UNCONSTRAINED ENDPOINT SECURITY SYSTEM: UEPTSS

Fatema Bannat Wala[1,2] and Chase Cotton[1]

[1]Department of Electrical & Computer Engineering, University of Delaware, Newark
[2]University of Delaware, Newark, USA

ABSTRACT

Modern information security management best practices dictate that an enterprise assumes full configuration control of end user computer systems (laptops, deskside computers, etc.). The benefit of this explicit control yields lower support costs since there are less variation of machines, operating systems, and applications to provide support on, but more importantly today, dictating specifically what software, hardware, and security configurations exist on an end user's machine can help reduce the occurrence of infection by malicious software significantly. If the data pertaining to end user systems is organized and catalogued as part of normal information security logging activities, an extended picture of what the end system actually is may be available to the investigator at a moment's notice to enhance incident response and mitigation. The purpose of this research is to provide a way of cataloguing this data by using and augmenting existing tools and open source software deployed in an enterprise network.

KEYWORDS

Endpoint security, device fingerprinting, scanning, inventory, BRO IDS, exploit.

1. INTRODUCTION

Some organizations cannot control some or all of their end user computer systems. Example organizations include universities, shared offices and start-up spaces, sites offering public Internet access (e.g. restaurants), and conferences. It is still assumed the enterprise is still responsible for the protection of infrastructure (servers, routers, switches, security devices, etc.) as well as end user computers used by staff performing critical business functions (e.g. human resources, financial records, personal information, etc.).

As such, the enterprise will be running modern external perimeter and internal protections strategies. Part of this security infrastructure will include the collection, logging, and analysis of sensor data inside and on the perimeter of the enterprise network.

Parts of this data will contain important information about end user systems in the enterprise including the systems not managed by the central organization. We call these unmanaged end user systems "unconstrained" systems. Part of the job of incident response in this enterprise will be to detect, triage, and investigate anomalous security events seen in the enterprise. In many cases these events will likely involve unconstrained systems.

The Unconstrained Endpoint Security System (UEPtSS) is a software system to fingerprint 'Unconstrained' end user computer systems/devices connected to the enterprise network, and is especially useful for the organizations with a bring your own device (BYOD) policy [1]. The reason it is important to fingerprint as much data as possible for an unmanaged device is it gives a complete picture to an organization of what kinds of devices are being used on the enterprise network and what services are running on those devices. Having this information pertaining to the devices also helps in malware incident response, as one of the most vulnerable security controls in an enterprise network which supports a BYOD policy, is the potentially untrusted software

DOI: 10.5121/ijnsa.2018.10201

running on the end user's device, as the enterprise has the least control on patching and keeping that software up to date so as to mitigate the risk of exploitation of latent vulnerabilities. A successful exploitation on one of these devices can help the attacker to pivot into the network and to use a chain of exploits against the crucial enterprise controlled and managed systems, as it is usually for the enterprise to 'trust' internal network and devices connecting to their network, often with additional firewall leniency provided to those systems as well. Hence, a vulnerable endpoint could potentially be low hanging fruit for the attacker to get a foothold inside the enterprise network and to pivot into the network to target next to the crown jewels of an enterprise. Hence, it's critical to keep track of all devices that connect to the enterprise network.

There are two commonly known solutions available to fingerprint these unconstrained endpoints that join the network:

Active Scanning: As the name suggest, the active scanning technique is the method to scan the devices currently connected on a network segment by actively sending different packets to the IPs in the given range and then analysing the responses to make a best guess of what the endpoint device could be and what kind of services it could potentially be running. This method is very common in especially determining the operating systems on various devices connected to the network. There are numerous open source tools such as Nmap[2], Nessus[3], etc. that provide the feature of active scanning of subset of the network segments.

- Pros: Active scanning can be accurate when using custom scripts to determine the software or service running on a device, but these capabilities might only be available for commercial scanning products.
- Cons: The main drawback of the active scanning is that the device must be active on the network, i.e. connected to the network when scanning is performed, which is very difficult in case of unconstrained devices, as they join and leave the network on random times. Thus to fingerprint the endpoints, the scanning must be run frequently (or even continuously) to catch all the devices that join and leave the network. And that on-going scanning can act as an unintentional DOS attack[4] on the network users, yielding a major drawback associated with this type of continuous active scanning.

Passive Scanning: On the other hand, the method opposite of active scanning is passive scanning. In this technique, normal traffic generated by devices on the networkis collected and analysed. Inspection of this traffic allows one to make a best guess of what kind systems and services that are connected to the network.

This technique is very commonly used by the IDS (Intrusion Detection Systems) [5] typically found running on the perimeter of the network. There are various open source tools that can be leveraged to fingerprint the endpoints by sniffing the traffic, like BRO IDS[6], Snort IDS[7], Suricata[8], etc.

