corelight

THREAT HUNTING GUIDE

How to threat hunt with Open NDR + MITRE ATT&CK®

This Threat Hunting Guide was created to teach you simple and relevant ways to discover attacks before they happen using Corelight network data. This document — organized around the MITRE ATT&CK® framework — is designed to help you develop a theory for threat hunting and establish prioritization.

MITRE ATT&CK is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. It's used as a foundation for specific threat models and methodologies in the private sector, government, and the cybersecurity industry. With the creation of ATT&CK, MITRE is fulfilling its mission to solve problems for a safer world — by bringing communities together to develop more effective cybersecurity. ATT&CK is open and available to any person or organization for use at no charge. [1]

WHAT IS THREAT HUNTING?

At a high level, threat hunting is actively looking for adversaries in your network when you don't know if they're inside. This is different from indicator matching, which is only watching for well-known signs of attackers, for example, IP address(es) or file hash. Usually conducting a threat hunt involves researching a theory, or hunch, and then analyzing data looking for something interesting. Items that are interesting can take many shapes, for example in The Cuckoo's Egg, by Clifford Stoll an accounting error initiated the hunt.

"Dave wandered into my office, mumbling about a hiccup in the Unix accounting system. Someone must have used a few seconds of computing time without paying for it. The computer's books didn't quite balance; last month's bills of $2,387 showed a 75-cent shortfall."

This 75-cent difference was the indicator that led to the discovery of multiple corporations and government systems that were compromised. The term "interesting" is used throughout this guide and it is only limited by your imagination.

WHY CONDUCT A THREAT HUNT?

Most host-or network-based detection systems rely on matching, otherwise known as signatures, to generate alerts to signal defenders that there is something unwanted in the network. However, attackers are continually evolving to evade detection, and signatures are developed only after the artifact was discovered in another network. So, if you're not hunting for artifacts in your environment, how will you discover that attackers are evading your current defenses?

Hunting has several positive outcomes. The first is you might find artifacts of an active intruder that your current defenses missed. While some may think this is a tragedy, it can be a huge win, especially if the intruder hasn't completed their objective(s). In every hunt, there's always something to find.

You may discover network or software misconfigurations that pose a threat, either because they degrade network performance or introduce a vulnerability. Next, the hunt could yield run-of-the-mill infections such as adware, or other dormant malware that aren't directly targeting your organization but are still a threat. Lastly, resource abuse and Shadow IT, services that are not officially supported, can introduce risk through degraded network performance or new adversary attack vectors. Every hunt teaches you something new about the network which will aid in your next investigation.

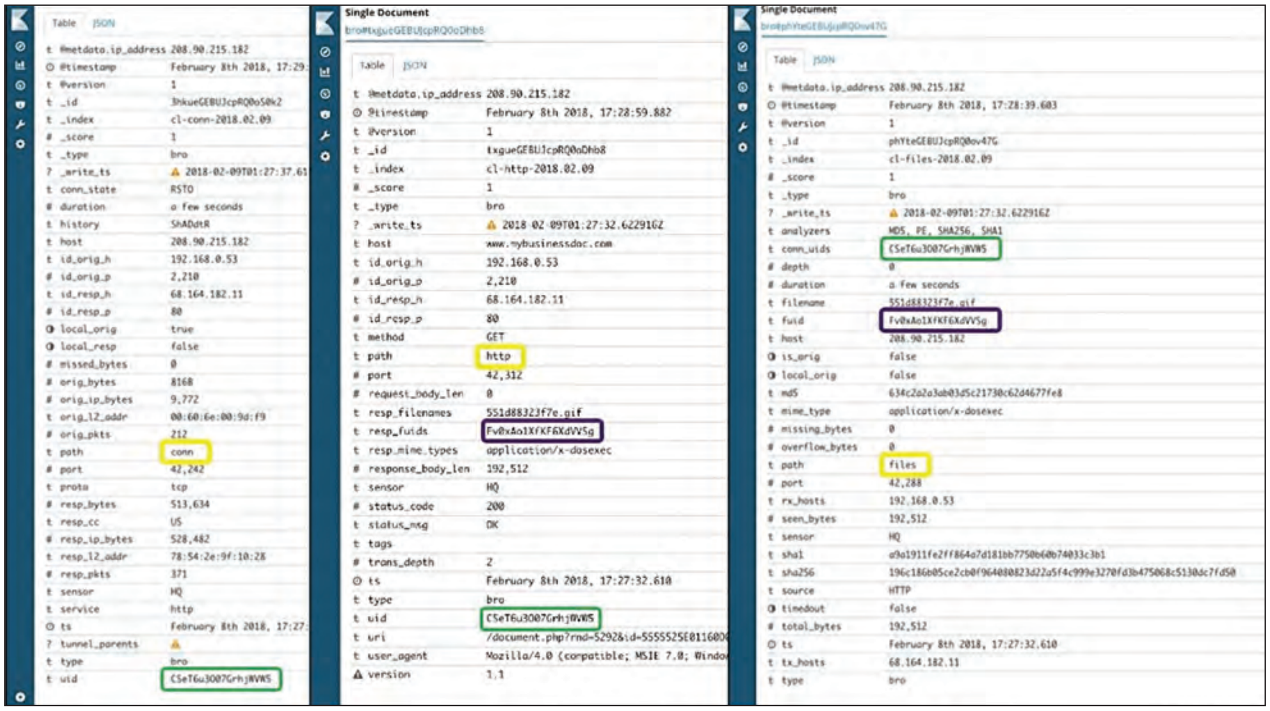
WHY HUNT WITH NETWORK DATA? PACKETS. DON'T. LIE.

