3T

3 TERRITORY SOLUTIONS

FLOW ANALYTICS PLATFORM FOR OPERATIONAL TECHNOLOGY (OT)

Insight into the workings of an enterprise is essential to effectively and efficiently manage and defend its assets.

8608 Dakota Drive, Gaithersburg, MD 20877 / interest@3tsoln.com / 301.926.1080 / 3tsoln.com

TABLE OF CONTENTS

PROBLEM STATEMENT

Insight into the workings of an enterprise is essential to effectively and efficiently manage and defend its assets. However, in the **operational technology (OT)** space, which includes **medical devices and equipment (MDE)** and **facility-related control systems (FRCS)**, stakeholders have a restricted understanding of what is actually happening within their organizations. This is painfully obvious when examining average dwell time (nearly one year), how quickly new devices are discovered (some "new" devices have been on the network for years), and the difficulty engineers / managers have with quantitively stating their positions and needs.

SUMMARY OF PROPOSED APPROACH

The following approach is intended to create a flow analytics platform for OT using security techniques whose scalability is compute-limited, functionally automatic, and agnostic to the network environment.

The approach is not intended to establish an illicit network discovery process. The proposed techniques and tools will passively collect information and will, at no time, inject packets, nor will they dive into or rely upon packet contents. All techniques will be rooted in command-line inspection using common, open, and simple tools to enable scripting. The approach will not use tools or techniques that cannot be scripted or executed in batch. Further, to maximize applicability, the tools will be restricted to default / standard libraries and configurations. The approach is intended to be modular, meaning that new scripts can be developed and added to the overall batch as they are built out to allow for infinite scalability (as compute allows, of course).

To minimize initial implementation costs, the tools and techniques should initially be used on traffic that will has a digestible Ethernet header. It is likely that existing sensors on the enclave will be or have been configurated to capture this protocol. Communication over twisted-pair and other less common bus protocols should be completed to round out the remaining 25% to 40% of traffic only after a stable baseline with Ethernet headers has been established.

FUNDAMENTAL GOALS

Dependency Identification: The first key function of flow analytics is to gain an understanding of what influences what, and how the timing of it all works together to create a process. As illustrated in the examples below, the approach allows for queries to determine the who's and what's. Meanwhile, time-stamping connection data allows for a sort of session recreation involving multiple machines, making it possible to identify soft spots, bottlenecks, and single points of failure.

Security Baseline: The second key function of flow analytics is to gain an understanding of what behavior is "normal" within a medical subnet. This is especially important with MDE and FRCS, as these systems are generally quite deterministic. The query results will be able to identify a baseline of active ports, protocols, and services; confirm hardware and possibly software inventories; and inform intrusion detection criteria.

Going Beyond "Just" Data Flow: Once a solid understanding of the data flows is achieved, the approach is to be augmented with real-world activities performed by the clinical staff to understand how these flows impact the organization's mission. While it may be tempting to immediately use the new information to inform and implement improvements to clinical efficiency, it is necessary to first establish a measurable baseline so that gains can be quantified and studied. Initially, the focus should be on how the flow of data might affect the delivery of care, patient safety, or the operational fitness of the facility.

Information Sharing & Integration: Far too often, solutions are developed in a vacuum, and the closest often seen to "sharing" is an after-the-fact API built using a "set it and forget it" methodology. This is not an effective way to scale, and it unnecessarily hinders defenders' capabilities, often forcing them to recreate a technique, script, or data set that someone else may have already executed. For this reason, all parts of the proposed approach must be technology-agnostic to allow integration with existing tools. Sharing results, scripts, and techniques is encouraged, within the allownce of HIPAA or other regulations. At a minimum, the outputs of queries should be compatible with MOSAICS and the HBSS suite.

Identification of User Behavior: Properly crafted queries will provide insight into how tools are being used / not used, and by which users. This will allow organizations to actively reduce their attack surface by removing unnecessary vectors.

Automation: The approach will minimize the need for human interaction / inspection during the development and implementation of new techniques and scripts. It will also reduce the maintenance tail by minimizing "special snowflakes" in favor of standardization and automation. This is crucial for scalability and freeing up resources to continue moving the overall platform forward.

