University of Amsterdam

Project Report

Behavioral profiling from network packet captures

March 31, 2020

Students:
Philipp Mieden
philipp.mieden@os3.nl

Tjeerd Slokker
tjeerd.slokker@os3.nl

Lecturer:
Jaap van Ginkel

Course:
Cyber Crime and Forensics
Security and Network Engineering

Contents

Abstract

In this paper we develop and evaluate a graphical link analysis based approach to forensic network traffic investigation, that accelerates incident response and simplifies communication to non-expert users. Traditional tooling is mostly targeted towards expert personnel, and visual representation of extracted information is hard to grasp for other user groups. Our solution overcomes this limitation by combining in-depth protocol analysis with device profiling and visualizing the results within the popular open source intelligence software Maltego. We evaluate the efficiency of the developed approach in form of a case study, by using it for the investigation of a series of network breaches within a small corporate network environment.

1 Introduction

Network traffic analysis for forensic analysis can be compared to searching for the needle in a haystack, as network communication produces huge amounts of digital evidence. While some things are hard to find, others are easy to miss. These circumstances require effective tooling and data correlation strategies, to allow processing such type of evidence in a reasonable amount of time and extract useful information. Driven by the increasing number of digital interconnected devices [15], the amount of data that is potentially interesting for investigators is constantly growing. TLS fingerprinting techniques continue to evolve and allow accurate identification for originators and providers of encrypted communication. Examples are *ja3* from salesforce and *joy* from Cisco. For plain-text communication Deep Packet Inspection (DPI) can be used to identify the application layer protocols based on the packet payloads, and data carving can be applied to extract information about transferred file formats [1]. Parsing support for application layer protocols is essential for an in-depth investigation, as understanding protocols such as DNS, HTTP or POP3 allows to paint an accurate picture of user activity and communication with other entities. In general, network security breaches require a fast response as decreasing the time from compromise to detection can greatly lighten economic damage. This requires efficient, verifiable and easy to use tooling.

1.1 Problem Description

Identifying user behaviour in network data that was acquired as digital evidence, is vital to reconstruct the subjects behaviour for the purpose of criminal investigation. However, this is manual and labour intensive work, that requires a significant amount of time and makes investigations slow and expensive. Retrieving enough information from the acquired data requires expert knowledge and specialized tooling. The combination of intelligence sources is a manual procedure as well, and should be automated to save valuable time. The visual representation of complex network architectures and data flows for a human analyst is also challenging, for the analyst during the investigative process and during the presentation of findings to less technical personnel. Due to the huge amounts of information, a suitable representation is important to avoid overloading the analyst with data.

1.2 Related Work

In 2008, Xu, Zhang, and Bhattacharyya [5] presented a clustering approach on network flow data to create behavioral profiles for several attacks, clients, servers, rare and deviant behaviour, targeting a use in network intrusion detection. Their approach focused on using traffic communication patterns without making any presumption on what is normal or anomalous. For our research project, we intend to go beyond flow analysis for profiling and use the packet contents to make assumptions about the actions inside the network.

In 2016, Alotibi, Clarke, Li, *et al.* [3] experimented with classification of user activity using a deep neural network. While a single user could be identified with a 100% accuracy in all test scenarios, for the majority results varied from 35-70%, leaving doubts about the accuracy being suitable for forensic investigation. We decided against the use of machine learning for our project, due to the lacking transparency in the decision making process for deep neural networks, and the computational overhead when training and evaluating models on the analysts machine.

In 2020, Fletcher [2] developed an integration for DPI into the Wireshark packet dissector and demonstrated its use for forensic investigations. This integration made use of Wiresharks built-in features such as analysis profiles, marking, filtering, annotation and colorization. This proved to be useful to increase the value for the analyst and speed up the process of data analysis. The implementation faced problems accommodated by the use of the Wireshark dissector, that ranged from difficulties with correlation of timeline data and intuitively displaying the resulting data for timeline analysis. We plan to overcome these limitations by using a framework for the implementation that gives us full control over the dissection and data analysis process and a specialized software for graphical link analysis.

1.3 Scope

We limit our scope to full network packet captures in the PCAP(NG) format [7], as those contain payloads that are most suited for forensic investigation and event reconstruction. Features for profiling will be timestamp information, network and hardware addresses, IP geolocation lookups, reverse DNS lookups, application Layer inspection for user profiling, and exchanged file format information as well as TLS fingerprinting to identify participants of encrypted communication. The proof of concept will support the linux and macOS platforms, and possibly in the following weeks also windows.

1.4 Outline

The rest of this paper is organized as follows: in Section 2, we will present our research questions and main objectives. Section 3 explains the methodology of the implemented approach, by establishing a theoretical framework and describes the actual implementation. Section 4 describes the data set and its entities and incidents, as well as the insights and strategy applied during our investigation with the developed tooling. Section 5 will mention limitations and address scalability concerns in regard to a possible use in the industry. In Section 6 will issue final thoughts and observations. Section 7 will list possible future enhancements and further useful optimizations.

2 Research Question

Our goal is to develop a structured approach to behavioral profiling of network packet captures for use in digital forensics, that aids criminal investigations by enriching and correlating information extracted from network packet traces. During this research project, we want to answer the following questions:

- How can the entity based investigation approach be applied for forensic network traffic inspection?
- Which impact does this approach have on the efficiency of the investigation?

