Enhancing Enterprise Security through Cost-effective and Highly Customizable Network Monitoring

Joshua Regenold
Department of Computer Science
Sam Houston State University
U.S.A.
jpr011@shsu.edu

Kai Wang
Department of Computer Science
Georgia Southern University
U.S.A.
kwang@georgiasouthern.edu

Gary Smith
Department of Computer Science
Sam Houston State University
U.S.A.
csc_gws@shsu.edu

Qingzhong Liu
Department of Computer Science
Sam Houston State University
U.S.A.
liu@shsu.edu

Lei Chen
Department of Information Technology
Georgia Southern University
U.S.A.
lchen@georgiasouthern.edu

ABSTRACT

Network monitoring and network traffic analysis software are common tools used in an enterprise, giving IT administrators valuable insight into the status of their servers and network devices. Limited research has been done to highlight the security benefits of low-level network traffic logging and analysis, though much of it involves testing the network activity of malicious software in lab environments, using cost-prohibitive software to analyze traffic for a pre-determined amount of time. This is a useful way to isolate network activity to only the malicious software, but it also eliminates valuable baseline traffic information for an enterprise network. There are significant security benefits to be gained from analyzing how malware reacts in – or alters – an enterprise network. This paper provides techniques for getting a baseline of enterprise network traffic and analyzes how different types of malware can affect this baseline. Using only low- and no-cost software and services, we analyze the storage requirements for historical network traffic data and present techniques to filter out much of the noise, significantly reducing the amount of data that must be stored and analyzed. The results of our technique are compared against traditional anti-malware and network traffic analysis methods, revealing our approach to be a cost-effective, highly customizable and effective

MOBIMEDIA 2017, July 13-14, Chongqing, People's Republic of China
Copyright © 2017 EAI 978-1-63190-156-0

1 INTRODUCTION

Sensitive information – including business secrets, consumer information, credit card numbers, and more – is increasingly being moved from a paper to a digital format. As a result, data security is a critical component of any enterprise. Typical data security methods focus on preventing the leak, exposure, or theft of any sensitive information. Firewalls, intrusion detection systems (IDS), intrusion prevention systems (IPS), and strict file access policies can help ensure that only the right people see certain data. However, cyber-attacks are on the rise and several high-profile network breaches have highlighted the need for more advanced data protection techniques.

Preventative techniques are necessary and useful for any enterprise and should be considered the first line of protection in a defense-in-depth strategy. However, the detection of and response to a potential network or data security breach are critical components of business security that are often either overlooked altogether or severely underutilized. There have been several high-profile, well-documented instances of malware infiltrating an enterprise network and stealing sensitive information for months on end before being detected and, eventually, removed. [1]

When properly implemented, advanced network monitoring and traffic analysis can help quickly detect and contain security breaches before the damage becomes irreparable. Typical network traffic analysis is often performed only for a short period of time and in search of specific problems – problems which are more often related to performance issues rather than security issues. One of the primary reasons network traffic analysis is underutilized as a security tool is because the storage requirements and complexity of network traffic captures for an entire enterprise can quickly balloon to levels that make its use an administrative nightmare. This paper presents a novel technique that can be used to ensure only security-relevant network traffic is captured and analyzed, resulting in significantly reduced storage requirements for keeping historical network data.

This paper is structured as follows. Section 2 is a background of traditional network monitoring and traffic analysis and its use – or lack thereof – as a security tool. Section 3 features popular network traffic analysis software. Emphasis is placed on where to position network traffic analysis hardware and software within the enterprise topology for efficient and thorough captures. Security-focused capture filters are presented to ensure chatty network protocols do not inflate traffic captures to an unmanageable size. Section 4 is the initial setup and a brief discussion on capture sizes. Section 5 covers the database structure and size. Section 6 is gathering a network traffic type baseline and performing quick internal checks. Section 7 discusses where to obtain external blacklist and IP geolocation data, in addition to how to import into and compare them against the traffic capture database. Section 8 is the impact malware has on network traffic. Section 9 closes with conclusions, limitations, and future work possibilities.

2 TRADITIONAL NETWORK MONITORING

Network monitoring has existed in various forms for several years. There are many free and commercial software products available that serve different purposes, such as Nagios, Cacti, Zabbix, Spiceworks, SolarWinds, PRTG, Observium, and more. [2-8] The features of each piece of software vary widely and it is not uncommon for an enterprise network to use more than one monitoring solution for specific types of devices, applications, and services; i.e., Nagios for the up/down status of network devices, Cacti for historical bandwidth reporting, and Spiceworks to monitor services on Windows servers. At its most basic level, network monitoring is simply alerting and reporting on the status of network devices. When a certain device triggers a pre-set condition – interface bandwidth has reached a certain percentage or a service on a mission-critical server has stopped, for example – a notification is sent to an administrator, ideally an administrator who will act upon the alert. This type of monitoring is a critical component of business continuity because very nearly all enterprises rely on network and server uptime to operate as efficiently as possible. Similarly, network traffic analysis has existed for several years, but its focus has largely been on troubleshooting specific network performance or connectivity issues. Even a small enterprise network can generate an enormous amount of traffic, so any network traffic capture is often narrow in focus and only between two devices – a server and a client computer or a router and a switch, for example.