- Pros: One of the main advantages of the passive scanning is that it is plug and play, i.e. no knowledge of when and where the devices will be connecting to the network is required, since as soon as any device connects to the network, it will start generating the network traffic to communicate on and use the network, which can then be sniffed to determine the presence on the device. Unlike active scanning, it is non-intrusive, i.e. and there is no threat of DOSing any legitimate user or services on the network.
- Cons: Since it is dependent on observed traffic patterns, all the cons can be applicable to passive scanning exist, specifically the reliance on (known) signatures. A potential attacking end user device can easily modify the traffic pattern to fool the passive scanning fingerprinting.

We will be focusing on the passive scanning technique to build an inventory of the unconstrained endpoint systems connecting, or ever connected, to the enterprise network, detecting, but not limited to, and following information pertaining the devices:

- Machine type (PC, Mac, smartphone, access point, printers, odd stuff, etc.)
- Operating system and version (Windows 7, OS X, etc.)
- Browsers in use (User Agent strings)
- Open ports (services)
- Applications and versions
- Dangerous behaviour history (prior loads of known malware)
- Different Plugins (Flash, OpenSSH)

What information is required for the fingerprinting and how to analyse this information to fingerprint the device will be discussed in next sections.

2. RELATED WORK

Various researches has been done in an effort to draw attention to how crucial it is to have knowledge of one's network, especially in terms of endpoints and IoT devices present on the network, and on an efficient way to detect those end points running in the environment. Rugg and DeLeeuw in "Increasing Security by Focusing on Endpoints" [9], point out that one of the most vulnerable targets that an attacker can leverage is the internal endpoints of a network, and that securing them is one of the key challenges faced by educational organizations, and especially for those environments where the organization doesn't have the full control of all the endpoints connecting to the network. This research, however, is more focused on system hardening and securing the individual endpoints, by various methods such as encrypting the laptops, deploying centrally managed anti-virus software, and making sure that the systems are in compliance with their PCI policy etc. Our research addresses a more realistic approach of cataloguing endpoints, by passively collecting all the relevant logs that can help identify endpoints in the network for use in the follow-up incident handling after an attack has been identified, when that kind of information is required the most, rather than by the almost impossible task of managing or hardening them individually in an uncontrolled environment.

Yang, et. al. [10] focus on the secure authentication and data communication between IoT devices by using an RFID-enabled solution aimed at protecting endpoints in the IoT supply chain. The research mainly focuses on how authentication can be secured in IoT supply chain management by focusing on the part of the application that can connect and talk to the network. The majority of the work is focused on mitigating the risk in a limited set of use cases, which is more applicable in an environment or organizations that share similar use cases and have resources to implement effective mitigation steps. A more generic approach introduced by Tokuyoshi [11], helps users understand what the implications are for BYOD policy supporting organizations and describes the security challenges of monitoring, vetting, and auditing these BYOD devices, and further highlights the measures for securing the network from un-managed BYOD devices. The work focuses on important points like enforcing application policy, device policy, and protecting data on the device that can help mitigate the risk. But again, the authors conclude that these are very limited measures, and securing the network in the first place and providing means to connect securely to it are required next generation security measures for BYOD devices. The research described in our paper closely relates to and extends the ideas suggested by Tokuyoshi, by providing means of collecting and cataloguing the information available to the security analysts to take a deeper look at the un-managed devices and to be able to take actions accordingly.

3. BUILDING THE UEPTSS

When working on this research, we primarily focused on BRO IDS system to gather the majority of information required to build an inventory for UEPtSS. The reason behind utilizing BRO IDS is it is open source and free software that can be easily deployed and installed for a proof of concept (POC) experiment. Also, it comes with built-in scripts deployed with the software to detect various software and services running on the systems. It also has strong scripting [12] and logging [13] frameworks which support the writing of custom scripts to determine various patterns in the traffic, and to generate user friendly logs. More information on BRO IDS can be found at their site [14].

Other than BRO IDS, there are other various passive scanning tools that can be used for gathering information for fingerprinting devices, such as Snort, Suricata, etc. Apart from open source and free tools, if the enterprise already has commercial IDS/IPS systems deployed for network security monitoring and control, then the logs from those security systems can also be potentially used to determine the software and services running on enterprise endpoints, such as the logs from a network based firewall, or web application firewall, or logs from the endpoint management system.

For the convenience and implementation perspective, BRO was chosen to do the POC of this research, as it is free and open source project, and hence can be used with minimum deployment cost in the enterprise.

3.1. LEVERAGE BRO IDS SCRIPTING FRAMEWORK

As mentioned earlier, the BRO IDS has a strong scripting framework. Leveraging it for detecting various software and operating system versions can be highly useful. BRO comes with some built-in scripts that detect various kind of software by analysing traffic patterns, which are enabled by default in the nominal configuration. Apart from these default scripts, we have written some custom scripts to detect the Mac operating system, and OSX for iPhone. There are few scripts that are available in custom written package (called Scan-NG package), that we have used to determine open ports on hosts. Details on which scripts have been used for fingerprinting are shown in Table 1.