3 Methodology

In order to develop a structured approach to behavioral profiling of network packet captures we will define a theoretical framework that describes the underlying procedures and logical flow of actions. We will implement the proposed approach using the NETCAP framework [6], and demonstrate its feasibility by evaluating it on a PCAP(NG) traffic dump [7]. We chose NETCAP for packet processing due to its open source code and configurable engine for packet processing. To analyze the resulting data graphically we will use Maltego, which is an open source intelligence (OSINT) and graphical link analysis tool for the purpose of collecting and analyzing information for forensic investigations [8]. We chose maltego due to its popularity and success in the field, as well as their straight forward way of allowing customization via local transforms. Link analysis is a powerful data exploration concept in digital forensics, that creates a structured presentation of interconnected objects. It is a central methodology for law enforcement and intelligence agencies. [12]. Maltego offers a free community edition, that has several feature limitations. As of writing this tool has no open source integration for the analysis of a PCAP(NG) file. For testing our approach we defined a theoretical framework that is explained in the following section. We evaluate our solution by analyzing the traffic from a network forensics workshop given by Erik Hjelmvik from the Swedish Armed Forces CERT, at the cyber security conference FIRST in 2015 [10] [11].

3.1 Theoretical Framework

To lead the investigate process, we have defined a generic series of procedures, that we will apply in an iterative fashion, to explore the data for the evaluation. It is inspired by the demonstrated approach of analysing the case files during the workshop.

1. Determine right time frame to research
   (a) If the data is too large, apply a *berkeley packet filter* [4] to pre select
   (b) Otherwise split the data by day to get a picture of a single working day
2. Load the selected PCAP file into Maltego and generate *DeviceProfiles* from it
3. Get hierarchical overview on PCAP, to answer the initial questions:
   (a) Which devices have been used?
   (b) At what time have the devices been active?
4. Use transformations to explore user behavior and convert human questions into queries over the raw data
   (a) What applications have been used by which addresses?
   (b) What emails have been sent between whom?
   (c) Which geolocations can be identified for the addresses?
   (d) What actions have been observed over HTTP?
   (e) What files did which host download?
   (f) What flows have been identified as suspicious?
5. Isolate entities and their relations of interest and create separate graphs for them
6. Rise, rinse, repeat

Figure 1 displays the theoretical framework for our investigation:

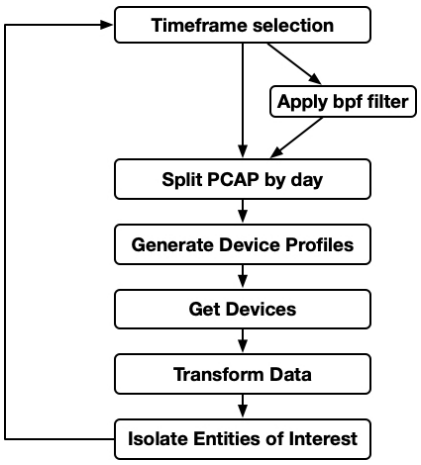

Figure 1: Theoretical framework for link based analysis of a computer networking security incident

3.2 Implementation

We implemented our approach in form of a new audit record type for the NETCAP packet processing engine. The *DeviceProfile* audit record type keeps track of network interactions for all source mac addresses, and builds up a profile with aggregated information about used IP addresses, contacted hosts, and network protocols. For resolving various identifiers to human readable strings, we implemented the resolvers package which offers an interface to resolve IANA port numbers to services, IPs to DNS names (passive and active), IPs to geolocation via the GeoLite database, MAC address to vendor, ja3 and ja3s TLS fingerprints to known services and to apply domain whitelist filtering. By collecting and aggregating information in this initial processing step, transformations during the investigation can be implemented as a fast query over the generated audit records. The generated data is encoded as protocol buffers, compressed and written to disk by netcap. To visualize the extracted information in Maltego, we implemented local transformations. A local transformation is essentially a simple command-line program, that gets called by Maltego with the value of the selected entity as an argument, together with additional entity properties. The program then processes the input and writes back a reply on the standard output conforming to Maltego's XML message model. In our case, all plugins are provided through a single framework binary *net.transform*, which gets called with the name of the transformation to execute, e.g. *GetDeviceProfiles*, as a first argument, followed by the data passed by Maltego. The initial transformation from a PCAP file to *DeviceProfiles* opens the dump file, configures a collector instance from NETCAP to generate only specific audit records, namely *DNS*, *DeviceProfile*, *Flow*, *HTTP*, *POP3*, *File* and *SMTP*, and creates an output directory for the new files in the same path as the input file, but with the added directory name extension *.net*. This directory also stores the extracted files structured by their detected MIME types in the *files* subdirectory for later analysis. We modeled 20 custom Maltego entities for netcap, that are used to represent the extracted data in the graph. Most of the entities carry a *path* property, that contains the filesystem path to the associated *DeviceProfiles.ncap.gz* file. This also allows to execute transformations over other audit record types, as the files for those are stored in the same directory. We've implemented a total of 40 transforms for Maltego, and created an easily extensible interface that allows to implement basic transformations over data provided by NETCAP with less than 50 lines of code. Transformations over several similar entities have been implemented by using inheritance for the target entity base types. An example for this is the *netcap.IPAddr* base entity, on which a majority of host transformations are operating on. The entities *InternalDeviceIP*, *ExternalDeviceIP*, *InternalContact* and *ExternalContact* are inheriting from the *IPAddr* base type and thus have all those host transformations available on them. The source code is available on github [9].