It's really as simple as that. If a network-resident intruder is active in your network, there will be network artifacts. In artifacts, there are clues to what is happening, or better yet, an exact moment-for-moment story of what happened. For example, if a command and control channel uses DNS as a transport mechanism, there will be DNS queries and replies. Additionally, the IP address(es) that are on the ends of a TCP connection must be accurate, they cannot be spoofed if data is exchanged. All attacks traverse the network, unless they are isolated to one host, so there will be packets.

CORELIGHT LOGS NOMENCLATURE

Corelight provides data-centric solutions that analyze network traffic and enhance automation tools by transforming network traffic into linked logs and extracting files. The central log is the conn log, which documents general information about all network sessions.

The conn log records information about each network endpoint and the service (application) and also assigns a uid (unique identifier). The uid links the conn log to related protocol logs, where specific session information is available. For example, the conn log can list http as the service, and using the uid you can pivot to the http log to get specific protocol information about the session. The uid separates Corelight solutions from other security tools. This field links otherwise disparate information into easily digestible logs. The uid is fundamental to conducting link analysis and a critically important field that facilities pivoting, or joining multiple logs together.



```
Table   JSON
t @metdata.ip_address 208.90.215.182
⊙ @timestamp          February 8th 2018, 17:29:
t @version            1
t _id                 3hkueGEBUJcpRQ0oS0k2
t _index              cl-conn-2018.02.09
# _score              1
t _type               bro
? _write_ts           ⚠ 2018-02-09T01:27:37.61
t conn_state          RSTO
# duration            a few seconds
t history             ShADdtR
t host                208.90.215.182
t id_orig_h           192.168.0.53
# id_orig_p           2,210
t id_resp_h           68.164.182.11
# id_resp_p           80
◑ local_orig          true
◑ local_resp          false
# missed_bytes        0
# orig_bytes          8168
# orig_ip_bytes       9,772
t orig_l2_addr        00:60:6e:00:9d:f9
# orig_pkts           212
t path                conn
# port                42,242
t proto               tcp
# resp_bytes          513,634
t resp_cc             US
# resp_ip_bytes       528,482
t resp_l2_addr        78:54:2e:9f:10:28
# resp_pkts           371
t sensor              HQ
t service             http
⊙ ts                  February 8th 2018, 17:27:
? tunnel_parents      ⚠
t type                bro
t uid                 CSeT6u3O07GrhjWVW5
```

```
Single Document
bro#txgueGEBUJcpRQ0oDhb8

Table   JSON
t @metdata.ip_address 208.90.215.182
⊙ @timestamp          February 8th 2018, 17:28:59.882
t @version            1
t _id                 txgueGEBUJcpRQ0oDhb8
t _index              cl-http-2018.02.09
# _score              1
t _type               bro
? _write_ts           ⚠ 2018-02-09T01:27:32.622916Z
t host                www.mybusinessdoc.com
t id_orig_h           192.168.0.53
# id_orig_p           2,210
t id_resp_h           68.164.182.11
# id_resp_p           80
t method              GET
t path                http
# port                42,312
# request_body_len    0
t resp_filenames      551d88323f7e.gif
t resp_fuids          Fv0xAo1XfKF6XdVV5g
t resp_mime_types     application/x-dosexec
# response_body_len   192,512
t sensor              HQ
# status_code         200
t status_msg          OK
t tags
# trans_depth         2
⊙ ts                  February 8th 2018, 17:27:32.610
t type                bro
t uid                 CSeT6u3O07GrhjWVW5
t uri                 /document.php?rnd=5292&id=5555525E011600
t user_agent          Mozilla/4.0 (compatible; MSIE 7.0; Windo
⚠ version             1.1
```

```
Single Document
bro#phYteGEBUJcpRQ0ov47G

Table   JSON
t @metdata.ip_address 208.90.215.182
⊙ @timestamp          February 8th 2018, 17:28:39.603
t @version            1
t _id                 phYteGEBUJcpRQ0ov47G
t _index              cl-files-2018.02.09
# _score              1
t _type               bro
? _write_ts           ⚠ 2018-02-09T01:27:32.622916Z
t analyzers           MD5, PE, SHA256, SHA1
t conn_uids           CSeT6u3O07GrhjWVW5
# depth               0
# duration            a few seconds
t filename            551d88323f7e.gif
t fuid                Fv0xAo1XfKF6XdVV5g
t host                208.90.215.182
◑ is_orig             false
◑ local_orig          false
t md5                 634c2a2a3ab03d5c21730c62d4677fe8
t mime_type           application/x-dosexec
# missing_bytes       0
# overflow_bytes      0
t path                files
# port                42,288
t rx_hosts            192.168.0.53
# seen_bytes          192,512
t sensor              HQ
t sha1                a9a1911fe2ff864a7d181bb7750b60b74033c3b1
t sha256              196c186b05ce2cb0f964080823d22a5f4c999e3270fd3b475068c5130dc7fd50
t source              HTTP
◑ timedout            false
# total_bytes         192,512
⊙ ts                  February 8th 2018, 17:27:32.610
t tx_hosts            68.164.182.11
t type                bro
```

The information about each network endpoint is summarized by the id field, which is usually represented as four separate fields:

- id.orig_h
- id.orig_p
- id.resp_h
- id.resp_p

This nomenclature may seem odd to use, because networking personnel traditionally refer to sessions using client and server; however, using orig (originator) and resp (responder) allows security personnel to accurately describe the connection. Think of the originating host (orig_h) as the source, or client, and the responding host (resp_h) as the destination, or server. The field id.orig_p and id.resp_p will be populated with the corresponding port numbers.

