**Lab 10: Network Security Monitoring and Security Onion**

This lab uses the following VMs:

* Security Onion

As usual, I have a ready-to-go Security Onion VM for you that you can download [here](https://www.dropbox.com/s/7nph5laki7e0o80/security-onion.ova?dl=0).

The “username:password” for the image I provide is “securityonion:Password1”.

# Part 1: Analyzing Netflow information

Although full-content data are powerful, they are less useful for fast querying and timely incident response. This is where Netflow records, some of the most powerful information sources available to incident responders, become very useful. These records are brief summaries of network traffic which can be maintained indefinitely due to their small file size. They provide a running history of network connections at the time of an incident. Read the below article and use Google to answer the following questions:

[https://en.wikipedia.org/wiki/Netflow](https://en.wikipedia.org/wiki/NetFlow)

**Question:**What types of information do Netflow records contain?

**Question:**What are at least three benefits of Netflow over full PCAP files?

Now that you have basic understandings of Netflow, we will use YAF and SiLK (open-source incident response tools for Linux) to analyze Netflow data. Further documentation can be found at http://www.appliednsm.com/silk-on-security-onion/.

We will analyze sample data hosted [here](https://tools.netsa.cert.org/silk/referencedata.html), which says the following about the data source:

This sample data is derived from anonymized enterprise packet header traces obtained from Lawrence Berkeley National Laboratory and ICSI, and is used here with their permission. This data covers selected hours on selected dates in late 2004 and early 2005.

1. First, you should run the following commands for the start and end dates to filter on:

export sd="--start-date=2004/10/04:20"

export sd="$sd --end-date=2005/01/08:05"

This is just a convenience variable that lets you not have to type out the --start-date and --end-date flags for each of the following commands. It is stored in $sd. You can view it by running echo $sd. Go on, it’s just text!

We can now start querying the sample data to find some interesting and important information about the sample network.

All commands that start with rw are part of the SiLK Tool Suite. rwfilter first selects records from Netflow files, and rwstats summarises those records.

1. Query the top “talkers” (those host-pairs that send and receive the most traffic) **by bytes**.

Enter the following on one line, or use the backslash “\” to let you press enter to continue the command on a new line.

rwfilter --data-rootdir=/data/SiLK-LBNL-05 --protocol=0- \

--type=all $sd --pass=stdout \

| rwstats --percentage=1 --fields=sip,dip --bytes

**Important:** Note that the - at the end of --protocol=0- is important. It indicates that all protocols from 0 and above will be included. It is equivalent to --protocol=0-255. If you don’t include the hyphen, no results will be returned. [IP protocols include](https://en.wikipedia.org/wiki/List_of_IP_protocol_numbers), but are not limited to, TCP and ICMP (ping, traceroute)

**Note:** The code following the | (pipe) symbol passes the output of rwfilter to rwstats, which provides fast and powerful statistics. The sip and dip fields stand for source and destination IP respectively. To understand “source” and “destination”, consider the client-server architecture pattern. In the vulnerability scanning lab, we identified services listening on ports on Metasploitable. One such service was an sshd – an SSH server listening on port 22. If we were to connect to this server with an SSH client, say, to log in to a Metasploitable account from kali, and if that login were captured in a Netflow record, then the “source” for that record would be kali, and the “destination” would be Metasploitable.

**Note:** The --percentage=1 flag specifies that we only want to retain a sip,dip pair if total bytes exchanged between the two comprised at least 1% of total network byte traffic. The --fields conceptually performs a “group-by” on the incoming rwfilter data for, in this case, unique sip,dip pairs. Then the --percentage flag filters based on ranked group aggregate values for, in this case, --bytes.

**Question:**Looking at the output, take note of the top 5 talkers source and destination IPs, ranked by the % of bytes the pair generated.

1. Query “top talkers” (those host-pairs that send and receive the most traffic) **by the number of Netflow records they generated**:

rwfilter --data-rootdir=/data/SiLK-LBNL-05 \

--proto=0- --type=all $sd \

--pass=stdout | rwstats --count=25 \

--fields=sip,dip

**Note:** The --count=25 flag specifies that we only want to retain a sip,dip pair if the number of Netflow records associated with traffic exchanged between that pair was in the top 25 of all Netflow-record pair-counts.