Another common use of network traffic analysis is runtime monitoring of malicious software. [9, Sec. 1] This type of network traffic analysis is done in an isolated, controlled environment to prevent the malicious software from spreading to other machines on the network and is an effective way of isolating traffic to only what is generated by the malware. Authors Rossow et al. present a system called Sandnet that enhances this analysis technique by allowing the malware to run for an extended period of time. [9, Sec. 3] While Sandnet is an improvement over traditional dynamic malware analysis methods, it is unfortunately still limited in that many of the 100,000 samples the authors tested would not run in the test environment and the malware that did run did not have the opportunity to coexist with enterprise network traffic in a realistic manner. It is not uncommon for malware to attempt to spread itself to other machines on a network or to interact with local servers in some way, and Sandnet, due to the isolated and controlled environment, would never have the opportunity to discover such network activity.

Compared to wired networks, wireless networks present a unique set of security challenges. Wireless access points (APs) operate like hubs from several years ago, in that they do not isolate traffic on a per-client basis [10], as is the case with Ethernet switches. All clients connected to a wireless access point can see traffic from all other clients. In addition to this vulnerability, it is difficult to physically restrict access to a wireless network, because they operate using radio waves instead of switch ports and, as such, often extend beyond the walls of a building. Security for a wireless network, then, depends nearly entirely on the encryption of data between the client and the access point. While encryption helps to prevent snooping outright, authors Fragkiadakis et al. note, "…regardless the strength of the security algorithm, and the effectiveness of the intrusion detection system, an adversary can still overhear the wireless channel and identify different periodic components by observing the encrypted traffic." [11] In simpler terms, this means that regardless of the encryption method used, an attacker only has to take note of where packets are coming from, where they are going, and in what intervals to gather useful reconnaissance information about a network such as routing protocols, operating systems in use, and more.

Network monitoring of wireless networks typically consists of little more than monitoring access points and controllers for a number of associated clients and alerting an administrator when that number exceeds a pre-defined maximum – which creates performance issues – as well as the usual bandwidth and up/down status of the APs themselves. This is useful for performance monitoring, AP placement, and estimating future bandwidth and hardware needs, but does little to enhance network security. However, when a device is passively listening in on a wireless network, it will often either intentionally or unintentionally respond to certain packet types such as ICMP. [12] If a wireless AP receives a packet back from a MAC address that does not match the ARP binding that is already in place, this could signal that an attack is taking place or, at the very least, that a device is listening in on wireless traffic. This type of traffic can be captured using network traffic logging and analysis.

3 NETWORK TRAFFIC ANALYSIS SOFTWARE AND PLACEMENT

Network traffic analysis software allows a user to view all network traffic that travels to and from a machine. There are several different pieces of software available and they vary in feature set and price. One of the most popular tools used is Wireshark, formerly known as Ethereal. [13, 14, 15] The figures and examples used in this paper will all be from Wireshark. Wireshark has two different types of filters: capture and display. Display filters, as the name implies, only filter traffic from actually displaying in the Wireshark analysis window – they do not prevent the traffic from actually being captured and stored. Traffic filtered with a display filter still consumes space to store and processor cycles to analyze. Capture filters, on the other hand, discard uninteresting traffic so that it is neither stored nor analyzed. Wireshark uses Berkeley Packet Filter (BPF) syntax to filter out uninteresting traffic. As a result, the capture filter examples used in this paper are portable and can be used in any network traffic analysis software that supports BPF syntax.

3.1 Capture Filters and Storage Requirements

A simple capture filter to focus only on traffic destined to an external – i.e., Internet – destination is

(port 53) or (ip and (dst net not 10.0.0.0/8 and dst net not 172.16.0.0/12 and dst net not 192.168.0.0/16 and dst net not 224.0.0.0/4 and dst net not 240.0.0.0/4 and dst net not 255.255.255.255/32 and dst net not 169.254.0.0/16))

This is a deliberately long-winded, versatile capture filter that can be easily updated if traffic destined for internal subnets needs to be captured as well. The dst net portion of the filter is an abbreviated form of destination network and the not tells the capture filter to discard that traffic. Subnets can be included in the filter in standard classless inter-domain routing (CIDR) format. The capture filter above excludes all standard private subnets, as well as broadcast and multicast subnets, thereby focusing only on traffic destined to – or return traffic from – the public Internet. This is a useful security capture filter because malware often contacts a command-and-control server at a location on the public Internet. This filter will capture that traffic and discard all chatty, non-DNS internal traffic.

Using the same test network previously discussed, Table 1 displays the results of a five minute capture both with and without the example filter.

Table 1. Capture results with and without capture filter

Capture Filter	File size	Packet count
Without capture filter	33.81MB	137,204
With capture filter	5.37MB	34,662

The capture filter clearly has a significant impact on both the packet count and file size of the capture. If viewing the traffic in Wireshark, display filters can be used to further remove redundant traffic from the capture, such as

not tcp.analysis.retransmission and not tcp.analysis.duplicate_ack

For the capture purposes outlined in this paper, the interesting information from network captures is the destination IP address. This display filter will remove all unnecessary or duplicate TCP requests to non-unique destination IP addresses, thereby reducing the number of packets that must be analyzed while still ensuring all unique destination IP addresses are captured and analyzed.

3.2 Traffic Analysis Machine Placement

As discussed previously, in the majority of enterprise networks today, switches are used in place of hubs. Switches offer both superior performance and increased security when compared to hubs, in that each switch port is isolated from all other ports, which makes snooping far more difficult and also prevents collisions from occurring. While this is undoubtedly desirable in nearly all deployment scenarios, it does make deploying a traffic analysis machine more difficult. Connecting a network traffic analysis machine to a switch that has been configured with no special concessions will allow the machine to view nothing more than its own traffic as well as any broadcast traffic on the subnet or VLAN the machine is currently using.