Many of the remaining fields within the conn log and other protocol logs are self-descriptive, but if you get stuck, look at the Zeek® documentation at https://docs.zeek.org/en/current/ for more detailed information or visit the community slack channel at http://corelightcommunity.slack.com/.

IDENTIFYING USERS AND DEVICES

When identifying devices on a network, the IP or MAC addresses are regularly used to create the 'identity.' The device IP address is used more often for the remote identity of a device because it survives router boundaries. When inside a network segment, the MAC address is preferred for identification because it can be a reliable identifier of a specific machine. Each identifier has pros and cons, and the ability of Corelight to capture both aids SOC personnel as they investigate events.

While IP addresses are durable[2] for internal investigations, they often are transient within a network due to most networks implementing DHCP (Dynamic Host Configuration Protocol). Transient IPs are problematic for defenders when the IDS alert identifies the session by IP addresses. Those IP addresses are only related to the alert at the time that the alert was generated.

You can use open source tools when conducting an investigation (e.g., nslookup), to provide DNS information for remote IPs. However, this is a point-in-time piece of information at the time of the investigation, not when the event occurred. A better technique is to use logs created at the time of the alert to capture the IP and FQDN (fully qualified domain name) for the remote device. To locate the internal device, you can mine DHCP logs to identify it. There are multiple ways to identify a host and Corelight provides this data in multiple logs that each tell a different aspect of the story. Exercise creativity and follow every lead.

Where hostnames can be found:

- **dhcp.log:** host_name and domain fields represent the hostname and domain reported by a host when requesting an IP address via DHCP, and the assigned_addr field is the IP address that was assigned to that host.
- **dns.log:** if there's an IP in the answers field, then the query field contains the hostname that the DNS server recorded (at that time) for the IP address.
- **ntlm.log:** server_dns_computer_name and server_nb_computer_name refer to the DNS and Netbios names of the machine with the IP address in the id.resp_h field. The hostname field is the hostname of the machine with the IP address in the id.orig_h field.
- **kerberos.log:** in a Windows environment, for domain-joined devices, kerberos requests where the client field contains a name ending in $, the client field is the hostname, and the id.orig\_h field is the IP address of that host. The client field is often structured like HOSTNAME$/EXAMPLEDOMAIN.COM where HOSTNAME is the hostname and EXAMPLEDOMAIN.COM is the Windows domain name and Kerberos realm name.
- **http.log:** the host field contains the hostname, domain name, or IP address of the client that requested data from the HTTP server. Sometimes this field is an indication of the identity of the server, the device with the IP address in the id.resp_h field.
- **ssl.log:** server_name field is extracted from the Server Name Indication (SNI) field in the TLS/SSL negotiation, and is used similarly to the host field of the http log. Also, the subject field is extracted from the subject of the server certificate, and the canonical name CN portion of the subject can provide clues to identify a server.

When identifying users, there are several logs that provide valuable information:

- **rdp.log:** depending on the version of the RDP protocol, the value of the cookie field is the username asserted by the client, and the client IP is in the id.orig_h field.3
- **ftp.log:** the user field contains the username asserted by the client, and the client IP address will be in the id.orig_h field.
- **irc.log:** the user field contains the username asserted by the client, and the client IP address will be in the id.orig_h field.
- **socks log:** the user field contains the username asserted by the client, and the client IP address will be in the id.orig_h field.

- **http.log:** the username field contains the username asserted by the client, and the client IP address will be in the id.orig_h field, or may be indicated in the proxied field if the connection was proxied. If proxied the id.orig_h field will contain the IP address of the proxy.
- **ntlm.log:** the username field contains the username asserted by the client, and the client IP address will be in the id.orig_h field.
- **kerberos.log:** in a Windows environment, kerberos requests contain the username in the client field (except for requests where the client field contains a name ending in $, which means that the asserting identity is a device, and the id.orig_h field is the IP address of the source device. The client field will often be structured like USERNAME/EXAMPLEDOMAIN.COM where USERNAME is the username and EXAMPLEDOMAIN.COM is the Windows domain name and kerberos realm name

A few words of warning about drawing conclusions about the identity of a machine or the user of a device: know your limits (and the limits of the data). Just because a username was recorded in network traffic does not mean that the actual person with that name is responsible — it is just a clue. You should check to see if the user authenticated successfully, as state-sponsored cyberspies and saboteurs have increasingly experimented with planting false flags.[4] The username could have been *asserted*, but if the authentication failed, then it is not a clear indicator that the user was involved. Don't forget that devices and software may cache credentials, so the user account may be active, but the actual person could still be innocent. You must continue to collect information before you can confirm nefarious behavior.