**Question:**Take note of the source and destination IPs of the top three talker-pairs, ranked by number of flow records.

**Consider:** Conceptually, why do you think it’s important information to know the top talkers on the network?

1. Query top SSH flows. This is typically done using a destination port (dport) filter for port 22, as follows:

rwfilter --data-rootdir=/data/SiLK-LBNL-05 \

--proto=0- --type=all $sd \

--dport=22 --pass=stdout | rwstats --count=10 \

--fields=sip,dip

**Question:**Take note of the host IP address that has the greatest number of flow records associated with SSH’ing to another specific host.

**Consider:** will the sIP or dIP represent the SSH-connection initiator, if the destination SSH port is 22?

1. Starting from the query in the previous step, do a follow-up analysis on a particular SSH-connection-starter.

**Note:** This question intentionally offers minimal guidance.

* + Write a query using a rwfilter flag to select only records associated with a single IP address (rwfilter --help and browse through the “partitioning switches” section). Filter to only SSH connections initiated by 128.3.161.229.
	+ Pipe that to rwstats, group by unique source IP and destination IP addresses, and examine the pairs with the 10 highest total number of SSH flow records.

**Question:**What are the top SSH destination IPs for 128.3.161.229? In other words, what hosts is this box SSH'ing to the most often?

1. Query for long standing SSH traffic:

rwfilter --data-rootdir=/data/SiLK-LBNL-05 \

--proto=0- --type=all \

--dport=22 --duration=1700- \

$sd --pass=stdout | rwcut

**Note:** rwcut dumps out all rwfilter-piped records – it performs no aggregations like until rwstats.

**Note:** In this example, --duration=1700- and --dport=22 filters to only records with a SSH connection time of at least 1700 seconds (almost 30 minutes).

**Question:**What is the IP address of the host client (source) that had an an ongoing SSH connection/session to an SSH server on another host (destination) of at least 30 minutes?

**Reflect:** Conceptually, why should we look for long standing SSH connections?

There are many other filters that we can use to analyze the network traffic in many situations, especially under incident response circumstances.
To learn more about YAF and SILK, you can find additional material on <https://tools.netsa.cert.org/>

# Part 2: Examining PCAP Files

In this section, you’ll examine the network traffic for a Windows VM that browsed to a compromised website that in turn referred the Windows VM to a server that delivered malware to the Windows VM. You’ll use Squert and Wireshark to investigate these events.

1. Ensure that an IDS signature rule 2000419 is enabled.

The following steps in this lab rely on a snort rule being enabled in Security Onion that will be tripped by a windows EXE being downloaded over a non-standard HTTP port. Downloading executables is a normal part of using operating systems, but perhaps not so much in a corporate environment where employees shouldn’t be downloading executable files onto their machines.

The signature rule we want to enable is [2000419](https://doc.emergingthreats.net/bin/view/Main/2000419). This signature is in a list of rules downloaded by PulledPork, which Security Onion uses to manage IDS rules. Ensure that it is enabled, even if it is default-disabled in downloaded rule sets, by adding this id to a PulledPork config file:

sudo bash

echo 1:2000419 >> /etc/nsm/PulledPork/enablesid.conf

rule-update

Examine the rule at the link above. When this rule is triggered, it will write “ET POLICY PE EXE or DLL Windows file download” or “ET POLICY PE EXE or DLL Windows file download Non-HTTP”, depending on the rule version in use.

1. Navigate to the /data/cases/ directory, where case.pcap is found (available [here](https://daveeargle.com/class/cu/mgmt4250/case.pcap) if you don’t already have it). Run the following command.

sudo tcpreplay -i eth0 -M 10.0 case.pcap

This command replays network traffic stored in the case.pcap file onto security onion’s network card, as if the network activity were happening again, live.

