterprise networks. We have open sourced a subset of this database that, at more than 1,900 fingerprints (and counting), is the largest and most informative fingerprint database released to the open-source community.   This database contains information about benign processes only; we are not able to publish fingerprints for malware at this time.



Given the records generated from the data fusion process, we report **all** processes and operating systems that use a TLS fingerprint, providing a count of the number of times we observed each process or operating system using the TLS fingerprint in real-world network traffic. This schema gives a more realistic picture of TLS fingerprint usage (in other words, many processes can map to a single fingerprint with some more likely than others).

Another advantage of our database is that it provides as much relevant contextual data per fingerprint as possible. The primary key into our database is a string that describes the TLS parameters that you would observe on the wire, which allows a system generating these keys to provide valuable information even in the case of no database matches. We associate each TLS parameter in the ClientHello with the RFC that first defined that parameter and use this information to report maximum and minimum implementation dates. These dates provide useful context on the age of the cryptographic parameters in the ClientHello and are not dependent on a database match.

Conclusions

We are continuing to develop applications on top of the fingerprint database we are developing, and any input is most welcome. To facilitate collaboration, we have open sourced a subset of the internal TLS fingerprint database along with several python tools as a part of the Joy project [1]. Our open-source contributions include:

* A fingerprint database that is updated weekly
* A Joy feature to generate fingerprint strings for TLS flows
* A set of python programs to identify TLS fingerprints in a packet capture or from a live network interface and generate new TLS fingerprints that can be contributed to the open-source community
* A web-based user interface to visualize the results of TLS fingerprinting

[1] <https://github.com/cisco/joy>

[2] <https://blog.ivanristic.com/2009/06/http-client-fingerprinting-using-ssl-handshake-analysis.html>

[3] <https://idea.popcount.org/2012-06-17-ssl-fingerprinting-for-p0f/>

[4] <https://github.com/synackpse/tls-fingerprinting>

[5] <https://github.com/salesforce/ja3>

[6] <https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=449962>

[7] <https://www.cisco.com/go/anyconnect>

[8] <https://www.cisco.com/go/threatgrid>

[9] <https://www.cisco.com/go/eta>