To effectively and thoroughly capture all traffic on a specific subnet or VLAN, a switch must be configured with port mirroring. Port mirroring allows certain switch ports to be designated as either source or destination ports. Traffic from designated source ports is mirrored to the destination port or ports. When the switch port connected to a router sub-interface in a router-on-a-stick configuration is used as a source port, this allows for all routed traffic on a subnet to be mirrored to the destination port(s), which is the switch port that the traffic analysis machine will be connected to. It is worth noting that a traffic analysis machine must have more than a single NIC, because when a switch port is configured as a destination port, it will no longer pass traffic from the machine connected to it – it simply mirrors all traffic from the designated source ports. Figure 1 illustrates a simple network topology with a traffic analysis machine deployed.

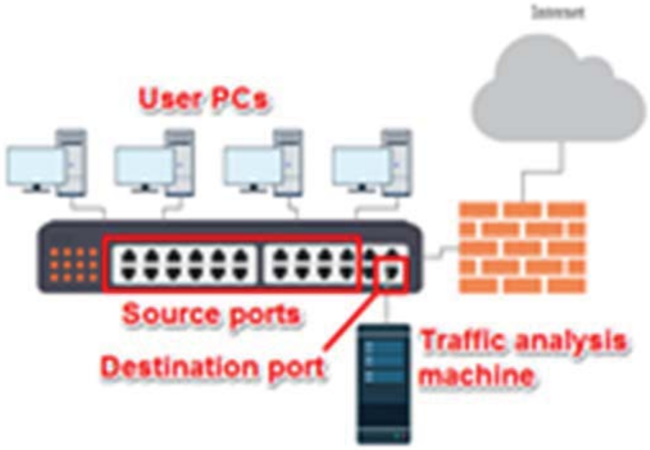

Figure 1: A simple topology with a traffic analysis machine.

In this example, only user PCs have their ports mirrored and, therefore, their traffic replicated to the destination port with the traffic analysis machine connected. In many enterprise networks, it is also critical that high-priority servers have their traffic mirrored to the traffic analysis machine to ensure they are appropriately monitored. It is possible, as discussed earlier, to mirror only the switch port connected to a router and capture all routed traffic as well. These different deployment scenarios highlight the flexibility that traffic logging and analysis permits, allowing this solution to be deployed in enterprises of all sizes and types.

4 INITIAL SETUP AND CAPTURE SIZES

The initial setup of the network monitoring and traffic analysis solution described in this paper is a multi-step process. We will be using the previously mentioned Wireshark for network captures, as well as the open-source tool C5 SIGMA, which "takes network packet capture data as input and produces a structured relational database." [16] C5 SIGMA is compatible with both SQL Server and MySQL, but our test deployment will use SQL Server exclusively. All queries, performance data, and storage requirement information for the remainder of this paper have been tested using SQL Server.

C5 SIGMA is command-line based and offers a great deal of flexibility, which makes scripting the import of capture files into a database a trivial task. The basic C5 SIGMA command used in our test deployment is

```
SIGMA.exe -in "..\captures" -out "..\captures\Output" -dbh "<database hostname or ip>" -dbu "C5" -dbp "pass" -ts "<tshark location>\tshark.exe" -pre Basic
```

The –dbu and –dbp switches allow for the use of an SQL Server authenticated user name and password, respectively. Windows integrated authentication is also available for increased security, by simply replacing the –dbu and –dbp switches with the –dbi switch. The –in switch specifies where traffic capture files are stored and the –out switch specifies where C5 will store the temporary files necessary to import information into the database. Note that even using relatively small capture file sizes, discussed next, results in C5 creating several temporary files that total approximately 1GB in size. Only a single set of temporary files is created regardless of the number of captures files being imported at once and these files can be deleted immediately once the database import is complete.

One issue we ran into with our test deployment was that C5 SIGMA would attempt to import capture files from Wireshark even while Wireshark was actively updating the open file. This would result in duplicate entries, because the file could not be deleted by our script – due to it being open and used in Wireshark – so the same data would be imported again during successive C5 runs. The workaround for this issue is to create a staging folder that Wireshark saves captures files in and then move those files to an input folder that C5 monitors, versus having C5 and Wireshark use the same single folder. The basic PowerShell script used in our test deployment is below.

```
Move-Item "C:\Users\Administrator\Desktop\staging\*" "C:\Users\Administrator\Desktop\captures\"
if (Test-Path C:\Users\Administrator\Desktop\captures\* -exclude output)
{
C:\scripts\C5script.bat
cd "C:\Users\Administrator\Desktop\captures"
Remove-Item * -exclude output
cd output
Remove-Item *
}
```

The Move-Item command moves all completed capture files into an input folder monitored by C5. This command will not move files that Wireshark has open, removing the risk of duplicate data. The second line of the script checks for the existence of capture files and, if found, runs the C5 command listed earlier in the paper, which we saved as a separate batch file called C5script.bat. Once the database import is complete, the script deletes the capture files as well as the temporary files created by C5. In our test deployment, we set this PowerShell script to run as a scheduled task once every five minutes, removing the need for user intervention beyond the initial setup.

With all of these prerequisites in place, the next step in the deployment is to ensure that the file size of the captures is going to be administratively manageable. Wireshark has a built-in feature that allows traffic captures to be split up either by file size or after a predetermined amount of time. For our test deployment, we tried several different capture file sizes, but the most efficient and usable configuration was to have Wireshark create a new capture file every 5MB. On our low-end database server outfitted with a mid-range quad-core processor, a standard 7200RPM hard disk, and 8GB of RAM, C5 SIGMA was able to import a 5MB capture file in approximately three minutes. This time could be significantly reduced using the newer, faster hardware that is typical of even low-end enterprise servers.