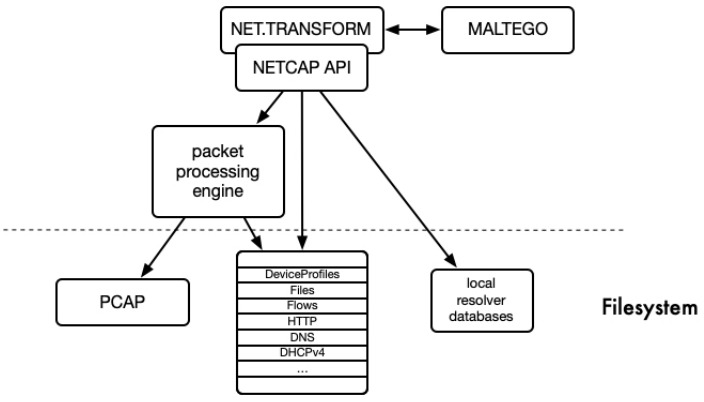

Figure 2: Transformation plugin architecture

4 Evaluation

To test our implementation we used a public data set made for a forensics workshop by the Swedish Armed Forces [10]. This data set contains network traffic which is captured at the network perimeter of a fictional corporation which was connected to the open internet. During the capture period of 40 days, several attack scenarios have been performed and recorded, along legitimate background traffic. The recordings are provided split by days, which fulfils the first step of our models requirements. The total aggregated size of all dump files is 4.2 GB. The network traffic is provided in a raw unaltered form with full payload information. Besides the traffic captures, there are also logs from the bro network security monitor [14], which we did not use for our investigation. Figure 3 shows the network map.

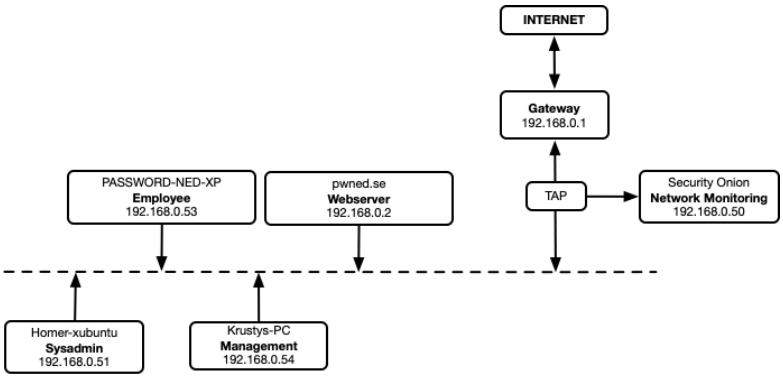

Figure 3: Network architecture of the network forensics data set

4.1 Users

As shown in Figure 3 the network contains six entities: three workstations, one webserver for hosting the companies webpage, the default gateway and a TAP interface. The three workstation computers belong to the employees. The first employee is the system administrator and his workstations hostname is *Homer-xubuntu*. The second computer is called *PASSWORD-NED-XP* and belongs to the employee Ned, who takes care of managing secure passwords for the companies clients. The last computer called *Krusty-PC* belongs to the PR/Marketing employee. The traffic is recorded before reaching the gateway at the indicated tap position by the Security Onion network monitoring instance, which is the 6th device visible in the captured traffic.

4.2 Incidents

The data set contains background traffic and multiple incidents. To test our implementation we will mainly focus on the investigation of the incidents, but during our test we also observed background traffic, including employee emails, web browsing and other application usage, such as Skype, Dropbox and HipChat traffic. Additionally there is background web browsing traffic with Facebook and search engine traffic. Various POP3 mail retrievals have been recorded, also some SMTP and IMAP traffic. The following incidents appear in the data set, in the listed order:

Incident 1 - Web Defacement A hacker collective named FrogSquad defaced the companies webserver and altered the homepage, to display a defacement image. The company is notified by a client who noticed the unusual look of the website.

Incident 2 - Malware The computer of the employee Ned gets infected with malicious software, and further components are being downloaded from multiple external hosts. We need to investigate where the malicious software came from. From forensic analysis of the employee workstation, the date of infection is known and also the name of a malicious file involved in the incident.

Incident 3 - Spear Phishing The threat actor group APT4711 send a spear phishing email to the marketing manager, who opened the malicious attachment which caused a trojan software to be downloaded and installed.

4.3 Results

Incident 1: Web Defacement Central questions in the first incident are how the defacement image was delivered to the server and from which IP address it came. In this scenario, the day of attack is known to be the 3rd of March 2015. We load the *PCAP* file for that day and generate the *DeviceProfiles*. The input *PCAP* file has a size of 37,2 MB and the generated audit records and extracted files make up for 33,6 MB of disk space in this case. The extraction took 4 minutes and 46 seconds, the file containing the audit records has a size of 337 KB. Using the *GetDevices* transformation profile all mac addresses will be added into the graph. To see which IP addresses these devices have been using, we run the *GetDeviceIPs* transformation. This adds the corresponding IPv4 and IPv6 addresses to the graph. Based on the provided network infrastructure map we add notes to the devices for known network components, such as the webserver, router and gateway, and users such as the sysadmin, the marketing manager, and the employee. Afterwards, one device is left, for which internal IP *192.168.50* we don't have any information in the network map. From the DNS information of the leftover device, obtained using tshark passive hosts file generation, we can see that the device has the hostname *sniffer.local*, so it's likely the Security Onion instance used for obtaining the actual traffic dump. Now that the networks local users are identified, we proceed with selecting an entity for further investigation. Figure 4 shows the current state of the graph in hierarchical topology.