For example:

- A user goes to lunch and leaves their device unlocked
- A device is compromised with a Remote Access Trojan (RAT) and a user halfway around the world is surreptitiously assuming the identity of our victim, while the original user is also using the device simultaneously to conduct regular business
- A malicious user within the organization has overheard a coworker saying their password out loud in conversation, and he or she is now trying to use those credentials to log in to other systems

Also, make sure you understand what pieces of information are controlled and asserted by the clients or servers, and consider who controls each. If an adversary is inside your network, determining what information is trustworthy is paramount when preparing the response plan. For example, an intruder could disable DHCP and statically assign an IP address and use it to navigate the network, making identification difficult, as the DHCP server records would provide conflicting information. Additionally, when a client requests a DHCP address, an intruder could provide a false MAC address. Thus the importance of capturing passive point-in-time logs when the event occurred.

GET A FREE SET OF LOG CHEATSHEETS AT OUR WEBSITE

irc.log | IRC communication details

FIELD	TYPE	DESCRIPTION
ts	time	Timestamp when command seen
uid & id		Underlying connection info > **See conn.log**
nick	string	Nickname given for connection
user	string	Username given for connection
command	string	Command given by client
value	string	Value for command given by client
addl	string	Any additional data for command
dcc_file_name	string	DCC filename requested
dcc_file_size	count	DCC transfer size as indicated by sender
dcc_mime_type	string	Sniffed mime type of file
fuid	string	File unique ID

socks.log | SOCKS proxy requests

FIELD	TYPE	DESCRIPTION
ts	time	Time when proxy connection detected
uid & id		Underlying connection info > **See conn.log**
version	count	Protocol version of SOCKS
user	string	Username used to request a login to proxy
password	string	Password used to request a login to proxy
status	string	Server status for attempt at using proxy
request	record SOCKS:: Address	Client requested SOCKS address
request_p	port	Client requested port
bound	record SOCKS:: Address	Server bound address
bound_p	port	Server bound port

INDEX OF TTPS

HOW TO HUNT FOR SPECIFIC TTPS

INITIAL ACCESS

Initial access is when intruders establish their initial foothold.

Drive-By Compromise
A drive-by compromise usually results when a file is surreptitiously downloaded from a website that is compromised. When you hunt for signs of drive-by compromise in Corelight data, your main focus is downloads from external websites. Begin the hunt with the http log and look for signs of downloaded executables:

1. Start with http logs where resp_fuids is not empty. This means there was a file returned from the responder.
2. If the data volume is too large, filter out local (in-network) responders. You can filter by joining the results to the conn log on the uid, then filtering out any records in which local_resp is "true" in the conn log.
3. Review the resp_mime_types from the http log, and filter uninteresting results (e.g., images, text, OCSP responses, and certificates). Often the most interesting results are executables, dlls, and archives/containers
4. Group the results by the host and resp_mime_types fields for easy analysis.

Scan through the results and look for anything interesting or odd, such as downloads of executable files, or file extension and mime-type mismatch.

As more attackers move to using TLS to encrypt exchanges between compromised clients and websites they control, there will be less visibility via the http log. To regain this visibility, consider using an enterprise SSL decryption solution and passing the decrypted HTTP traffic to your Corelight Sensor.

External Remote Services
External remote services are used by adversaries to connect to internal network resources, and hunting for misuse of remote services usually involves two steps: discovery, and analysis. First, you must discover what remote services are in use. Asset and service inventory information should be collected first, but usually it's insufficient. Often, there is natural "drift" as IT teams make changes to infrastructure and struggle to keep asset documentation current. Empowered users make this more difficult by setting up assets and services without involving or informing IT, a process known as "shadow IT."

Traditional remote services, for example: RDP, VNC (remote framebuffer), and SSH (secure shell) contain a server component and a client component. If you have a remote service hosted in your environment, attackers can exploit externally accessible services to compromise machines inside the network. To identify these services, look for conn log entries in which the service field contains rfb, rdp, or ssh, and where local_orig is false and local_resp is true, or where the originator IP (id.orig_h) is external and the responder IP (id.resp_h) is on the organization network. Make note of any RFP/VNC, RDP, or SSH servers that are accepting connections from the internet.

Some remote services work in reverse, where an agent is installed on the local device, and it reaches outward from inside the network to a set of external servers, for example, GoToMyPC and TeamViewer. This configuration is designed to assist users (primarily home users) who don't control the NAT or the firewall or aren't sophisticated enough to be able to manage port forwarding or firewall rule management.

To discover if these remote services are in use in your environment, look for signs of outbound connections to the services. For example, TeamViewer uses TCP port 5938 to communicate with TeamViewer servers, so simply review the conn logs for connections where the id.resp_p is 5938 and local_orig is true and local_resp is false. TeamViewer also uses SSL, and the domain name of the connections should be *.teamviewer.com, so additionally you can look for entries in the ssl log in which the server_name contains, or better yet ends with, “teamviewer.com.” (Note: because this session works in reverse, the id.orig_h is the device in your network that has the TeamViewer client installed.) Our second example, GoToMyPC, attempts to contact poll.gotomypc.com. Examine the http log host field for poll.gotomypc.com, or entries in the ssl log in which the server_name is poll.gotomypc.com. For each client software package, the list of ports and domain names varies.