You should see the following result (ignore the error messages for the 20 failed packets):

Statistics for network device: eth0

Attempted packets: 4682

Successful packets: 4662

Failed packets: 20

Retried packets (ENOBUFS): 0

Retried packets (EAGAIN): 0

1. Log in to Squert using the icon on the Security Onion desktop using analyst:analyst for the username:password. (Bypass the SSL warning by clicking “Advanced” then “Proceed to site.” From SquertProject.org:

“Squert is a web application that is used to query and view event data stored in a Sguil database (typically IDS alert data). Squert is a visual tool that attempts to provide additional context to events through the use of metadata, time series representations and weighted and logically grouped result sets. The hope is that these views will prompt questions that otherwise may not have been asked.”

(IDS stands for Intrusion Detection System.)

1. Each row in Squert lists an IDS event. Click on the QUEUE “2” button on the row with ET POLICY PE EXE or DLL Windows file download to see the detail of this alert.



When you click this number, more details will appear below the accordion expansion, including the source and destination IP of the associated IDS event.

This particular record is a response to a HTTP web browser request which downloaded a malicious executable payload. Because it is a response, the source IP represents the attack machine, and the destination IP represents the victim machine. This convention will not always hold for this case analysis-- it depends on whether a query or a response is being examined.

**Question:**What is the IP address of the Windows VM to where the malware payload was sent (the destination IP)?

**Question:**What is the IP address of the host that sent the malware payload? (the source IP)?

You can also click on the Summary tab to see a map and summary of traffic by countries.



If the map isn’t working, just search for the IP address using Wolfram-Alpha. Or, follow the instructions in the box below to update your map.

**Question:**What country does this malware payload comes from?

If you don’t have any country information showing, first make sure that you have an internet connection (try ping google.com),

If you can’t ping google.com, run this:

sudo ifdown eth1 && sudo ifup eth1

Once you can ping google.com, run this:

cd /var/www/so/squert/.scripts

sudo ./ip2c.tcl

Then press Squert’s “Refresh” button (not the browser refresh button):



1. You can pivot from Squert to other network forensics tools for follow-up analyses. From the Events view, drill-down one level deeper by clicking on the second QUEUE “2” button that appeared after clicking on the first. You should now see two events. Click on one of these “event ids.” Doing so will pivot you to another tool called CapME. Log in with username:password analyst:analyst.

After a moment, you will see a representation of the HTTP web request which fetched and downloaded the malware payload. In **red text** is the HTTP request that the browser made for the download, and in **blue text** is the malicious server’s response in which the payload was actually downloaded.

In the RED text you should see a HOST header, as well as a GET header. the HOST header shows the domain name and port that the browser requested. This is the malicious web domain. the GET header shows the specific URL that was requested that delivered the malware payload.

**Note:** If the host were still serving the malware, you could theoretically combine the HOST and the GET values and paste them into your browser, and download the malware anew! But there is no need to do this if it is the malware you seek. You already can extract it because you have a record of the entire malware download network activity. This is the power of NSM -- to know all, see all, recreate all, for anything going to or sent from anyone on your network (assuming it's not encrypted).

**Question:**What is the domain name that served the malicious payload?

1. At the top and on the bottom of the CAPme report, you will see links to download a .pcap file. Do so, then open the download from the browser. This will pivot to Wireshark, another network forensics tool, with a different view of the same event.

The very first row in the Wireshark view shows the packet that the victim sent to the attack machine to begin the request to download the malware payload. So, the “source” is the victim, and the “destination” is the attack machine.

We are interested in knowing the MAC address of the victim machine so that we can do follow-up analysis. Examine this first packet. Expand the “Ethernet II” frame in the packet view, and note the 6-octet-long address delineated by :.

**Question:**What is the MAC address of the victim VM?

1. But what was this host doing that led to them downloading malware? What sent them to that malicious domain? Let’s investigate!

In Wireshark’s File menu, choose “Open,” navigate to /data/cases/case.pcap file, click “Open.” This will load the entire traffic file – not just the traffic directly associated with the malware download.