3.2. LEVERAGE BRO IDS LOGGING FRAMEWORK

There are various logs that get generated whenever BRO determines any particular network traffic pattern, and that information related to the pattern is written in the ASCII log files. When additional or custom the scripts of interest are loading or enabled in the BRO IDS system, they also generate logs for the corresponding detections and write them to various log files. Hence, it is important to know which log files to look at while gathering information for UEPtSS. As the scripts themselves will do no good if they are not writing the correct information in corresponding log files. Details on which log files to analyse for fingerprinting are shown in Table 1.

Table 1. BRO IDS Scripts and Logs

Scripts to load	Logs generated in file	Information logged
Windows-version-detection.bro [15] (built-in)	software.log	Operating system and version
Mac-version-detection.bro[16] (custom)	software.log	Operating system and version
iPhone-detection.bro[17] (custom)	software.log	Operating system and version
Host-profiling.bro[18] (Scan-NG package)	site_host_open_ports.log	Open ports on a host
Software-browser-plugins.bro[19] (built-in)	software.log	Different browsers and plugins
Known-services.bro[20] (built-in)	known_services.log	Known services
Software.bro[21] (built-in)	software.log	Various servers/applications and versions

4. AN INVENTORY OF UNCONSTRAINED ENDPOINTS

Once the above mentioned scripts are loaded and running in the BRO IDS, and the sensors are placed at a point in network site to be able to observe a majority of the overall traffic, or even the complete traffic of the enterprise, then the logs will start getting generated and useful information can then be extracted from the log files, to build an inventory of endpoints and their characteristics. Furthermore, it is important to note that the key, e.g. as in a database index, in the log files used to identify a given software or a service corresponding to a device, is the device's IP address. And as we know that an IP address can be reused and mapped to many devices during a period of time on most enterprise networks implementing dynamic addressing (e.g. DHCP), it is important to know the MAC address of the device that was using that particular IP during the time of fingerprinting in order to actually pinpoint the device that was running the software or service at given point in time. Hence, to get the information regarding the MAC address, the best place to look at is the enterprise DHCP logs. We have used the logs from the DHCP servers, to map the MAC addresses to the corresponding IP addresses while fingerprint the network.

Also, to collaborate with the detected OS version by the scripts, we have used the freely available MAC OUI vendor listing from IEEE Standards Public listing (MA-L) from their website [22], to determine the machine type (manufacturer) for the system i.e. Apple, Dell, etc.
A discussion of the information gathered from the various log files to build the inventory follows.

4.1. GATHERING INFORMATION REGARDING OS AND VERSION

When the Operating System detection scripts are loaded, they start generating the logs in the software.log file corresponding to the network traffic patterns seen by the device. This information is useful, as most exploits are targeted towards very specific OS type and do not unlock themselves when the underlying OS detected is not same as the target OS for the vulnerability exploitation. Figure 1 is a screenshot showing some of the OS detected by the script.

Note the extraction command line at the top of this and other figures (1-5). Note also the left column in these figures is the time of the observation in Linux epoch time [23].

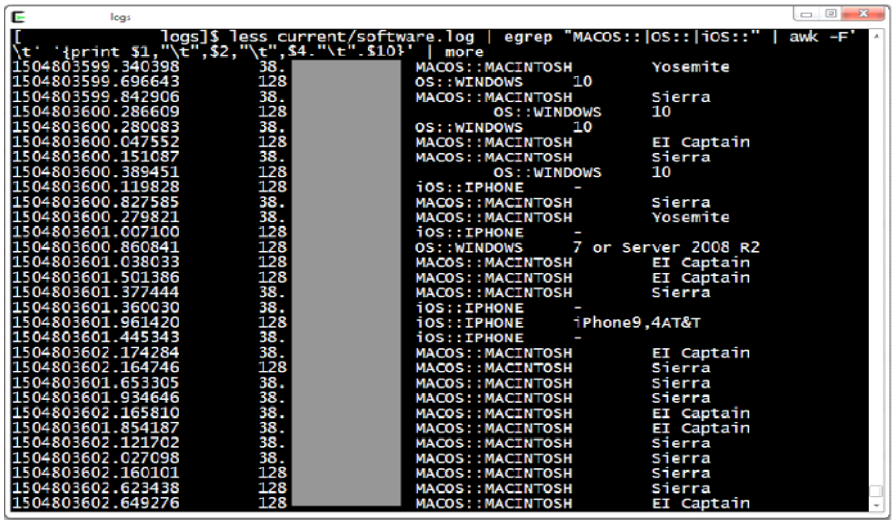

Figure 1. Operating System Detection

4.2. Gathering Information Regarding Browsers In Use

There are many vulnerabilities found in different browsers. Hence if the browsers that the device was using during the incident response is known, the scope of the infection or potential exploit can be further narrowed down, specifically in the case of browser specific vulnerabilities. Figure 2is a screenshot showing some of the browsers detected by the scripts.