4.1 Controlling Capture Sizes

As previously discussed, one of the primary reasons that network traffic capture and analysis is not used more frequently or as a security tool is because of the sheer amount of data that is generated. While a large portion of this data will be discarded using the capture filter above, it is still important to understand exactly how much storage space will be required both from a capture file and database perspective. To achieve this, before beginning a traffic capture on our entire test network, we isolated single machines running unaltered, fresh installs of varying operating systems. This allowed us to see how much traffic they would generate in a twenty-four hour period. That data can be seen in Table 2 below.

Table 2: Capture sizes of popular operating systems

Operating System	File size	Packet count
Ubuntu 14.04.2 LTS	123KB	1,178
Windows 7 Professional	84KB	193
Windows Server 2008 R2	258KB	1,427
Windows XP	118KB	1,274

The URLs generated by each of these captures can be found in appendix B. This data is useful because not only does it provide a good idea of what URLs can be filtered out or discarded as harmless when querying the traffic capture database, but it also shows just how little data is generated by a machine at rest. One thing to note is that these capture sizes do not include address resolution protocol (ARP) traffic. Based on the network topology and configuration – the DHCP lease time, specifically – capturing ARP traffic may or may not be necessary. This traffic can be useful in associating a machine with its unique media access control (MAC) address for accurate identifications, though ARP is a chatty protocol that frequently uses broadcasts and, as a result, can increase capture sizes significantly based on the number of machines on a subnet or VLAN. The queries used later in this paper will alternate between including and excluding the MAC addresses of devices.

Capturing traffic on the test network previously discussed resulted in a capture size of approximately 2.5MB per minute during the business hours of 8AM to 5PM, but this dropped to just over 400KB per minute during non-business hours. These averages come to approximately 1.35GB of capture traffic during business hours and 400MB during non-business hours for a 24-hour total of just under 1.75GB – a moderate amount of data by any standard, though this will increase when imported into a database, as discussed next. These averages were taken using two full weeks of capture data on our test network.

5 DATABASE STRUCTURE AND SIZE

Using C5 SIGMA to import capture files into a database results in the creation of several tables, which can vary in both size and number depending on how much and what type of network traffic is included in the capture files. For example, if ARP traffic is excluded from the traffic capture, there will not be an ARP table in the database – the tables are dynamically created based on traffic type. In the queries presented in this paper, we are most interested in the following tables: ip, arp, dns_answer_namespec, tcp, udp, icmp, http, http_httpdata, and http__value, as well as two manually created tables called locations and domains for IP geolocation information and blacklist data, respectively. To view how the database is generally built, the structures of the ip, tcp, and dns_answer_namespec tables can be seen in appendices C through E, respectively.

When importing captures files into a database with C5, the size of the database is initially approximately 16 times larger than the captures files themselves. Using the 1.75GB example from the previous section, this would result in a database size of 28GB, which is administratively unmanageable for a single day's worth of traffic captures. However, when using the database solely for security purposes, approximately seventy percent of the information stored in the largest tables can be dropped, resulting in a database of just over 8GB, which is far more usable. Using data from our test network, Table 3 displays the top 13 largest tables and their size percentage of the entire database.

Table 3. Top largest tables

Table	% of total DB size
ip	18.26
geninfo	12.36
eth_dst	11.11
eth_src	11.11
tcp	9.16
tcp_flags	8.04
ip_dsfield	6.27
ip_flags	4.02
eth	2.52
tcp_analysis	2.31
ip_checksum	2.23
ssl_handshake_ciphersuites	2.13
tcp_checksum	2.06

The ip and geninfo table make up more than 30% of the entire database. The geninfo table contains information on the name and location of the capture files and, because this information is of no value in a security context, can be completely dropped. From the ip table, the only relevant security information are the unique source and destination addresses. In our test network, this information made up less than 1% of the total addresses, so the majority of that table can be dropped. Using the capture filters described earlier in the paper, the eth_dst table will contain information of virtually no security value, so it can be completely dropped. The eth_src table is only necessary if the DHCP lease time on the network where traffic is captured is short and MAC addresses are needed to accurately identify machines, versus simply using their IP addresses. Similar to the geninfo table, the tcp_flags, ip_dsfield, and ip_flags, tcp_analysis, ip_checksum, ssl_handshake_ciphersuites, and tcp_checksum tables all contain information of no security value in the deployment described in this paper and can therefore be dropped. Once each of these tables is either cleaned up or dropped altogether, the database size is far more manageable. This step combined with the capture filter described earlier are essential in ensuring that the amount and type of data gathered from network traffic captures remains useful and manageable.

6 NETWORK BASELINE AND QUICK CHECKS

With all of the initial setup previously described in place, network traffic can be captured and imported into a database in real time while keeping both the capture file size and the database itself administratively manageable. Before this database is of any security value, however, it must be queried and analyzed. Many of the tables created by C5 – and the ip table especially – can quickly grow to several million rows. Sorting through the noise and getting to the security-relevant traffic can be done quickly and efficiently through standard SQL queries, many of which are described in the sections to follow.