Note the source IP address for packet number 1. This is the Windows VM that gets infected. This entire network trace only pertains to web-based traffic associated with our victim for a certain time window.

Right-click the first packet, and choose Follow TCP Stream. This will bundle together all of the network packets associated with this single HTTP web request, as did CAPme for the single HTTP event that resulted in the malware download.

**Question:**What HOST did the victim machine make a request to in the initial TCP session?

1. We know from earlier that the malicious HTTP download request included a request for a page located at /cars.php?honda=1185... etc. Let’s find that event, and its corresponding stream, by filtering for it:
2. http contains "cars.php"

This should return one result – the one web request associated with fetching a route called “cars.php”. For this record, note again the IP address of the attack host – in this case, the destination IP address.

For this one record, right-click the destination field value (the IP address), then choose Apply as Filter > Selected. This will filter the entire tracefile to only activity with a destination IP matching this field. Let’s filter even further to only select the records with HTTP protocol. This can be done by appending and http after your ip.dst == filter expression.

You should now see three HTTP requests to this malicious IP. We recognize the second one – the GET /cars.php one. It is the one that delivered the malware. Let’s look at the first one – right-click Follow TCP Stream it. You will notice that this HTTP request as a REFERER header. This is http-speak for the site that redirected the browser to the current one. Note the value for the REFERER. There is a good chance that this is a compromised website. They’re probably all compromised, but hey, world we live in. Fix or blacklist one site at a time.

**Question:**What is the domain name of the “referer” website that referred the Windows VM to the IP that delivered the malware?

We are also interested in knowing the IP address of this referer website – the host at that IP may be hosting other compromised sites, so we may want to blacklist the address. One way that you can find that IP address is by applying the following filter: http contains "Host: name.of.the.referer" (case-sensitive, and do not include the protocol (e.g., http:// ))

**Question:**What was the IP address of this referer website?

1. What exactly was downloaded? Let’s carve it out and search online for a report on what it does.

Follow again the tcp.stream related to the http request for getting cars.php (review above if you have forgotten how). From the popup window, choose “Save As”, and save the stream somewhere.

Note the “MZ” string at the top of the blue response stream. This string is a magic number that identifies the file type that is being downloaded in this request (see <https://asecuritysite.com/forensics/magic>). Note also the This program… message. This is another indicator of the file type.

Next, open a terminal and navigate to the directory where you saved the stream, and use a forensics file carving tool called Foremost:

foremost -i the.name.you.chose -o name.of.the.directory.where.you.want.to.save.the.carved\_files

This will create a directory name.of.the.directory.where.you.want.to.save.the.carved\_files containing all of the files that Foremost carved out of the network stream.

**Foremost stuck on "reading from stdin"?** This happens when foremost cannot find the file you reference on -i. It will never end. stdin means that it is sitting there waiting for you to enter something, because it couldn't find an input otherwise. It’s an odd behavior, but makes sense I guess in some programmer's mind. Navigate to the directory where your input file is located, and then run foremost.

Inside your carved files directory, you will find a subdirectory for each file type recovered. For this analysis, you should see two subdirectories – one for extracted png files, and another for extracted exe files. The .exe in the exe directory is the malware payload.

Use a hashing algorithm such as sha256sum to hash the extracted .exe file.

**Question:**What is the sha256 hash of the exe payload?

Browse to a website such as [virustotal.com](https://daveeargle.com/security-assignments/lab_security_onion/virustotal.com) or [hybrid-analysis.com](https://daveeargle.com/security-assignments/lab_security_onion/hybrid-analysis.com) and search the site using the payload’s hash. This can tell you more about what you’re dealing with in your network and potentially how to clean it up.

**Question:**As shown on virustotal.com, what does Kaspersky antivirus report the exe to likely be?

[Read about](https://krebsonsecurity.com/2014/01/feds-to-charge-alleged-spyeye-trojan-author/) the arrest of, charges against, and plea to conspiracy from Aleksander Panin, the author of the variant of malware with which we are dealing.