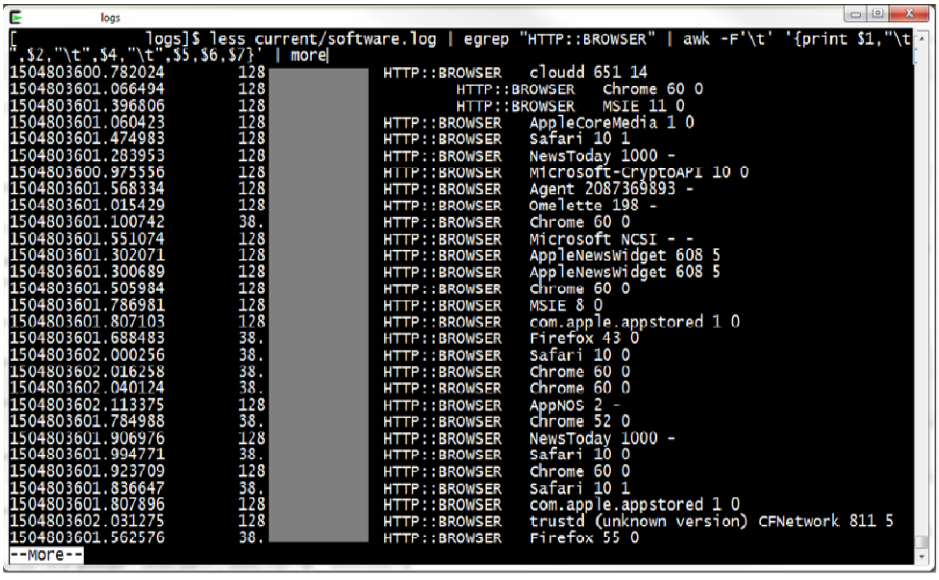

Figure 2. Browser Detection

4.3. Gathering Information Regarding Applications And Versions

It is also very crucial to know whether the endpoint is running the most up to date and supported versions of applications. This information can then be used for the policy enforcement by the

enterprise security and policy compliance to make sure that there are minimal unsupported versions of any software or application is running on the enterprise network. This use-case of policy enforcement will be discussed in more detail in the Usefulness section. Figure 3 is a screenshot showing some of the applications detected running on the endpoints.

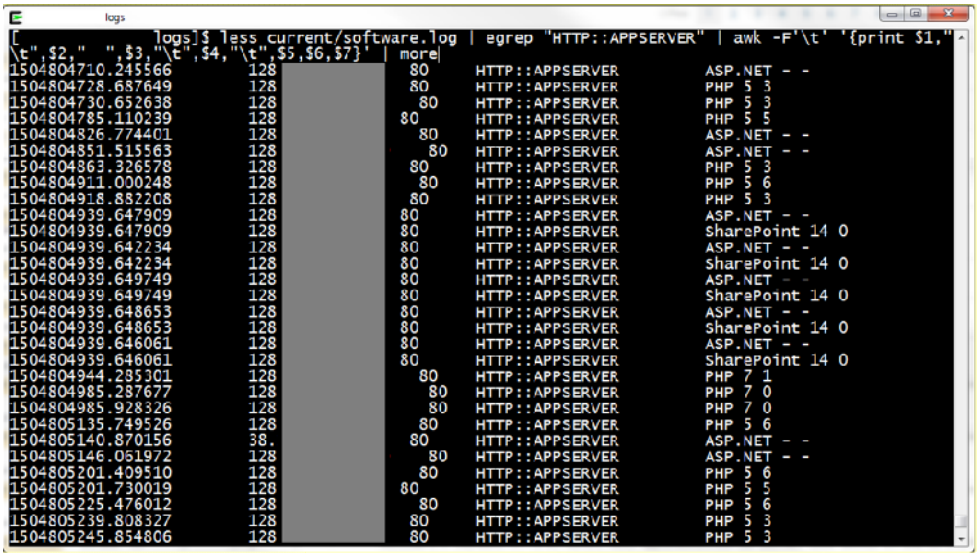

Figure 3. Software Version Detection

4.4. Gathering Information Regarding Different Browser Plugins

Sometimes when the user has installed various supporting browser plugins on their browsers, this information gets advertised whenever the user connects to the network using the browser. Hence, that information can be sniffed to determine what all browser plugins are available or in use by the endpoint. Many times the vulnerabilities are present in various plugins that could potentially be exploited and can cause harm to the enterprise network, hence this information, together with other fingerprinting information can be used during an incident triage. Figure 4 is a screenshot shows some of the plugin information detected on the endpoints.