Now that we’ve discussed the discovery of remote services, you should compare Corelight data to a list of all remote services that the IT department offers, such as:

- RDP Gateways
- VDI (Virtual Desktop Infrastructure) Gateways
- VPN (Virtual Private Network) Gateways
- SSH Servers

For each service exposed to the internet, aggregate a list of connections to that service from the conn log, and include the following fields:

- id.orig_h: Origin IP address (client)
- id.resp_h: Responder IP address (server)
- id.resp_p: Responder port
- service: the application protocol that Zeek detected
- history: the history of the connection, e.g. what types of TCP flags were seen
- orig_cc: The originator’s country code

When filtering logs, ensure the history field starts with “Sh.” For TCP connections this means that the originator sent a SYN, and the responder replied with a SYNACK (handshake). This check eliminates connections where the server is not listening, or there is a firewall blocking the connection.

After you have gathered all the data, begin sifting through the logs for anything interesting, such as a connection from a country that is not expected. Use the UID from the conn log to follow-up with the application-specific Zeek logs (rdp, rfb, ssh). For example, the rdp log contains more details about the connection, such as the cookie field that can contain the username of the authenticating user. The last step is to check with the user to determine whether they were actively using the system at that time.

Corelight customers have access to the Encrypted Traffic Collection (ETC) that generates inferences, or insights, about encrypted traffic. The ssh log contains interesting information inferred about the SSH connection, such as:

- KS for connections that appear to contain client keystrokes
- FU and FD for connections which appear to contain a file upload or download, respectively
- ABP for connections which appear not to contain any authentication, but still are successful (“authentication bypass”)
- SV or SC for clients that appear to be version or capability scanning, respectively

If you’d like to learn more about the Corelight ETC, please contact our sales team at (510) 281-0760.

Spearphishing Attachment
As a method of entry into an organization, an adversary may send a well-crafted malicious attachment to an individual or a small group in a spearphishing campaign. The attachment could be a document that instructs the user to take some action, such as clicking a link and/or logging in to a portal; or it could be a file crafted to exploit a vulnerability in the software used to open it, such as Adobe Acrobat or Microsoft Word.

The Corelight smtp log contains records in the fuids field if there were any files attached to a message delivered over SMTP. This field can be used to pivot to the files log which contains detailed information about the file including filename, hashes, and the source.

For example, examine this sample log at left.

```
path: smtp
from: Your Friend <Jeremy.Rigeur@gmail.com>
fuids: [ Fh5GBc1wdVp3x9MKxc ]
mailfrom: attacker@fake-mail.com
rcptto: [ victim@corp-mail.com ]
subject: Definitely not a spear-phish
to: [ victim@corp-mail.com ]
uid: CzKseq1Y3zo2qsTYH5
user_agent: Apple Mail (2.3608.80.23.2.2)
```

To hunt for potential spearphish attempts, you can search in the files log:

1. The value in the source field is SMTP.
2. Filter out any uninteresting mime_type and/or filename values, as previously mentioned.
3. Use the hash (MD5, SHA1, or SHA256) with a file reputation service (such as Virustotal) to look for known malicious files.

```
path: files
conn_uids: [ CzKseq1Y3zo2qsTYH5 ]
filename: WIRE_FRAUD.pdf
fuid: Fh5GBc1wdVp3x9MKxc
md5: e71c36cddd2aa42670d89d63e653d1da
mime_type: application/pdf
sha1: bb24829550c0ca17db73d80a1d2f969e3b06ff5f
source: SMTP
```

Additionally, you may start from the smtp log:

1. To reduce the data look for entries where the fuids field is non-empty.
2. Filter out known good combinations of mailfrom and from values.
3. Filter out uninteresting subject values.
4. Consider using the fuid value from the remaining records to pivot to the files log to get more information about the file.

Corelight can perform high-speed file extraction and can filter based on MIME type, so any interesting files, such as executables, Office documents, and PDFs are available for more scrutiny if desired.

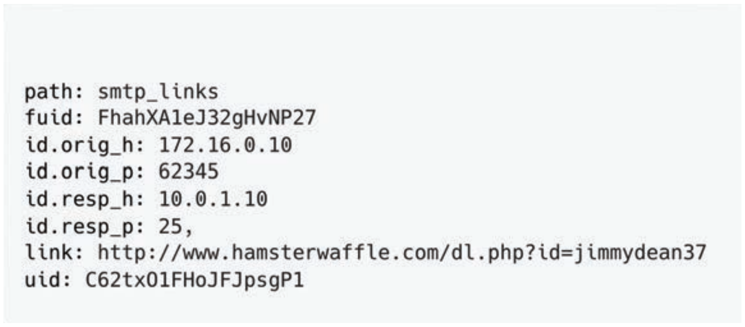
Much of the mail that crosses the internet today is encrypted via STARTTLS over the SMTP protocol, and this hinders visibility. To achieve better visibility without sacrificing privacy and security for your users, it is a best practice to accept inbound SMTP at a system that supports STARTTLS, then proxy the mail to the internal mail system, so that Corelight can generate the corresponding logs.