ASSUMPTIONS

- The techniques and tools must be batch run at the command-line level and capable of real-time inspection at high speed and large volume, against live logs.
- As OT protocols vary widely, it must assume that packet data contents are obfuscated, crypt, trixie, or of an unknown "proprietary" structure.
- There is no use of Splunk, Wireshark, tshark, SIMs, or any other "fancy" tools. No session reconstruction. Snort usage will be limited as part of the front-line batch, but it could be considered after real-time parsing activities to check against Virus Total and/or Snort alerts.

SUMMARY OF APPROACH

The creation and subsequent maintenance of flow analytics will need to be done in phases, with each phase building on the previous ones. All steps are to be executed using existing enclave services.

ENCLAVE DATA EXTRACTION

1. As there is likely a large volume of collected data from existing sensors, a logical starting place would be to obtain a random selection of packet captures that loosely represent the entire enterprise at scale.

2. A manual inquiry into the existing instrumentation should be requested and documented.

3. A request to connect to a sample of live sensors should also be made early on, as it will quickly become part of the process critical path.

4. Ultimately, the capture of packets by use of in-line sensors or by mirroring ports at switches is desired.

5. Note: The packets being captured will likely contain ePHI and other regulated information. As such, it is very important that the early iterations remove any non-header information to reduce the administrative burden for these types of regulated data. Creating this process, in an auditable fashion, will be a challenge. However, it can, and should, be executed at the capture point.

DATA PARSING / SCRIPT DEVELOPMENT

1. Assuming clear blue sky instrumentation, develop an initial command line script(s), as outlined below, that accomplishes the desired query.

2. Run the script(s) against static pcaps. If written properly, most scripts should work okay, as they should use only the highest-level header information. Any errors will inform of gaps in instrumentation configuration. Use these gaps to request changes to capture configurations and/or sensor locations.

3. Where the instrumentation cannot be modified, rework / hack the scripts to obtain the desired end result. It is critical to keep this minimized, as "special snowflake" configurations and approaches will reduce automatability, increase the maintenance tail, and greatly reduce resiliency.

4. Repeat steps 1 through 3 for all proposed queries until the stress testing against a static environment is satisfactory for the baseline query set.

5. Once the baseline is built and working against a static environment, gradually introduce one query at a time to the dynamic/real-time environment. Test and retool along the way using an approach similar to steps 1 through 4. This will likely require minor reworking of scripts to use netflow gathering techniques over static pcaps.

USER CONSUMPTION

It is suggested that an existing user interface / front end be selected versus building a new, dedicated interface for flow analytics. Creating a new front end would be inefficient, and it would fail to take advantage of the scale of other existing platforms.

METHOD TO DEFINE, DEVELOP, AND IMPLEMENT QUERIES

What an organization wants to do always exceeds what actual development and implementation resources allow. It is suggested that each desired query be administratively routed through this summary process. Then, the full list of queries can be scored/prioritized, and a cut line can be established and, finally, developed. Ideally, the organization will leverage agile methodologies to administer the development process, asking:

- What do we want to look for?
- Why are we looking for it?
- How do we want to find it?
- What are we going to do with the results?

WEAKNESSES

The approach is fundamentally based on incident handler / defender security techniques, so initially, it will have gaps:

- Sniffers must be turned on and appropriately instrumented, with an additional focus on lateral movement points, as most existing sensors will likely be inbound / outbound due to the protective IT model.
  - An edge-edge perimeter is likely.
  - Should include a server enclave perimeter.
  - Should include a desktop enclave perimeter.
  - Switches with important servers and such.
- (Security) Stage 1 and some Stage 2 exploits are nearly impossible to detect and are almost not worth the effort. Stage 2 will include C2 and crypto libs, and they must be transmitted in the clear, so these are a little easier to detect. Stage 1 is typically very tiny; e.g., buffer overflow.
- Much of the data involved will be subject to HIPAA and other regulations; therefore, an atypical amount of care will be necessary when handling and manipulating data to avoid violations or unnecessary protection costs.

SUMMARY OF TECHNICAL STRUCTURE

The following is a summary of the top-level details for an initial rollout. With these items and the proper instrumentation, one might have insight into as much as 60% to 75% of enterprise activities.