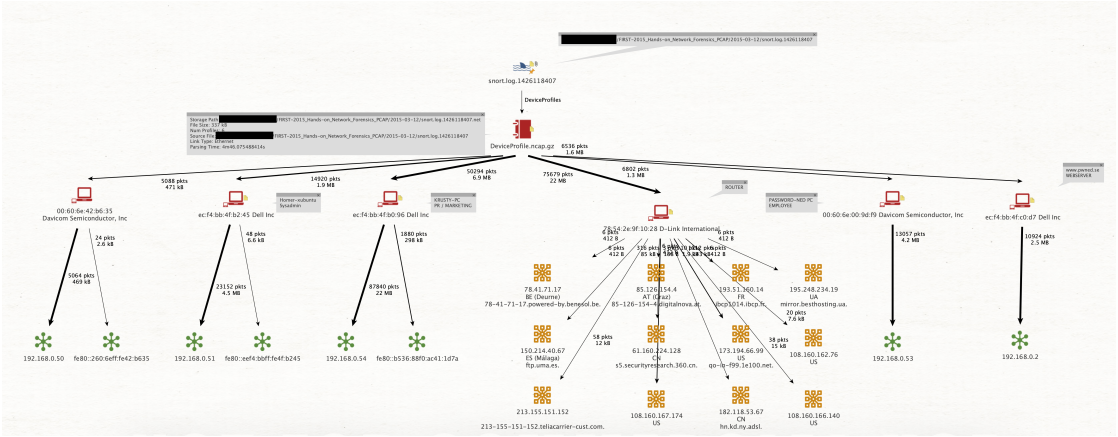

Figure 4: Hierarchical graph after GetDeviceIPs

Since the webserver has been defaced, we choose this component for further inspection. Using the *GetFileTypes* transformation on the internal IP address *192.168.0.2* of the Dell webserver, we can expand the graph with further nodes for each seen MIME filetype for that host. Among executable file formats (*application/octet-stream*), we see several image formats that have been transferred, namely *image/png*, *image/jpeg* and *image/gif*. Selecting

these image nodes and using the *GetFileForContentType* transformation, we add the actual image into the graph. We find the defacement image and look at the added meta information, which shows that this file was served as a HTTP response from the host *217.195.49.146* to the companies webserver, which is suspicious, because we don't expect to see client behavior from the webservice. Using the *GetDeviceContacts* transformation, we retrieve all hosts that have been contacted by our webserver. Maltego summarizes entities of the same type that exceed a predefined number of elements in collections. We query the generated collection of *ExternalContacts* for the source IP address of the suspected image transfer, and place the matched host out on the main graph for further investigation. From the added host geolocation and reverse DNS information we see that the attacker has an IP address from Latvia, Riga registered at the Balticom ISP.

Now that we identified a suspect, we use the *GetHTTPUserAgents* transformation to retrieve the User Agents seen for the host. As a result we yield a windows and Linux user agent, which indicates the use of virtualization. The Linux agent originates from the Iceweasel browser, which is preinstalled on the Kali Linux distribution. We add the geolocation to the graph, by running *GetGeolocation*. Next, we utilize the *GetApplications* transform to retrieve the OSI layer 7 protocols seen for the suspect host. We find SSH, FTP and HTTP activity. The *GetHTTPHosts* transform returns only a single accessed HTTP host: *pwned.se*. *GetHTTPServerNames* returns all unique values seen for the HTTP *Server* header field for the selected host. This reveals the string *Apache/2.4.6 (CentOS) PHP/5.4.16*, which is likely the webserver used to provide the defacement file by the attacker.