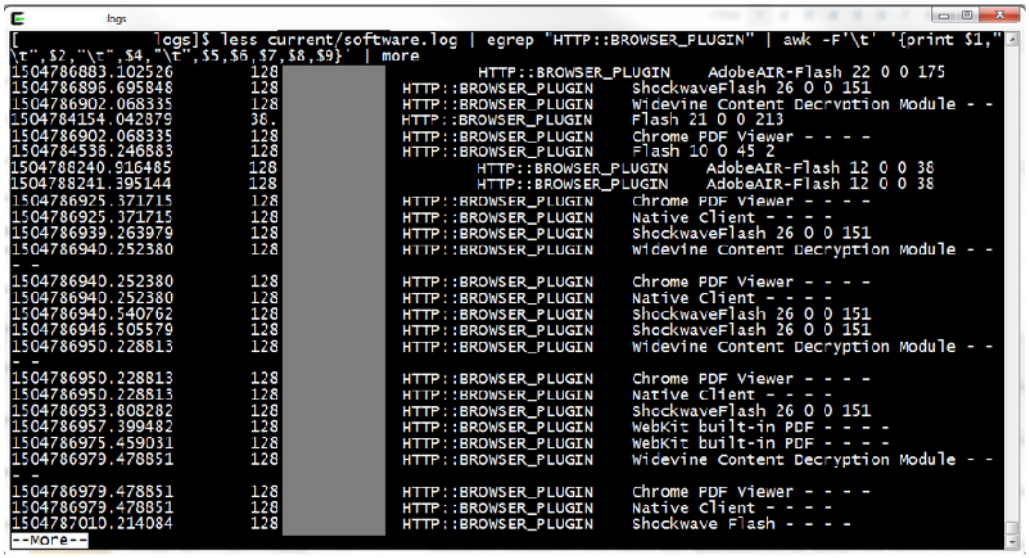

Figure 4. Browser Plug-in Detection

4.5. GATHERING INFORMATION REGARDING OPEN PORTS (KNOWN SERVICES)

One of the most important pieces of information regarding an endpoint is open network ports open or the services it is running. This information helps in enumerating the types of services available on the network and all the systems or servers are advertising that service as available. Also, it is not typically common for user end systems to be running any kind of services like SSH, HTTP, or DNS. And hence this information could be useful to assess what services are open on the systems on a given subset of the network, and whether these systems should be running those services open to the internet or not under the enterprise security or acceptable usage policy. Again, this use case will be discussed in more detail in the Usefulness section to be discussed later. Figure 6 is a screenshot provides a view of the services running on some of the systems.