Spearphishing Link

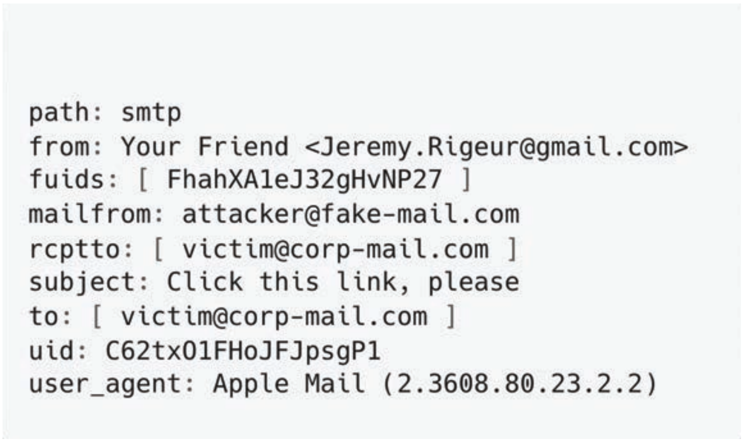
Instead of sending files into an organization where they can be scrutinized by a corporate mail filter, some adversaries send emails that only contain links. These links lead to websites that are controlled by the attacker, and attempt to dupe the user into:

- Entering credentials that the attackers harvest
- Exploiting a vulnerability in the user's browser
- Downloading a file to exploit another application on the user's device

Corelight Sensors have a package[5] that can log links from SMTP messages into a separate log, the smtp_links log. This log contains a fuid field, which links the smtp_links log to the smtp log. You can quickly pivot to the smtp log with the details about the message that delivered the malicious link. For example, see the log examples.



```
path: smtp_links
fuid: FhahXA1eJ32gHvNP27
id.orig_h: 172.16.0.10
id.orig_p: 62345
id.resp_h: 10.0.1.10
id.resp_p: 25,
link: http://www.hamsterwaffle.com/dl.php?id=jimmydean37
uid: C62tx01FHoJFJpsgP1
```



```
path: smtp
from: Your Friend <Jeremy.Rigeur@gmail.com>
fuids: [ FhahXA1eJ32gHvNP27 ]
mailfrom: attacker@fake-mail.com
rcptto: [ victim@corp-mail.com ]
subject: Click this link, please
to: [ victim@corp-mail.com ]
uid: C62tx01FHoJFJpsgP1
user_agent: Apple Mail (2.3608.80.23.2.2)
```

To hunt for spearphishing links, start with the smtp_links log and review the link field, filtering out benign domains until you find interesting results. Another option is to join the smtp_links log to the smtp log via the fuids or uid field, and filter out benign combinations of mailfrom and from fields to look for messages from unique senders.

While much of the mail that crosses the internet today is encrypted via STARTTLS over SMTP. To achieve better visibility without sacrificing privacy and security for your users, it is a best practice to accept inbound SMTP at a system that supports STARTTLS, then proxy the mail to the internal mail system, so that a Corelight solution can generate the corresponding logs.

EXECUTION

The adversary is trying to run malicious code.

Command Line Interface, PowerShell

Command line interface scripting has long been used to manage *nix-based systems, and the ability to build and execute scripts is often exploited by attackers. For years there was no equivalent available on Windows, and in the early 2000s Microsoft began development of a new approach to command line management. Soon thereafter, PowerShell (PS) 1.0 was created. PS, in its various iterations, is a built-in tool based on the .NET framework that's used to automate system administration tasks. It provides an interface for users to access services of the Windows operating system.

Although certain PS commands are restricted by default, many commands are available to obtain system information without an executable file. You can use LNK extensions to bypass safeguards and execute a PS script. LNK files are usually seen as shortcuts, generally found on users' Desktop and Start Menu.

Malicious LNK files are often embedded within what appears to be legitimate documents or pictures. Once opened, the LNK executes a legitimate windows application CMD.exe or MSHTA.exe to bypass security settings. Corelight's file extraction capabilities and integration with various intel platforms provide insight into malware obfuscated by file type. By utilizing Corelight's built-in filtering, you can tune the file extraction parameters to target specific mime-types that are commonly used for malware delivery, including:

- Compressed files
- Microsoft Office (Word, PowerPoint, etc)
- PDF files
- TXT files (powershell, vbs)

PERSISTENCE

Persistence is the adversary trying to maintain their foothold.

BITS Jobs

Microsoft Background Intelligent Transfer Service (BITS) was created in 2001 as a mechanism for managing file transfers that minimize disruption to the end user. BITS is commonly used to download Windows updates and other software updates from major vendors.

Attackers have two methods of abusing BITS:

- The most common is to create a BITS transfer job directly on a host, allowing a download of secondary payloads through a built-in Windows service that typically bypasses firewalls and other security controls.
- Another alternative is to exfiltrate data through a BITS upload job. Uploads must connect to an IIS server for BITS to function properly, but this requirement is trivial for malware authors to subvert.