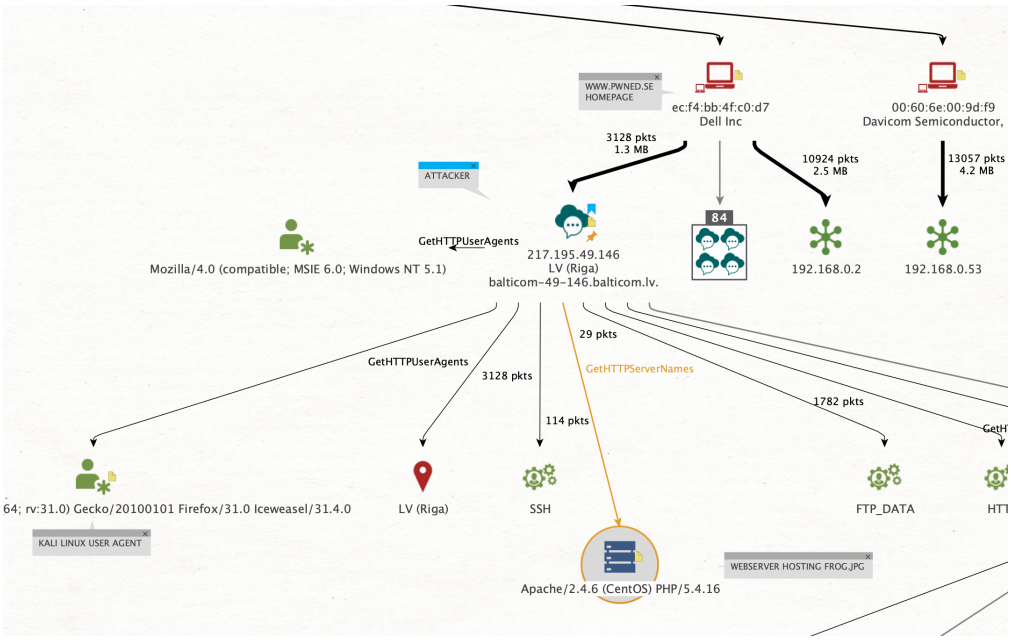

Figure 5: Attacker entity

Using *GetHTTPURLs* on the attacker IP entity, we get a list of URLs seen from that host. In this list, two unusual paths stand out: *pwned.se/cm0.php* and *pwned.se/skyblue/cm0p.php*. We retrieve all seen HTTP parameters for the attacker host with *GetHTTPParameters* and expand the graph with all unique values seen for the parameter names with *GetHTTPParameterValues*. Immediately an indicator of compromise shows up: there are Linux shell commands transferred via HTTP parameters. We start with the *name* parameter, that contains commands Perl code to open a socket and connect to the attackers IP, as well as shell commands to ping the attacker and check if the connection establishment was successful.

An internet search for the parameter name and the name of the used PHP webserver

SkyBlueCanvas, reveal that this is an exploitation of *CVE-2014-1683*, a command injection vulnerability, where the attacker can control the contents of the *$msg* PHP variable via the name HTTP parameter.

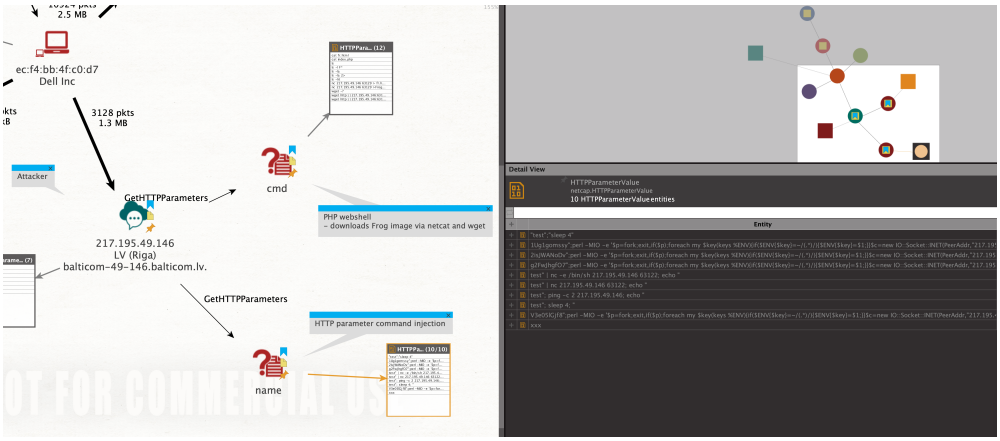

Figure 6: HTTP parameter injection

The second parameter with shell commands, *cmd* contains the commands to download the defacement image via *wget* and *netcat*. Figure 7 shows a table view of the executed commands in Maltego.

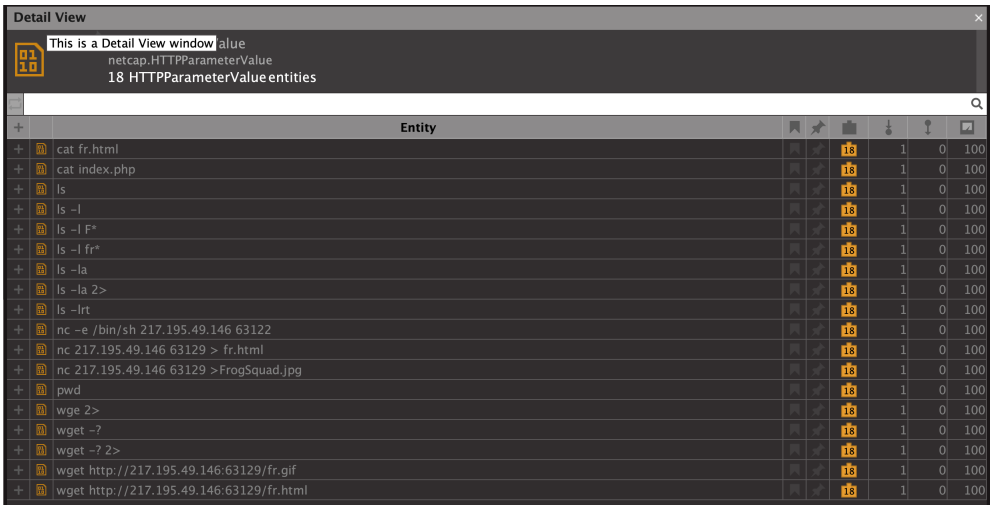

Figure 7: PHP webshell commands

With the *GetFilesForContentType* transformation, we also fetch the HTML files seen for the attacker host and analyze the HTTP responses the attacker received from the infected webservice, from the suspicious */cm0.php* URLs. We find the code from the installed PHP backdoor being transferred, as well as responses from using the webshell to execute Linux commands. Figure 8 displays a zoomed out graph overview during the investigation in hierarchical and organic topology. The organic arrangement has shown to be especially useful for larger amounts of entities with many interconnections, which lead to excessive horizontal growth in the hierarchical variant.