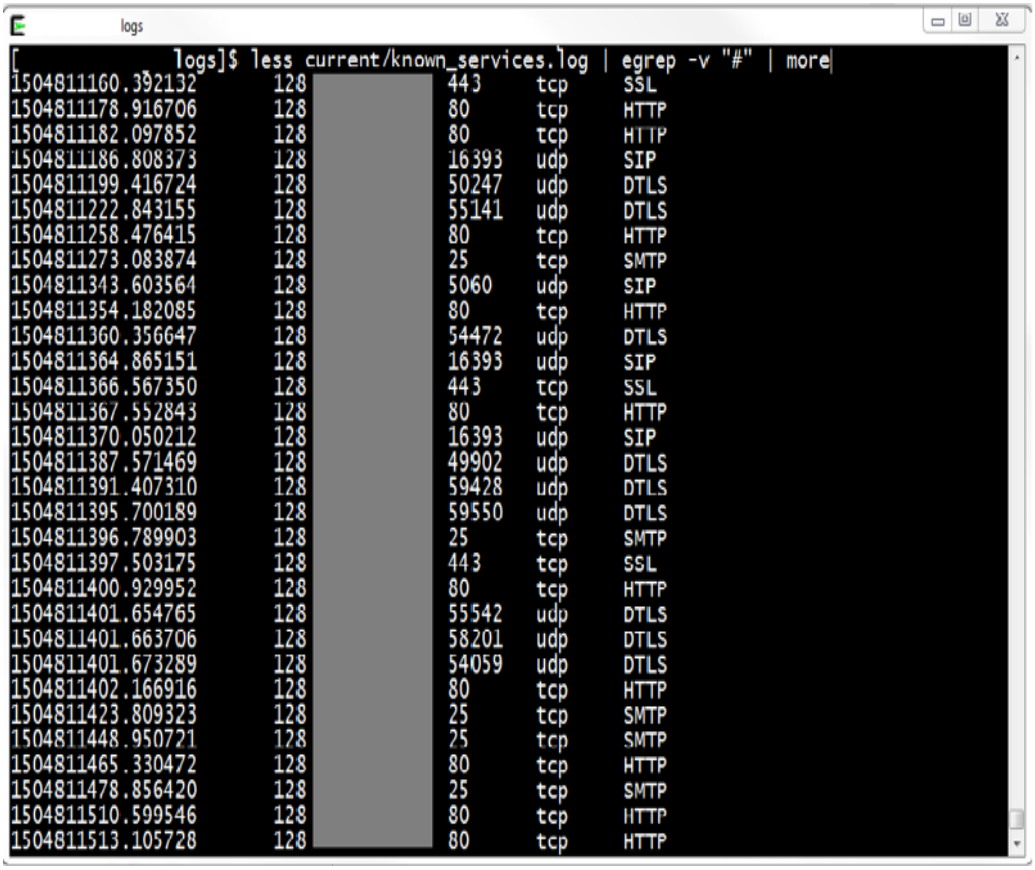

Figure 5. Service Discovery

4.6. PUTTING EVERYTHING TOGETHER

When the above mentioned component information is recorded in the log files, together with the machine manufacturer information and DHCP logs, a more complete picture can be realized corresponding the endpoints connecting to the network. Any log aggregation tool can be used to glue all the information together, with IP address being the primary key in each type of log file to get an inventory of unconstrained endpoints on the network. Figure 6 shows a screenshot of what an inventory of systems would look like.

client_ip	latest_time	mac	dmacs	vendor	known_services	software_type	software_info
128.4.	08/20/2017 15:01:33	10:41:7f	1	Apple, Inc.		iOS::IPHONE	iPhone,10,3,iPhone7,2AT&T
128.	08/20/2017 14:50:17	34:17:eb:	1	Dell Inc.	22,tcp,(empty)	SSH::SERVER	OpenSSH,5,3,-
128.	08/20/2017 15:06:18.763725	78:2b:cb:	1	Dell Inc.	22,tcp,SSH	SSH::SERVER	OpenSSH,6,6p1
128.	08/20/2017 14:54:20	00:1e:68:	1	QUANTA COMPUTER INC.	22,tcp,(empty)	SSH::SERVER	OpenSSH,5,5p1
128.	08/20/2017 15:04:22.440988	90:b1:1c:	1	Dell Inc.	22,tcp,SSH	SSH::SERVER	OpenSSH,6,6,-
128.	08/20/2017 14:51:21	14:da:e9:	1	ASUSTek COMPUTER INC.	22,tcp,(empty)	SSH::SERVER	OpenSSH,7,2p2
128.	08/20/2017 15:02:13	4c:cc:6a:	1	Micro-Star INTL CO., LTD.	22,tcp,(empty)	SSH::SERVER	OpenSSH,7,2p2
128.	08/20/2017 14:55:01	98:90:96:	1	Dell Inc.		OS::WINDOWS	Windows,10,,10
128.	08/20/2017 14:39:30	14:fe:b5:	1	Dell Inc.		OS::WINDOWS	Windows,10,,10
128.	08/20/2017 15:06:00.641491	e0:9d:31:	1	Intel Corporate		OS::WINDOWS	Windows,6.1?? or Server 2008 R2
128.	08/20/2017 15:06:21.108368	ac:87:a3:	1	Apple, Inc.		MACOS::MACINTOSH	Macintosh,1?,10,Yosemite
128.	08/20/2017 14:16:03.344146	ac:87:a3:	1	Apple, Inc.		MACOS::MACINTOSH	Macintosh,1?,12,Sierra
128.	08/20/2017 15:02:41.363188	10:9a:dd:	1	Apple, Inc.	22,tcp,(empty) 80,tcp,HTTP	HTTP::SERVER SSH::SERVER	Apache,2.4,Unix OpenSSH,6,5,-
128.	08/20/2017 15:06:02.732062	70:8b:cd:	1	ASUSTek COMPUTER INC.	22,tcp,(empty) 80,tcp,HTTP	HTTP::SERVER SSH::SERVER	Apache,2,4,Ubuntu OpenSSH,7,2p2
128.	08/20/2017 14:59:34.233089	00:50:56:	1	VMware, Inc.	80,tcp,(empty)	HTTP::SERVER	Microsoft-HTTPAPI,2,0,-
128.	08/20/2017 15:05:14.453996	54:9f:35:	1	Dell Inc.	22,tcp,(empty) 3690,tcp,(empty) 443,tcp,SSL 80,tcp,HTTP 8080,tcp,HTTP	HTTP::SERVER	Apache,2.4,Ubuntu
128.	08/20/2017 14:43:03	00:1c:c0:	1	Intel Corporate		HTTP::BROWSER_PLUGIN	ShockwaveFlash,26,0,-