6.1 ARP to IP and Baseline URLs

In the event that the network topology and DHCP lease time require MAC addresses to be captured in addition to IP addresses, an important first step before querying against the database is to put the IP to MAC bindings in a new table. This new table reduces the number of joins that must be executed in many queries and ensures that two of the largest database tables – the ip and eth_src tables – are not repeatedly searched. The query for creating this new table is below.

```
select ip.ip_src_host, ip.ip_dst, arp.arp_src_hw_mac
into arp_to_ip
from ip
join arp on ip.ip_src_host = arp.arp_src_proto_ipv4
where ip.ip_src_host is not null
group by ip.ip_src_host, ip.ip_dst, arp_src_hw_mac
order by ip.ip_src_host, ip.ip_dst;
```

This creates a new table called arp_to_ip which is both much smaller and much faster than querying both the ip and eth_src tables repeatedly to associate MAC addresses with source IP addresses. The arp_to_ip table created, the next basic query to check for any information specific to a network that can be discarded as harmless is below.

```
select distinct arp_to_ip.arp_src_hw_mac,
arp_to_ip.ip_src_host, arp_to_ip.ip_dst,
dns_answers_namespec.dns_resp_name
from arp_to_ip
join dns_answers_namespec on arp_to_ip.ip_dst =
dns_answers_namespec.dns_resp_addr
where dns_resp_name not like '%<domain>%';
```

This query will search the newly created arp_to_ip table in addition to one of the DNS tables that C5 creates, called dns_answers_namespec, returning all DNS responses on a per-machine basis. The where dns_resp_name not like '%<domain>%' line at the end will vary between networks and filters out the majority of internal-only traffic. This query will initially come back with several results, though many of the domains included can be identified as non-malicious – Facebook, Google, Evernote, Microsoft, etc. – and the query updated as a result.

6.2 Connection Count Check

With the database appropriately setup and a good baseline of public Internet traffic to build off of using the queries above, we can now move forward with more malware- and security-focused queries. Malware itself varies significantly in both how it infects a machine and what it does to the machine once it is infected. It is not uncommon for malware to turn an Internet-connected computer into a bot, which is a machine that reports back to a botmaster or command-and-control server somewhere on the public Internet. If a piece of malware is effective enough, it can spread to several Internet-connected machines all reporting to the same botmaster, creating a bot army or botnet. [17] One of the more popular tasks these botnets are used for is a distributed denial-of-service (DDoS) attack, which occur when the botmaster instructs the botnets to attack a single site or small group of sites on the Internet in an effort to bring the target site offline. During these attacks, each bot within the botnet will repeatedly attempt to access a certain portion of the target site, using as much Internet bandwidth as it has available. From a network capture perspective, this type of attack is very noisy. The attack can be quickly identified by analyzing the network capture database for the number of external connections every machine has made, accomplished using the query below.

```
select ip.ip_src, count(ip.ip_src) as 'conn_count'
from ip
group by ip.ip_src
order by conn_count desc;
```

This query will count each of the connections made by every machine on the network where traffic is captured and order them from greatest to least. Certain machines – a DNS server, for instance – are expected to have a high connection count and so this does not necessarily indicate the presence of malware. However, any client machine that has a connection count several times higher than the rest of the client machines on the network has likely been infected. On occasion, in an effort to avoid detection, certain types of malware will not run during business hours. If this is the case, the query can be updated to what is below.

```
select ip.ip_src, count(ip.ip_src) as 'conn_count'
from ip
where (cast(_timestamp as time) >= '18:00'
and cast(_timestamp as time) <= '23:59')
or (cast(_timestamp as time) <= '07:00'
and cast(_timestamp as time) >= '00:00')
group by ip.ip_src
order by conn_count desc;
```

This is the same as the original query for checking connection count, but casts the _timestamp portion of the database table that C5 creates as an actual time, allowing traffic to be flitered by business hours. What will be considered an unusually high connection count will vary from network to network, but having the connection count of all machines available at once makes relative comparisons possible and infected machines will stand out.

6.3 Non-Standard Port and UDP Detection

Much like a high connection count on a machine, non-standard TCP or UDP port use could indicate the presence of malware or, potentially worse still, the use of peer-to-peer (P2P) software. P2P – also commonly known as torrenting – software allows users to download bits and pieces of large files from several different other machines on the Internet and is well-known within enterprise networks to be both bandwidth intensive and a good way for users to unknowingly download malicious software. While many modern P2P clients can, with the correct configuration, operate on top of standard, well-known ports such as HTTP port 80, [18] in their default configuration this is not often the case. Using the query below, we can check our traffic capture database for non-standard port use.

```
select distinct ip.ip_src, ip.ip_dst, tcp.tcp_dstport
```

```
from ip
join tcp on ip._timestamp = tcp._timestamp
where tcp_dstport != '80'
and tcp_dstport != '443'
and tcp_dstport != '53'
and tcp_dstport != '3389'
```

This will generate a list of the unique TCP destination ports of all machines on the network where traffic is captured. The query above excludes four well-known ports – HTTP, HTTPS, DNS, and RDP, in that order – but can be updated on a per-network basis to exclude uncommon ports that custom developed internal applications may use.

Many P2P clients also rely heavily on the use of UDP, which is a type of network traffic that will stand out in a capture database. As a result, getting the UDP connection count of all machines on the network is a good indicator of machines that are possibly using P2P software and can be accomplished using the query below.

```
select arp_to_ip.arp_src_hw_mac, ip.ip_src,
count(udp.udp_dstport) as 'conn_count'
from udp
join ip on ip._timestamp = udp._timestamp
join arp_to_ip on arp_to_ip.ip_src_host = ip.ip_src_host
group by arp_to_ip.arp_src_hw_mac, ip.ip_src
order by conn_count desc;
```

Much like the IP connection count query described previously, this query will list the number of UDP connections made by every machine on the network in order from greatest to least. One thing to note from the results of this query is that another common protocol that uses UDP is DNS, so it is very likely that a DNS server will be at or near the top of the list. This can be safely ignored as non-malicious and the query updated to exclude the IP in the future.