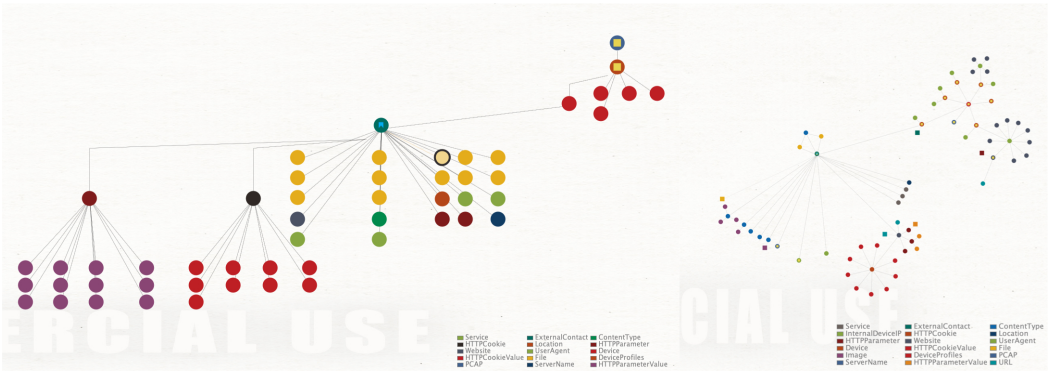

Figure 8: Hierarchical graph topology on the left and the organic variant on the right

To answer the question whether the intruders came back at a later point in time, we filter the entire PCAP data set for the suspect subnet *217.195.49.0/24* and load the resulting PCAP file into Maltego. Now only two devices show up in the graph: the webserver and the router. When running the *GetDeviceContacts* transformation on the router, we receive three hosts from the suspected subnet. After extracting the transferred formats and files of the webserver as described above, we can analyze the generated meta information for updates to identical entities over time. A first entity of interest here is the *skyblue.gz* file from *applications/x-gzip*, which is the compressed webserver response for the */skyblue* url. On the Netcap Info section in the Detail View, we observe several accesses from the attacker IP to the webserver on the *11th, 12th, 16th and 19th of March*. Additionally we look at the extracted source and destination ports for the suspect hosts, using the *GetSrcPorts* and *GetDstPorts* transformations. From the detail view we can again observe HTTP accesses from before, but also see SSH access on the *11th and 12th of March*.

Incident 2: Malware We need to determine where the malicious software found on the employees workstation originated from. For this we will apply flow analysis, and reduce the number of results using whitelist domain filtering. Additionally, we adapt the *GetIncomingFlowsFiltered* transformation to return only the top 12 results, based on the amount of transferred bytes. From the returned results we start to analyze the highest volume flows from top to bottom. The largest data transaction of size 1.6 MB is observed from the IP address *193.9.28.35*. The second largest download is from *1.web-counter.info* and of size 1.4 MB. The third largest download occurred from *www.mybusinessdoc.com*. From the stream meta informations, we noted down the point in time and date for each, as well as IP address information. To retrieve the files, we load the PCAP for the identified date of transfer 7th of March, and run the *GetHTTPContentTypes* transformation on the InternalIP address of the infected employee. We utilize the search graph functionality of Maltego, to search for occurrences of the suspect domain IP addresses in the entities and their meta information in the current graph and analyze the results. We look at the files from the flows in question, and use the *GetFileType* transformation on the transferred files, which returns the file type *PE32 executable [...] for MS Windows*. From the meta information we find the path to the extracted files on disk and upload them to Virustotal, all three matching positive. The state of the graph is shown in figure 9. When opening the webpage downloaded from *193.9.28.35* from the extracted files folder, we notice that the HTML consists of only HTML and body tags with an embedded base64 encoded payload. This is a common obfuscation technique for script based malware, and is used to hide identifiable strings from signature based detection methods.

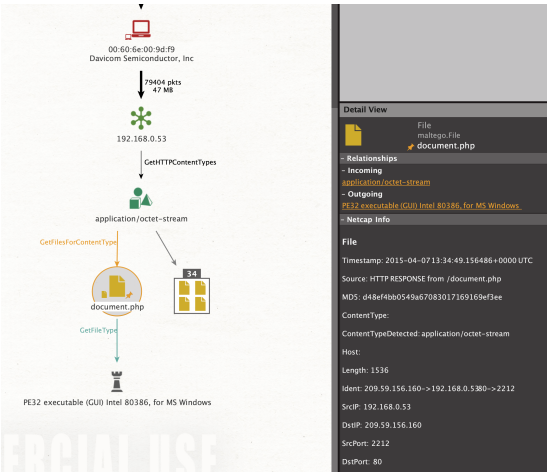

Figure 9: Real type of suspicious file is PE executable

To determine how the second malicious file *Delivery_Notification_00000529832.zip* file landed on the employees computer, we load the PCAP file for the provided date of execution: 2015-04-07. We expand the graph up until the IP address of the workstation *192.168.0.53* and use the *GetMails* transformation to add all mails fetched via POP3 from the infected host to the graph. Only a single mail appears for that day, that was sent from the manager to the employee. Inspecting the mail contents in the detail view, we can see an attachment with the filename *Delivery_Notification_00000529832.zip*, with the file contents being encoded in base64. We copy the base64 string, decode it and write the result into a file on disk, to retrieve the malicious javascript file from the compressed zip archive. Figure 10 displays the flow graph, with the incoming flows on the top and the outgoing flows on the bottom.