Figure 6. Endpoint Aggregate Fingerprint

5. USEFULNESS OF UEPTSS

There are three main use-cases where the inventory of unconstrained endpoints is very useful. They are as follows:

5.1. POLICY ENFORCEMENT

As mentionedbefore, the inventory can be used to determine which applications and their versions are detected on the enterprise network. This information can be used by the Security and Compliance team of the enterprise to enforce any particular policy and to make sure that there is no unsupported software or application is running on the network. For example, from the inventory, a list of the systems running old OpenSSH[24] can be easily determined by searching for 'SSH::SERVER' keyword, and a notice can be sent to the owner of those devices to update the software and comply with the policies with the enterprise. Another example could be searching for OpenSSL[25] versions from the software.log file and can get a list of the endpoints running old versions of OpenSSL library. Also, if the enterprise has any particular policy for users to be running specific Operating Systems then policy enforcement can also be used to ensure that the systems are running recommended operating system by the enterprise.

5.2. ENUMERATING SERVERS/SERVICES

Another big advantage of having an inventory of systems running various services, is whenever somebody wants to know how many servers are on the network that are running any particular service, then this can be quickly found out by searching the inventory for that particular service. For example, if the security analysts want to know how many DNS servers are running on the network, or how many HTTP servers are running on the network for enumerating purposes, or to restrict the users from running well known services on their locally managed devices. This will be quick and easy as compared to doing an active scan of the network for that service (like port 53 for DNS or port 80 for http), which could take hours of scanning and could potentially DOS, or overload the network. This use-case helps answer the questions like: what all servers are providing xyz service on the network, or what all systems have port xx open to the internet (where xyz and xx can be replaced by any service protocol or port).

5.3. MALWARE INCIDENT RESPONSE

One of the main use-cases we were looking at when coming up with the idea of UEPtSS, is that if the information pertaining to a particular system is saved as normal logging activity, then an extended picture of what the system is and whether it can be affected by malware, can be used for a preliminary analysis of the incident response. The majority of malware or exploits are targeted towards a particular OS version, exploiting a particular vulnerability. Hence, if the OS is known during the time of malware incident response, and what all services and versions were being run on the system, it can quickly help in steering the investigation in the right direction. For example one of the hyped ransomware infection, Petya, [26] is particularly targeted towards infecting Windows systems, and Mac OS X are not affected by it. Hence just knowing what OS the system is running can help in diagnosing the malware infection.

Apart from these three use cases, the inventory can be used for looking up information regarding any endpoint for audit or any other purposes. Also, open ports information can be used to audit the network firewall policies, to know whether the ports are seen open on the devices belonging to a particular subset of the network that is behind the firewall and shouldn't have any communication going on those ports. Hence, to have this information logged in an inventory would be very useful and help to investigate various network anomalies.

6. CONCLUSIONS

There are many commercial software systems available that provide for fingerprinting the systems on an enterprise network. All of them require either an agent to be deployed on the client systems (endpoints) or are based on active scanning. These solutions are targeted towards a limited set of endpoints, and sometimes have their charging model based on the number of endpoints that the enterprise wants to fingerprint or keep track of. Hence they work fine when the number of endpoints in any organization is mostly constant or don't change. Unlike organizations like Universities, where students come and go, some graduate every semester and many come as newly admitted. Hence, in this kind of environment where the number of end users and type endpoints keep on changing drastically, it becomes harder to deploy any commercial solution to keep track of the software/application running on almost all the devices that connect and leave the network. That was the motivation towards starting UEPtSS, as students always come and go, bring new devices to the University network. And more importantly a diversity of students bring diverse types of applications local to the students' native countries, which makes it even more important to keep track of what all applications are seen on the network, and whether they are kept up to date or not. Another challenge with any of the commercial solution is the consent of from the user to agree to deploy any third party software on their devices. And sometimes the user is not willing to agree to put any third party software on their system and hence an enterprise can get a lot of resistance in deploying any commercial solution globally in the network. And finally the last factor to consider in a commercially versus the open source UEPtSS solution is cost, as some of the commercial solutions pricing is directly proportional to the number of endpoints, and it becomes very expensive both regarding the pricing and amount of time and effort required to deploy and maintain these solutions.

REFERENCES

[1] R Afreen, "Bring your own device (BYOD) in higher education: opportunities and challenges", IJETTCS Publishing, 2014.

[2] Nmap.[Online]. Available: https://nmap.org/

[3] J. Beale, R. Deraison, H. Meer, R. Temmingh, C.V.D. Walt, Nessus network auditing, Syngress Publishing, 2004.

[4] H. Hasbullah, I. Ahmed Soomro, J.AbManan, "Denial of Service (DOS) Attack and Its Possible Solutions in VANET", World Academy of Science, Engineering and Technology, IJECE Publishing, 2010.