Data transfers using the BITS service can take place over HTTP, SSL, and SMB. When BITS uses HTTP traffic, there is a distinctive User-Agent string of "Microsoft BITS/7.5" (or 7.8 in later versions). Unfortunately, there are no distinguishing characteristics of BITS SSL and SMB network traffic. Therefore, the presence of BITS network traffic is not necessarily suspicious, because it is present anywhere Windows machines are connected to the internet. Analysts still can use Corelight data to assess if the BITS traffic is legitimate by analyzing remote systems being used for BITS data transfers. If they are outside of CDNs or major software providers' networks, all BITS uploads should be investigated until proven benign, as this use case is especially rare among legitimate software vendors.

The code sample (next page) is an http log showing what the BITS data looks like if it is over HTTP.

External Remote Services

- See *Initial Access: External Remote Services*

Port Knocking

Port knocking is a technique to get a remote system to enable access to an otherwise closed port. It typically consists of a pre-defined sequence of connections to other (often closed) ports, sometimes with special protocol-level flags, Layer 7 banner strings, etc.

Zeek summarizes each TCP, UDP, and ICMP connection in the conn log. This detailed log provides useful statistics about connections. The history, conn_state, and network tuple (src/dest ip/port) fields provide the information necessary to sight port knocking. It is important to note that sighting port knocking without an additional hint can be a daunting task, as it is easy to hide intentional sequences of connections among the noise of a typical network.

```
path: http,
uid: Ca9LrF3xl5kVCxe2K4,
id.orig_h: 10.10.199.31,
id.orig_p: 49987,
id.resp_h: 151.205.0.135,
id.resp_p: 80,
trans_depth: 1,
method: GET,host:151.205.0.135,
uri:/pdata/0731497c8fa1dce5/download.windowsupdate.com/d/msdownload/update/software/secu/20
18/05/windows10.0-kb4103723-x64_0722ab30824410046f954417ada8556d2ac308a6.cab,
version: 1.1,
user_agent: Microsoft BITS/7.8,
request_body_len: 0,
response_body_len: 1333068983,
status_code: 200,
status_msg: OK,
resp_fuids: FD283F3hrZH8yzYmb8,
resp_filenames: windows10.0-kb4103723-x64_0722ab30824410046f954417ada8556d2ac308a6.cab,
resp_mime_types: [application/vnd.ms-cab-compressed],
accept_encoding: identity,
accept: */*
```

Server Software Component: Web Shell
A web shell is a web-based implementation of a command shell. A web shell is generally a malicious web page or code snippet introduced into an existing web server or application to provide unauthorized access. This access can be an actual CLI shell, file management, or database access tool. This is a common tactic used by web shells that includes blending malicious traffic with benign traffic to/from the web server, making it difficult to identify via IDS signatures due to the ease of changing specific web shell characteristics.

When a web shell executes, it runs with limited web server software user permissions. Attackers often use web shells to attempt privilege escalation attacks by exploiting local vulnerabilities on the system to assume root privileges.

Detecting web shells on the network using signature-based detections is relatively straightforward, as web shells have specific file paths, communication methods, or other behaviors that can trigger an alert. However, like most 'atomic' IOCs, they are easy to evade because they identify specific behaviors that can easily be changed. Therefore, it is recommended to supplement signature detection with a threat hunting program to find more general behaviors of anomalous activity.

Web shells attempt to hide malicious activity in normal HTTP traffic, making the http.log an excellent data source for investigating web shell activity. Examples of hunt hypotheses supported by Corelight HTTP data are:

- Unusual HTTP POST activity. This may be as simple as unexpected HTTP POSTs in the 'method' field of the http.log where GETs are expected (if the affected site is primarily serving content).

- 'Normal' web traffic travels to a shortlist of common pages, with navigation via an internal hyperlink. A web shell goes directly to the hidden page and appears as an HTTP request with no referring page. Additionally, web traffic shows a variety of requesting IPs, user-agent strings, JA3s, etc. A web shell can have a more homogenous group of users.

- Ferreting out suspicious logins originating from internal subnets to DMZ servers and vice versa.

This type of hunt analysis and anomaly detection is an effective way to identify malicious (or suspicious) activity, but modern networks are noisy, chaotic places. As with most hunts, you must know what 'normal' data looks like so that you can successfully filter it out. (https://github.com/nsacyber/Mitigating-Web-Shells)

DEFENSE EVASION

Defense evasion consists of techniques that adversaries use to avoid detection throughout their compromise.

BITS Jobs

- See *Persistence: BITS Jobs*

Port Knocking

- See *Persistence: Port Knocking*

Install root certificate

Public certificates are used to establish secure TLS/SSL communications. Root certificates are used to identify the root certificate authority (CA). Root certificates are self-signed and form an anchor of trust for public key cryptography. For example, when a root certificate is installed, the system or application will trust certificates in the root's chain of trust. While no network-level device (e.g., routers and switches) can show the certificate chain installed on a client system, the point of installing a malicious root certificate is to bypass trust validation.

Using Corelight data, you can observe all aspects of the TLS/SSL session using the ssl and x509 logs. These two logs allow analysts to identify certificates that seem suspicious by:

1. Searching the ssl log for any entries where the validation_status field doesn't have a value of ok.
2. Reviewing records where the validation_status field either has a self-signed certificate or contains a self-signed certificate in the certificate chain.
3. Reviewing the subject and server_name fields to determine the likely organization or website that controls the server.
4. Filtering results where there are legitimate self-signed certificates in use, such as in communications between IOT devices and the supporting cloud infrastructure.
5. Investigating the id.resp_h IP address to see what Autonomous System the session belongs to and whether it's a reasonable AS organization (such as the organization that matches the information on the server, or a commonly-used cloud hosting provider)
6. For remaining connections, use the values in the cert_chain_fuids to pivot to the certificates in the x509 log and review the certificate details.