TOOLS

L2/3:	L4+:
arpwatch	bro & bro-cut
netflows	after-the-fact Snort
cam tables from important switches	tcpdump

TECHNIQUES

- Bro analytics logs for input (bro-cut to parse)
  - conn.log (network connections)
  - dns.log (resolutions)
  - files.log (any files moved or changed)
  - Use http.log ("internet" communications)

- Collaboration and correlation

- Timeline analysis; develop an understandable sequence of events without single machine session reconstruction. Flow reconstruction will be critical.

- Perform initial reconnaissance:
  - When starting and stopping
  - What things are the stations and hosts doing?
  - What protocols are involved? Any obvious abnormalities?
  - Who is who?

- Long tail review (least frequent events)

- Security scripting that looks for:
  - LLMNR games
  - Entropy detection for DGA and crypto of all sorts
  - Arp games

EXAMPLE COMMANDS AND METHODOLOGY

Following are a series of basic, standard commands that illustrate the "how" element of the technical implementation. As noted above, the approach focuses exclusively on scriptable command line inputs. This list is meant to be a sampling and is by no means intended to be exhaustive (or really even scratch the surface).

- **$ capinfos [name].pcap** looking to:
  - Start and stop timestamps
  - Establish PCAP duration
  - Establish a baseline for the timeline analysis

- **$ bro [name].pcap** will prepare the capture for use with bro-cut commands and create conn.log, files.log, dns.log, http.log plus many others not referenced in this paper

- Bro uses **tcp[13] & 7 != 0** to capture any TCP packets with SYN, FIN, or RST control bits set. These packets delimit the beginning (SYN) and end (FIN or RST) of each TCP connection. Because the TCP/IP packet headers contain considerable information about each TCP connection, using just these control packets, one can extract connection start time, duration, participating hosts, and the number of bytes sent in each direction. Thus, by capturing only 4 packets (the two initial SYN packets exchanged and the final two FIN packets exchanged), one can determine a great deal about a connection even though one has not looked at any data packets.

EXAMPLE COMMANDS AND METHODOLOGY (cont.)

- **$ wc -l conn.log** tells us how many conversations have been made. This is an example of a normalizing value that can, with future inputs, be used as part of a statistical analysis. It also lets us see how "heavy" a segment might be.
- **$ cat conn.log | bro-cut id.resp_p | sort | uniq -c | sort -nr** will provide a count list on the responding port; this will identify the chatty ports to be correlated against the expected list of ports.
- **$ cat conn.log | bro-cut id.resp_h id.resp_p | grep [port to be investigated] | sort | uniq -c | sort -nr** will provide a list of investigated ports with responding IPs; i.e., who is listening and on what port.
- **$ cat conn.log | bro-cut id.orig_h id.orig_p | grep [port to be investigated] | sort | uniq -c | sort -nr** will provide a list of investigated ports with originating IPs; i.e., who is talking and on what port.
- The previous commands can be augmented with the bro-cut **-d** or **-u** flag to incorporate timestamping in local or UTC formats. This is important because, from a flow perspective, it will allow for reconstruction of communication channels. However, it will initially require heavy human interaction. For example, once Device A converses with Device B, Device A sends a UDP status packet to Device C.
- **$ cat conn.log | bro-cut id.resp_p | grep '25\|110\|135\|139\|143\|[plus many others]' | sort | uniq -c | sort -nr** provides a list of Windows services that probably shouldn't be part of a control system network segment, as their existence indicates a comingling of business and control functions.
- MDE and FRCS are deterministic and may use UDP; the commands to parse this information include **udp_request** and **udp_reply.**
- (Security) **$ cat files.log | bro-cut file_hash | sort** will provide a list of MD5 hashes for files that can be submitted to Virus Total or be run against the local AV / HBSS. Note: the hash activity requires consumption of the data packets though can be performed without actually inspecting contents. Hashes meet the intent of the effort and allow for greatly reduced storage and regulatory protections as the data packets are processed then dropped.
- Bro-cut further supports regular expressions, so how parsing is created is limited only by the imagination.
- Funnel off malformed header packets. Inherent to bro, this allows for a stack of "What are these?" packets to be manually reviewed and coded against.