6.4 Network Scans and ICMP

Getting away from checking for malware for a section, another potential security vulnerability in an enterprise network is port scans. Port scans are a common method of probing a network for open services on servers or desktop machines in an attempt to find vulnerabilities. While there are many different types of software available for port scans, one of the most common protocols they all use is ICMP. Using the capture filter listed earlier in this paper, ICMP traffic is not captured, but this can be remedied simply by including icmp in parentheses alongside port 53. Once ICMP traffic is stored in the capture database, any machine that scans for open ports can be identified using the query below.

```
select count(ip.ip_dst) as 'icmp_count', ip.ip_src
from ip
join icmp on icmp._timestamp = ip._timestamp
group by ip.ip_src
having count(ip.ip_dst) > 100
order by icmp_count desc;
```

One thing to note about port scans is that they can be either high- or low-impact, sometimes called “stealth,” scans. [19] High-impact port scans will send out as many ICMP probes as the machine can handle in an effort to get data back as quickly as possible. These types of scans are the easiest to detect because of how noisy they are. Low-impact scans send out probes at a much slower rate, in an effort to avoid detection. The above query will immediately detect a high-impact scan, but for a low impact scan the 100 in the query would need to be significantly reduced, depending on the date range of data being queried. If the query above comes back with an IP address, the following query will reveal what ports the machine was scanning to find open.

```
select distinct ip.ip_src, ip.ip_dst, tcp.tcp_dstport
from ip
join tcp on ip._timestamp = tcp._timestamp
where ip.ip_src = ‘<IP from previous query>’
order by ip.ip_dst;
```

This query will list each of the destination IPs that the machine scanning the network probed as well as the TCP ports that were scanned. It is important to note that Internet-facing interfaces on servers are constantly being probed from various locations, but because our traffic capture machine is deployed internally, these scans will not be included. External probes are a common security issue that is typically dealt with from a firewall, but internal probes are more dangerous and difficult to detect. If the machine probing a network is on the same subnet or VLAN as the machines being probed, it is going to be difficult to detect port scans using any security method other than the traffic capture database described in this paper.

6.5 HTTP Checks

Virtually all enterprise networks use HTTP and HTTPS both internally and externally for web traffic. As a result, firewalls commonly allow outbound traffic on ports 80 and 443 with few restrictions. A popular method for ensuring that this web traffic is securely managed is to send it through a proxy server. A proxy server is a machine that network traffic filters through on its way to the intended destination and the proxy server chooses to either permit or deny the traffic, based on predefined rules or lists. While a proxy server can be used for several different protocols, it is most commonly used to filter web traffic. When deploying a traffic capture machine, where and how web traffic traverses the network is an important consideration.

Ensuring that web traffic is being accurately and efficiently captured, one way of detecting malware is by viewing the HTTP user agent. The user agent of an HTTP session can be manually coded by a piece of software, so while it is far from an iron clad malware detection method, it is still not uncommon for poorly coded malware to use a user agent that stands out. The query below will check and count all unique user agents in the traffic capture database.

```
select http_user_agent, count(http_user_agent) as 'count'
from http
where http_user_agent is not null
group by http_user_agent
order by count desc;
```

A cursory glance at the data generated by the list will generally reveal that two or three unique user agents are far more popular than others, though this will vary by network. Many user agents can be dismissed offhand as non-malicious, though if the list contains one or two either with a low number of occurrences or an unusual name, this could indicate the presence of malware. This query can be expanded to compare the list generated from the traffic capture database to an external user agent blacklist –

similar to the queries presented in the next section – though the above is useful for getting a baseline of different user agents and how often they appear.

Another security-focused HTTP query is checking the traffic capture database for the presence of the HTTP POST command. HTTP POST is used both in form submission and for file uploads. The query below will list all HTTP POST commands from the traffic capture database.

```
select cast(http_httpdata._timestamp as date) as date,
cast(http_httpdata._timestamp as time) as time, ip.ip_src,
http.http_host, MAX(http.http_request_full_uri) as
'http_full_uri', http__value._value,
http_httpdata.http_request_method
from dbo.http_httpdata
join http on http._timestamp = http_httpdata._timestamp
join ip on ip._timestamp = http_httpdata._timestamp
join http__value on http__value._timestamp =
http_httpdata._timestamp
where
http_request_method = 'POST'
group by http_httpdata._timestamp, ip.ip_src,
http.http_host, http__value._value,
http_httpdata.http_request_method
order by ip_src, date;
```

Because POST is also used for form submission, there will be several lines in the results that are not necessarily file uploads. However, these generally come from non-malicious domains and so can be excluded by updating the where portion of the query to include http_host not like '%<domain>', which will vary from network to network.

7 USING EXTERNAL BLACKLISTS AND GEOLOCATION DATA

Each of the queries described in the previous section are useful in getting a baseline of internal network traffic and spotting outliers using data relative to machines on the network. Another benefit of storing network traffic data in a relational database is that it becomes a simple task to use IP and URL location and blacklists that are available online and compare stored network traffic against these lists. While there are several free and paid sites available [20], as a proof-of-concept, the list from http://malware-domains.com/files is what is used in this paper. This list is distributed as a flat text file, which we imported into our database as a table called "domains" using SQL Management Studio's Import feature. Many of the sites distributing these lists use Really Simple Syndication (RSS) to push out updated versions, which makes automating the download and import of the most up-to-date list in SQL Server a trivial task.