**Question:**What does Brian Krebs indicate this malware is typically used for?

**For fun:** Install mirage and use it open the png inside your carved-files directory if you’re curious what it looks like (preface install commands with sudo on this system), Inside that subdirectory, use the file command to identify the file type.

If you ever have a storage device that corrupts and is reported to be unreadable by your operating system, it is possible that you can use a tool like foremost to carve out files from the media.

# Part 3: Operation Aurora Case

## Preparation: Clear Security Onion History

1. Double-click the “Setup” icon on the Security Onion desktop, and enter the password “Password1”.
2. Click “Yes, skip network configuration!”
3. Click OK with the default setting of “evaluation mode.”
4. Click OK with the default setting of “eth0” for the monitoring interface.
5. Enter “analyst” for the Sguil username.
6. Enter “analyst” for the Sguil password, and confirm.
7. Click “Yes, proceed with changes!”
8. Click OK on the remaining dialog messages.

The logs on Security Onion have been reset, giving you a clean slate for the case below.

## Case Scenario

You will use all the skills you’ve learned in this lab so far to solve the following case based on a real hack called Operation Aurora. First, watch [this video](https://en.wikipedia.org/wiki/Operation_Aurora) or read [this Wikipedia article](https://www.youtube.com/watch?v=T2DqLj1nQkc) about Operation Aurora, which was an attack on Google and other companies. Then, read the following scenario:

Claire Young is after GumTiger’s killer app source code. She’s been trailing the lead developer, Alex Stephens, to figure out how she can remotely access GumTiger’s servers. One night, while conducting reconnaissance, she sees him log into his laptop 10.10.10.70 and VPN into GumTiger’s headquarters.

Leveraging her connections with international hacking organizations, Claire obtains a [0-day exploit for Internet Explorer](http://www.symantec.com/connect/blogs/trojanhydraq-incident-analysis-aurora-0-day-exploit) and launches a client-side spear phishing attack against Alex Stephens. Claire carefully crafts an email to Alex containing tips on how to improve the source code and sends it. Seeing an opportunity that could get him that Vice President of Product Development title (and corner office) that he’s been coveting, Alex clicks on the link. Claire is ready to strike…

You are the forensic investigator. Your mission is to analyze [the packet capture](http://forensicscontest.com/contest06/evidence06.pcap) containing Claire’s exploit, build a timeline, and answer the questions below.

**Note: Intentional lack of specific steps ahead.** Apply skills covered in Part 2 to solve this case.

If you are using my class VM, the packet capture evidence file (the .pcap file) is downloaded to your VM and available at: /data/cases/evidence.pcap

1. Find the full URL of Alex Stephens’ original web request, including the port.

**Question:**What was the full URI of Alex Stephens' original web request? (Please include the port in your URI.)

1. In response, the malicious web server sent back obfuscated JavaScript, which contained a zero-day exploit. Near the beginning of this JavaScript code, the attacker created a JavaScript array named “UWnHADOfYHiHDDXj” with 1300 “COMMENT” HTML elements, then set the data property of each of those elements to a string. What was the value of this string? (hint: look for .data a few lines down within the loop.

**Question:**What was the string value of JavaScript variable “UWnHADOfYHiHDDXj”?

**Note**:JavaScript is the programming language that makes webpages interactive. With it, code can dynamically create and manipulate HTML elements on a webpage. While a web server sends basic html in response to a page request, any JavaScript is executed on the client’s browser, not on the server. That is important—any code that can be executed on your own computer can potentially do bad things to your computer! Web browsers such as Internet Explorer are designed to not allow code like JavaScript to do anything outside of the scope of the webpage you are viewing... but this 0-day escapes the jail of the browser.

The Symantec analysis above reports that:

This exploit was used to deliver a malicious payload, known by the name of Trojan.Hydraq, the main purpose of which was to steal information from the compromised computer and report it back to the attackers.

The exploit code makes use of known techniques to exploit a vulnerability that exists in the way Internet Explorer handles a deleted object. The final purpose of the exploit itself is to access an object that was previously deleted, causing the code to reference a memory location over which the attacker has control and in which the attacker dropped his malicious code.