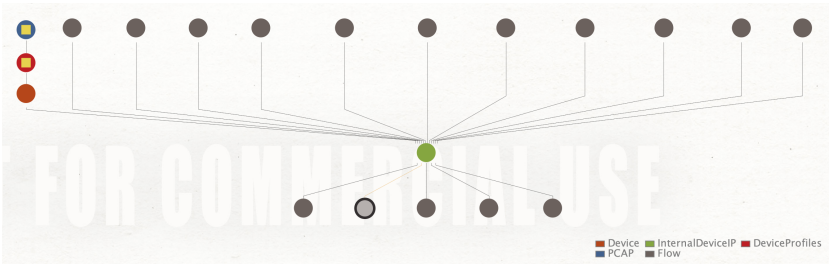

Figure 10: Flow graph

Incident 3: Spear Phishing We load the dump file for the 18th of March and retrieve the marketing managers IP address *192.168.0.54* via *GetDeviceIPs*. Using *GetIcomingFlows-Filtered* and *GetOutgoingFlowsFiltered* on this entity, and studying the resulting flows and their port numbers, we notice that there are connections to unsual ports on our webserver, indicating client behavior of the webserver as in the previous incident. Using *GetSrcPorts* on the webserver further hardens this suspicion. All of the identified flows belong to the address *103.10.197.187*, extracting it from the *DeviceContacts* and using *GetGeolocation* reveals the geolocation HK, China. The HTTP UserAgent seen for this host is the same Iceweasel browser identifier as from the previous incident. The deep packet inspection via *GetApplications* shows unusual results for the suspects address, namely *NONSTANDARD_HTTP* and *NO_PAYLOAD*. The implemented solution is currently not able to analyse the contents of the TCP connections, apart from the deep packet inspection. Due to time constraints, we had to stop the investigation at this point, leaving the remaining bonus questions of the workshop unanswered.

5 Discussion

5.1 Limitations

Current limitations include the lack of parsing support for transferred emails over IMAP and SMTP, which leads to missing emails in the case study. The free edition of Maltego only allows a maximum of 12 results per transaction, and drops these without presenting a warning to the user. However, since Maltego deduplicates results for us, the same transformation can be executed several times yielding up to 12 new results per execution. Loading PCAP files into Maltego is not yet optimal, the entities need a path property, to inform the *GetDeviceProfiles* transformation where the file resides on disk. After adding the file to Maltego, this property has to be set by hand. The path property will appear when the files entity type is set to *netcap.PCAP*. Due to the use of the deep packet inspection libraries *nDPI* and *libprotoident* via C bindings, cross compilation for different platforms is complicated, which is the reason why there is currently no support for windows, but for Linux and macOS. Fragments of the entire data, such as a PCAP dump for a single day for example, often do not contain enough DNS traffic to identify all local hosts within the network. To solve this problem, we gathered the DNS information from the entire available data a priori using *tshark* from the Wireshark toolsuite and applied it with the passive DNS resolver from NETCAP. This also drastically increases throughput, since all lookups can be done locally in memory, without contacting external DNS resolvers. Looking at file contents is currently not possible within Maltego, the *OpenFile* transform can be used to pass the file to the default application registered to the OS for handling the files type.

5.2 Scalability

NETCAP and the developed transformation set is entirely written in the high level system programming language Go, which is a modern language designed to take advantage of multi core processing and a safe garbage collected runtime. The NETCAP engine is implemented to take advantage of this from the ground up, and distributes the workload of packet decoding to a configurable pool of workers. The *DeviceProfile* audit record type is potentially impacted by lock congestion, as state information about devices and IP addresses has to be kept at runtime and shared across the asynchronous packet decoding worker pool. Furthermore, disk storage space is a limiting factor, as audit record and especially file extraction and automated decompression can result in generated data that has several times the size of the input PCAP data. Durations for dump file processing in our experiments varied due to system load. For example, processing the 39,7 MB large file *snort.log.1426636809* from incident 3 with deep packet inspection enabled took 88 seconds and resulted in a 1.4 MB *DeviceProfile* large audit record file. With DPI enabled, the packet processing rate can drop to a few hundred packets per second and the process can take twice as long. This heavily depends on the type of traffic and was out of scope of our research. It is likely related to the *cgo* bindings used but still subject to our ongoing analysis. Batching the C invocations could help to mitigate this issue, by collecting a pre-configured number of packets for a flow before calling into the C library for classification. The dataset was processed on a 2018 MacBook Pro with 32 GB 2400 MHz DDR4, 2,9 GHz Intel Core i9, running macOS 10.15. To avoid resource intensive processing on the investigator workstations, generated device profiles and extracted files can be shared across multiple investigators, as can be Maltego graphs. For larger data sets it can also be considered to offload the *DeviceProfile* generation to a dedicated server and manage jobs using job queuing. Careful preselection often eliminates the need for processing the entire available data, but of course this is not possible in every scenario.