Once the blacklist has been imported into a database table, captured traffic stored in the database can be compared against it using the query below.

```
select distinct arp_to_ip.arp_src_hw_mac,
arp_to_ip.ip_src_host, arp_to_ip.ip_dst,
dns_answers_namespec.dns_resp_name
from arp_to_ip
join dns_answers_namespec on arp_to_ip.ip_dst =
dns_answers_namespec.dns_resp_addr
join domains on domains.url =
dns_answers_namespec.dns_resp_name
where dns_resp_name not like '%<domain>%';
```

If this query returns any results, it immediately raises a red flag. Because each of the rows in our domain table contains a known-malicious URL, any machine sending or receiving traffic from that URL is likely infected with malware. For near real-time blacklist detection, an SQL job can be setup to run every X number of minutes with the following query.

```
IF EXISTS (
<above query>
)
EXEC msdb.dbo.sp_send_dbmail
@profile_name = 'Blackist Item Detected',
@recipients = 'email@domain.com',
@subject = 'Blacklist Item Detected',
@query = '<above query>',
@attach_query_result_as_file = 0;
```

This same query can be used to setup a daily or weekly report with each of the queries presented in this paper. However, while the other queries are useful for gathering baseline data and looking for possible anomalies, blacklist detection virtually guarantees an infected machine should be checked more frequently.

Similar to comparing URLs against a blacklist, IPs stored in the traffic capture database can be queried against IP geolocation information to gather information on where, geographically, network traffic is flowing. The geolocation database we use in this paper is available from http://dev.maxmind.com/geoip [21], but there are several, more accurate lists available for a nominal fee. We used a table called "locations" in our database to store the information. Importing the geolocation information can be done as either a flat file or using an automated process with an RSS feed, just like with the blacklist data.

One issue we ran into when comparing our traffic capture database against the freely available IP geolocation databases is that the geolocation databases all use IP ranges instead of several individual IP addresses. Because IP addresses cannot simply be compared as number, we had to come up with a way to accurately and efficiently compare a single IP to see if it was within a range of IPs. Fortunately, stackoverflow user RBarryYoung [22] had already come up with the solution, as seen below.

```
CREATE FUNCTION dbo.fnBinaryIPv4(@ip
VARCHAR(15)) RETURNS BINARY(4)
AS
BEGIN
   DECLARE @bin AS BINARY(4)
   SELECT @bin = CAST( CAST( PARSENAME( @ip, 4 )
AS INTEGER) AS BINARY(1))
         + CAST( CAST( PARSENAME( @ip, 3 ) AS
INTEGER) AS BINARY(1))
         + CAST( CAST( PARSENAME( @ip, 2 ) AS
INTEGER) AS BINARY(1))
         + CAST( CAST( PARSENAME( @ip, 1 ) AS
INTEGER) AS BINARY(1))
   RETURN @bin
END
go
```

This creates a function called fnBinaryIPv4, which will convert the IP addresses stored in our traffic capture database to binary after breaking them into four parts. Converting the IPs stored in the geolocation database to this same format will allow the two tables to be accurately compared, which can be accomplished using the query below.

```
select distinct ip.ip_src, ip.ip_dst, locations.country,
dns_answers_namespec.dns_resp_name
from ip
join locations on dbo.fnBinaryIPv4(ip.ip_dst) between
convert(varbinary, locations.startip) and
convert(varbinary, locations.endip)
left join dns_answers_namespec on ip.ip_dst =
dns_answers_namespec.dns_resp_addr
where locations.country != 'United States'
order by ip.ip_src;
```

This query will compare all public IPs in our database against the IP geolocation table and return back any IP and, if available, DNS name that is not in the United States. Malware very often originates from and phones home to countries outside of the United States – China, Russia, and the UK, for instance – so if a domestic-only company sees an abnormal amount of network traffic traveling overseas, this could be indicative of an infected machine.

8 MALWARE'S IMPACT ON NETWORK TRAFFIC

To test our real-time network traffic analysis deployment, we installed various types of popular malware on our Windows 7 virtual machine. The virtual machine was running on an isolated network with only one other machine, which was a separate Windows 7 virtual machine running several popular services – HTTP and DNS, most notably – to give the appearance of an active Internet connection to the malware-infected machine. With Wireshark running on the malware-infected machine, we were able to capture all relevant network traffic and then import that file into our traffic capture database. Because these traffic captures were taken simultaneously, this approach allowed us to merge all network traffic from our malware-infected machine with traffic from our test network.

The most notable observation from our tests is that outside of botnet machines participating in a DDoS attack, very rarely did any malware generate an abnormally large amount of network traffic. Unusual port use was common, with the Neutrino Exploit Kit (EK) using port 8000 to both send and receive traffic. Other types of malware used varying port numbers as well, each of which stood out in our capture database. Another common theme among many of the different pieces of malware we tested was that they would check for an active Internet connection before doing anything else. One of the most common URLs that would appear during this step was http://checkip.dyndns.com. While not a malicious site in and of itself, this URL could be added to the blacklist table in an effort to spot potential malware checking for Internet connectivity. In our test network, it was only malware that attempted to access this site.

Overall, we were surprised to find that as far as a percentage of overall network traffic volume, malware-infected machines do not stand out – finding them becomes similar to searching for a needle in a haystack. However, with an accurate network baseline gathered from the queries presented in this paper, it is possible to dismiss a large majority of harmless traffic all at once, leaving only a small amount of unique and malicious traffic to analyze.