The payload is created with the JavaScript var LLVcUmerhpt, and the memory is manipulated by overwriting those earlier .data properties with a value which be interpreted as a memory address and which will cause the browser to execute the payload from that attacker-chosen location in memory.

1. The loaded webpage included an [IFRAME](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe) element which caused Alex’s computer to make a second HTTP GET request for an object.

**Question:**What was the filename of the second HTTP object that was requested?

**Question:**What is the MD5 hash of the second file requested (the second HTTP response object)?

**Hint:**

You can first filter to `http` traffic with `GET` requests, and note the name and properties of the second one. Then, You can use foremost to extract the http response object from a saved copy of the http tcp stream.

Alternatively and equivalently, you can export HTTP response objects in Wireshark by going to the Wireshark File menu, and selecting “Export Objects” > HTTP. You should see two "HTTP"-requested objects. The first is the initial webpage visited, and the second is a gif, the downloaded of which was triggered by the initial webpage loaded. Note the name of the second file requested. Save the second one to a file. This saved file is the equivalent of what foremost would have carved.

1. This new malicious object opened a TCP session on port 4444 between Alex’ computer and Claire’s machine.

Find the packet for when the TCP session on port 4444 opened, and also find the one when it closed. Use the timestamp difference between these two to determine how long the port was open. This is significant because it tells us how long the attackers had to perform their attack.

**Hint** Use the Wireshark filter “tcp.port == 4444”. For the time, see the value in Wireshark’s “Time” column for the first row in the filtered results. Right-click the first packet in the filtered results, and choose Set Time Reference. Then note the time value for the last row in the filtered results. With the first set as the time reference, the last packet will indicate how long this tcp stream was open.

**Question:**How long was the TCP session on port 4444 open?

1. In packet 17, the malicious server began to send a file to the client over the port 4444 TCP session. What type of file was it?

Hints:

* + Examine the magic number at the beginning of the download data
	+ Extract the downloaded file from the tcp stream associated with packet 17, carve out the files, and use the file command from the terminal.

**Question:**What type of file was it, according to the magic number and to the file command?

**Question:**What was the MD5sum of this file?

1. Search for a hash of the downloaded file on [Virustotal.com](https://daveeargle.com/security-assignments/lab_security_onion/Virustotal.com) and on [hybrid-analysis.com](https://daveeargle.com/security-assignments/lab_security_onion/hybrid-analysis.com).

**Question:**What type of file is this, according to hybrid-analysis and Virustotal?

1. You notice that the victim’s computer repeatedly tried to connect back to the malicious server on port 4445, even after the original connection on port 4444 was closed. Eventually, the malicious server responded and opened a new connection on port 4445. Subsequently, the malicious server sent a second executable file to the client on port 4445.

**Question:**What was the MD5 sum of this second, new executable file downloaded over the port 4445 TCP session?

**Hint:**Search for traffic with this new port. You will see a lot that are red and black -- these are failed connection attempts. The black is the victim sending TCP SYN packets to the attackbox port 4445, and the RST/ACK response is the attack box indicating that the [port is closed](https://stackoverflow.com/questions/5272026/tcp-server-sends-rst-ack-immediately-after-receiving-syn-from-client).

Eventually, the attackbox responds with SYN/ACK -- port is open and ready for business, establish the TCP session! The remainder of the TCP session is spent downloading the executable.

You can be square and just scroll around until the colors change, or you can use the power of Wireshark filters. Try this one:

tcp.port == 4445 and tcp.flags.syn == 1 and tcp.flags.ack == 1

This will give you the start of the TCP stream that downloaded the second file.

# Sources

* <http://malware-traffic-analysis.net>
* <http://forensicscontest.com/2010/05/21/puzzle-6-anns-aurora>
* McManus, Joe. Using Net Flow Data for Incident Response. Nov 2014. University of Colorado Boulder

**Note**: This lab was created together with Dr. David Eargle at the University of Colorado Boulder.