[5] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges", Computers & Security, Elsevier, Vol. 28, Issues 1-2, Pages 18-28, 2009.

[6] V. Paxson, "Bro: A system for detecting network intruders in real-time",Computer Networks, vol. 31, no. 23-24, pp. 2435-2463, 1999.

[7] Snort IDS.[Online]. Available:https://www.snort.org/

[8] H. Jiang, G. Zhang, G. Xie, K. Salamatian, and L. Mathy, "Scalable high-performance parallel design for network intrusion detection systems on many-core processors", Proc. Ninth ACM/IEEE symposium on Architectures for networking and communications systems (ANCS '13), IEEE Press, 137-146, 2013.

[9] B. Rugg, B. DeLeeuw, "Increasing Security by Focusing on Endpoints", SIGUCCS'17, ACM Annual Conference on SIGUCCS, Pages 145-148, 2017.

[10] K. Yang, D. Forte, M. Tehranipoor, "Protecting Endpoint Devices in IoT Supply Chain", ICCAD'15, IEEE/ACM International Conference on Computer-Aided Design, Pages 351-356, 2015.

[11] B. Tokuyoshi, "The security implications of BYOD", Network Security, Elsevier, Vol. 2013, Issue 4, Pages 12-13, 2013.

[12] BRO Scripting Framework.[Online].Available:https://www.bro.org/sphinx/scripting/

[13] BRO Logging Framework.[Online]. Available:https://www.bro.org/sphinx-git/frameworks/logging.html

[14] BRO Documentation.[Online]. Available: https://www.bro.org/documentation/index.html

[15] Windows version detection.[Online].Available: https://www.bro.org/sphinx/scripts/policy/frameworks/software/windows-version-detection.bro.html

[16] Mac version detection.[Online]. Available: https://github.com/fatemabw/bro-scripts/blob/master/Mac-version-detection.bro

[17] iPhone detection.[Online].Available: https://github.com/fatemabw/bro-scripts/blob/master/iPhone-detection.bro

[18] Scan NG Package.[Online].Available: https://github.com/initconf/scan-NG/tree/master/scripts

[19] Software Browser Plugins.[Online].Available: https://www.bro.org/sphinx/scripts/policy/protocols/http/software-browser-plugins.bro.html

[20] Known Services.[Online]. Available: https://www.bro.org/sphinx/scripts/policy/protocols/conn/known-services.bro.html

[21] Software detection.[Online]. Available: https://www.bro.org/sphinx/scripts/base/frameworks/software/main.bro.html

[22] IEEE MA-L listing.[Online]. Available:https://regauth.standards.ieee.org/standards-ra-web/pub/view.html#registries

[23] Matthew Neil, Stones Richard,"The Linux Environment". Beginning Linux Programming. Indianapolis, Indiana, US,Wiley. p. 148 (2008).

[24] M. Stahnke, Pro OpenSSH, Apress, 2006.

[25] J. Viega, M. Messier, P. Chandra, Network Security with Open SSL:Cryptography for Secure Communications, O'Reilly, 2002.

[26] Alert (TA17-181A) Petya Ransomware, July 28, 2017, [Online].US-CERT, Available: https://www.us-cert.gov/ncas/alerts/TA17-181A

AUTHORS

Ms. Fatema Bannat Wala(MS ECE, UD, 2015; BE Electronics & Instrumentation Engineering, DAVV University, 2012;CISSP) is PhD candidate in theDepartment of Electrical and Computer Engineering at the University of Delaware researching cybersecurity issues. Ms. Bannat Wala, formerly a software engineer with Accenture, is currently a Security Engineer in UD's Technical Security Group (TSG) where she is responsible for the University's Intrusion Detection Systems and SIEMs. Ms. Bannat Wala speaks often at security industry forums and holds the CISSP and is a GIAC Certified Intrusion Analyst (GCIA) and GIAC Certified Penetration Tester (GPEN).

Over the past 30 years, **Dr.Chase Cotton** (Ph.D. EE, UD, 1984; BS ME, UT Austin, 1975; CISSP) has held a variety of research, development and engineering roles, mostly in telecommunications. In both the corporate and academic worlds, he has been involved in computer, communications, and security research in roles including communication carrier executive, product manager, consultant, and educator for the technologies used in Internet and data services.Since 2008, Dr. Cotton has been at the University of Delaware in the Department of Electrical and Computer Engineering. His research interests include cybersecurity and high-availability software systems with funding drawn from the NSF, ARL, CERDEC, JPMorgan Chase, and other industrial sponsors. He currently is involved in the ongoing development of a multi-faceted educational initiative at UD where he is developing new security courses and degree programs including a minor, graduate Master's degree, and graduate Certificates in Cybersecurity. Dr. Cotton currently consults on communications and Internet architectures, software, and security issues for many carriers and equipment vendors worldwide.