9 CONCLUSIONS, LIMITATIONS, AND FUTURE WORK

As more and more personal data is converted from a paper to a digital format and transferred across the Internet, the attack surface and potential return on investment (ROI) for cyber criminals significantly increases. Large-scale network and data breaches are becoming commonplace and they cost corporations millions of dollars. Malware that sits on supposedly secure networks for months on end and transfers private data to an external location is a common culprit and the deployment ideas and examples presented in this paper are an attempt to detect and mitigate this type of attack. Information security should always adopt a multilayer approach and the preventative techniques that enterprises typically rely exclusively on offer little protection for detecting or mitigating attacks after they have breached the network perimeter. We attempted to design and deploy a solution that was both cost-effective and end-user transparent, increasing security without increasing costs or impacting network or machine performance. In light of these goals, our deployment proved a success. Port mirroring did not significantly affect network or switch performance and capturing and storing traffic in a database can be done using only no-cost software. With the appropriate capture filters and database queries to drop unnecessary or redundant tables and data, captured traffic can be kept at a manageable size.

Limitations to the ideas presented in this paper include the assumption that enterprise networks are already operating with all of the prerequisites discussed in this deployment. Without managed switches capable of port mirroring already in place, additional hardware would need to be purchased. Additionally, a network topology that lends itself well to breaking network traffic into manageable subnets is required. These are limitations that can be worked around by limiting the scope of the deployment – monitor only the highest priority servers, for instance – or by replacing unmanaged switches with their managed counterparts, which brings a wealth of additional network performance and security features.

Future work possibilities for this project are virtually endless. It is a safe assumption that cyber-attacks will never stop, so different queries can be developed to detect new malicious traffic and protocols. A web interface could be developed to allow the queries to be run on demand, while charting historical data and automatically detecting anomalies. The deployment presented here is intended simply as a proof-of-concept that network traffic captures do not have to be relegated to spot-checking performance or connectivity issue and can have enormous security value if properly handled.

REFERENCES

[1]. Michael Riley, Ben Elgin, Dune Lawrence, and Carol Matlack, "Missed Alarms and 40 Million Stolen Credit Card Numbers: How Target Blew It," March 2014. Available: http://www.bloomberg.com/bw/articles/2014-03-13/target-missed-alarms-in-epic-hack-of-credit-card-data

[2]. Nagios. (2015). Retrieved Feb. 10, 2017 from http://www.nagios.org/

[3]. Cacti. The complete rrdtool-based graphing solution. (2012). Retrieved Feb. 10, 2017 from http://www.cacti.net/

[4]. Zabbix. The Enterprise-class Monitoring Solution for Everyone. (2015). Retrieved Feb. 10, 2017 from http://www.zabbix.com/

[5]. Spiceworks: Where IT goes to work. (2015). Retrieved Feb. 10, 2017 from http://www.spiceworks.com/

[6]. Solarwinds. The Power to Manage IT. (2015). Retrieved Feb. 10, 2017 from http://www.solarwinds.com/

[7]. Paessler. The network monitoring company. (2015). Retrieved Feb. 10, 2017 from http://www.paessler.com/prtg

[8]. Observium. Network monitoring with intuition. (2015). Retrieved Feb. 10, 2017 from http://www.observium.org/

[9]. Christian Rossow, Christian J. Dietrich, Herbert Bos, Lorenzo Cavallaro, Maarten van Steen, Felix C. Freiling, and Norbert Pohlmann. 2011. Sandnet: network traffic analysis of malicious software. In Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS '11). ACM, New York, NY, USA, 78-88.

[10]. Xiaohong Yuan, Percy Vega, Jinsheng Xu, Huiming Yu, and Yaohang Li. 2007. Using packet sniffer simulator in the class: experience and evaluation. In Proceedings of the 45th annual southeast regional conference (ACM-SE 45). ACM, New York, NY, USA, 116-121.

[11]. Alexandros Fragkiadakis, Ioannis Askoxylakis, "Malicious traffic analysis in wireless sensor networks using advanced signal processing techniques," 2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), pp. 1-6

[12]. Liran Ma; Teymorian, A.Y.; Xiuzhen Cheng, "Passive Listening and Intrusion Management in Commodity Wi-Fi Networks," Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE, pp.327-331, 26-30 Nov. 2007

[13]. Vasil Y. Hnatyshin and Andrea F. Lobo. 2008. Undergraduate data communications and networking projects using opnet and wireshark software. SIGCSE Bull. 40, 1 (March 2008), 241-245.

[14]. Wireshark. Go Deep. (2015). Retrieved Feb. 10, 2017 from https://www.wireshark.org/about.html

[15]. Xin Wu, Daniel Turner, Chao-Chih Chen, David A. Maltz, Xiaowei Yang, Lihua Yuan, and Ming Zhang. 2012. NetPilot: automating datacenter network failure mitigation. SIGCOMM Comput. Commun. Rev. 42, 4 (August 2012), 419-430.

[16]. Command Five. (2015). Retrieved Feb. 10, 2017 from https://www.commandfive.com/downloads/c5sigma.html

[17]. Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. 2006. A multifaceted approach to understanding the botnet phenomenon. In Proceedings of the 6th ACM SIGCOMM conference on Internet measurement (IMC '06). ACM, New York, NY, USA, 41-52.

[18]. Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and Kc claffy. 2004. Transport layer identification of P2P traffic. In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement (IMC '04). ACM, New York, NY, USA, 121-134.

[19]. Lawrence Teo. 2000. Port Scans and Ping Sweeps Explained. Linux J. 2000, 80es, Article 2 (November 2000)

[20]. Malware Domain Blocklist. (2008). Retrieved Feb. 10, 2017 from http://www.malwaredomains.com/

[21]. GeoLite Legacy Downloadable Databases. (2015). Retrieved Feb. 10, 2017 from http://dev.maxmind.com/geoip/legacy/geolite/

[22]. Datatype for Storing IP Address in SQL Server. (September 2009). Retrieved from http://stackoverflow.com/questions/1385552/datatype-for-storing-ip-address-in-sql-server