6 Conclusion

The evaluation in our case study has shown that a graph based analysis strategy can be used to quickly analyze and understand malicious activity in a small corporate environment. To apply the approach it was necessary to model the Maltego entities using inheritance, to allow using transformations across several base types. Investigation of the provided *PCAP* data occurred in an interactive and constructive way, led by asking questions to pinpoint events and interactions of entities on the graph. Especially the hierarchical ordering was helpful in visualizing relationships that led to detecting the incidents. Compared to traditional tooling seen in this field, such as *Wireshark* and *Network Miner*, that are targeted towards expert users, we can imagine that our solution is suitable for the use by non-technical personnel, while being extensible and configurable for security analysts and engineers. Our approach combined the human ease to deal with an investigative process using questions and linked entities, rather than querying raw data manually or in a semi-assisted manner. However, as described in the theoretical framework, for larger amounts of data, splitting or filtering the data will be necessary to avoid drowning in information. We see potential in the implemented solution to help investigators accelerate their analysis of network traffic captures, and ease communication of key events to non-technical personnel through a visually appealing representation.

7 Future Work

Adding parsing support for further mail protocols, especially *IMAP* would increase value for the proposed integration during a criminal investigation, as emails do account for a large share of online communication. For convenience, file extraction in NETCAP should also be extended to extract files from parsed emails. Transformations for local threat intelligence for the extracted files could help in faster triage, such as *yara* signature scans or an integration with alerts generated by *suricata* or *snort*. Numerous integrations for online services exist already, for example Virustotal. Maltego offers a feature named machines to chain transformations, to automate the use of several transformations in a row. This can be very useful since large data sets can result in thousands of nodes, and therefore even the task of applying transformations to selected nodes, can require a notable amount of time. Future research could evaluate the use of such machines to improve investigation efficiency. Whitelist filtering could also make use of the *ja3* hashes for filtering away known legitimate TLS connections and hosts, instead of only using domain names for this purpose. Also the TLS fingerprinting technique *joy* from Cisco could be integrated for identifying hosts that use encrypted communication against their provided fingerprint database, which could further improve the identification of encrypted traffic streams [13]. Text extraction transformations could help to retrieve further informations from text based content, such as IP addresses, domains, links, phone numbers, names, emails and more. In general, the updates from the NETCAP transformations to the Maltego display information sections will be enriched and unified, to ease tracking updates to entities over time.

References

[1] S. Davidoff and J. Ham, *Network Forensics: Tracking Hackers through Cyberspace*. Pearson Education, 2012, ISBN: 9780132565103. [Online]. Available: https://books.google.nl/books?id=a6zoPYHLLiYC.

[2] D. Fletcher, "Forensic timeline analysis using wireshark giac (gcfa) gold certification", *SANS Institute Information Security Reading Room*, 2020.

[3] G. Alotibi, N. Clarke, F. Li, and S. Furnell, "Identifying users by network traffic metadata", *International Journal of Chaotic Computing*, vol. 4, Dec. 2016. DOI: 10.20533/ijcc.2046.3359.2016.0013.

[4] S. MCCanne and V. Jacobson, "The bsd packet filter: A new architecture for user-level packet capture", p. 11, Dec. 1992. [Online]. Available: http://www.tcpdump.org/papers/bpf-usenix93.pdf.

[5] K. Xu, Z. Zhang, and S. Bhattacharyya, "Internet traffic behavior profiling for network security monitoring", *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1241–1252, 2008, ISSN: 1558-2566. DOI: 10.1109/TNET.2007.911438.

[6] P. Mieden. (2019). Netcap - a framework for secure and scalable network traffic analysis, [Online]. Available: https://github.com/dreadl0ck/netcap.

[7] T. I. S. IETF. (2004). Pcap next generation format rfc, [Online]. Available: https://www.tcpdump.org/pcap/pcap.html.

[8] Maltego. (2020). Maltego, [Online]. Available: https://www.maltego.com/.

[9] P. Mieden. (2019). Netcap network traffic analysis framework, [Online]. Available: https://github.com/dreadl0ck/netcap.

[10] E. HJELMVIK. (2015). Hands-on network forensics workshop, [Online]. Available: https://www.first.org/conference/2015/program#hands-on-network-forensics.

[11] ——, (2015). Hands-on network forensics workshop slides, [Online]. Available: https://www.first.org/resources/papers/conf2015/first_2015_-_hjelmvik-_erik_-_hands-on_network_forensics_20150604.pdf.

[12] A. Årnes, *Digital Forensics*. Wiley, 2017, ISBN: 9781119262404. [Online]. Available: https://books.google.nl/books?id=Fk0nDwAAQBAJ.

[13] Cisco. (2019). Joy - a package for capturing and analyzing network flow data and intraflow data, for network research, forensics, and security monitoring., [Online]. Available: https://github.com/cisco/joy.

[14] V. Paxson, "Bro: A system for detecting network intruders in real-time", *7th USENIX Security Symposium*, 1998.

[15] Gartner. (2020). Gartner forecasts global device shipments will grow 0.9% in 2020, [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-01-21-gartner-forecasts-global-device-shipments-will-grow-0.