Focus your investigations by inspecting the local root certificate authority on the endpoint.

CREDENTIAL ACCESS

Credential access is when the adversary tries to steal account names and passwords.

Brute Force

An adversary attempts to gain unauthorized access by systematically guessing a user's password using a repetitive or iterative mechanism. Sometimes a brute force attack originates from a list of known information, increasing the likelihood of success.

For example, an attacker attempting to guess the password of an Active Directory account likely results in many connections to a Domain Controller on the LDAP port (389 or 636). An attacker attempting to discover API URLs in an e-commerce system generates many more connections to the web server than other clients in a similar time period and creates more HTTP status codes in the 400 and 500 range (errors) compared to other clients.

To look for a brute force attack:

1. In the conn log, aggregate by id.orig_h, id.resp_h, id.resp_p, proto, and (optionally) service.
2. Add a count for the number of operations and sort by the highest counts.
3. Choose a time period that makes sense, based on the size of the network/data set, starting small and gradually increasing.
4. Filter records that are obviously permissible, such as repeated contacts from network or application performance monitoring systems, vulnerability management systems, or business applications
5. For unknown or suspicious records, perform a deeper investigation on that behavior. For example, look for other connections originating from the remote IP address.
6. For protocols that can maintain connections over multiple transactions or attempts, look for long-standing connections. These long-standing connections can also indicate repetitive behavior.

Corelight Sensors include a script that logs connections that are maintained for longer than a set of thresholds, starting at ten minutes and continuing up to three days. If you are not a Corelight customer, but use open source Zeek, this script is available through the Corelight GitHub page.

To hunt for long connections with the Long Connections package installed:

1. Examine the notice log.
2. Review the entries where the note is "LongConnection::found,"
3. Review each set of id.orig_h, id.resp_h, id.resp_p to understand whether these devices should have long connections.

To hunt for long connections without the Long Connections package installed:

1. Examine the conn log.
2. Gather a list of all connections with the following fields for each: id.orig_h, id.resp_h, id.resp_p, proto, service, and duration fields. This only includes connections that completed, either properly or via timeout. Currently-open connections are not represented in the results.
3. Sort the results by duration, bringing the longest connections to the top.
4. Investigate each result to determine whether it's legitimate or expected behavior.
5. Filter out expected behaviors and thoroughly investigate anything that seems suspicious.

Corelight also gives you the Encrypted Traffic Collection (ETC), which automatically looks for brute force password guessing attempts against SSH servers within a single connection.

Forced Authentication
Some protocols automatically authenticate when a user accesses a resource without first checking to see if the resource being accessed is trusted. For example, an attacker can embed a reference in a Microsoft Office document to a file that's hosted on an attacker-controlled UNC path (\servername\sharename\path\to\file). When the user opens the file, the machine attempts to access the resource. The attacker-controlled server then challenges the machine for authentication, and under most circumstances, the victim machine automatically provides cached credentials, usually in the form of an NTLM hash. The attacker can then attempt to use the credentials for unauthorized access, usually through reversing the hash to get the password, or re-using the hash in a pass-the-hash attack.

This method requires the attacker to control server infrastructure. As a result, the most likely attack vector is spearphishing. The attacker phishes a user on the network, and the victim machine then reaches out to the attacker-controlled server across the internet. To hunt for this behavior, look for authentication across the internet:

1. Look in the ntlm log for any signs of NTLM authentication in which the destination IP is on the external network.
2. Look for entries in the conn log in which the service field contains smb (and/or ntlm), and local_resp is false.

In LLMNR or NBT-NS poisoning, an attacker listens to local LLMNR or NBT-NS broadcasts asking for a particular resource by name. The attacker then responds to the querying client spoofing the actual resource. If the resource is one that usually requires authentication, then the attacker can challenge the client for authentication. When the client authenticates, usually with a password hash, the attacker uses the credentials to impersonate the client and access resources.

You can effectively hunt for these attacks with Corelight data, but the sensor needs to be inside of the broadcast domain because broadcast traffic doesn't typically traverse routers. Typically you need to span or mirror entire VLANs, or forward LLMNR or NBT-NS traffic from client subnets and VLANs, to places on the network that Corelight is monitoring.

Look for dns logs where id.resp_p=5355 (LLMNR) or id.resp_p=137 (NBT-NS), and filter for records where the answers field is non-empty. Then count the number of distinct query fields per id.resp_h. This search yields IPs that respond to more than one name.

Network Sniffing
You can't detect an intruder who is sniffing traffic on your network using network logs because the action is invisible; however, *you can detect an intruder by sniffing on your own network* because your adversary can't see it.

Corelight Sensors enable you to deploy an out-of-band sensor grid that generates linked logs. These logs speed reliable observation and detection, and assist in avoiding the pitfall of prevention dependence, while providing context for a deeper and more accurate historical analysis. As Rob Joyce, Chief of the NSA Tailored Access Operations division, put it in his 2016 USENIX talk, "We are doing nation-state exploitation...what can you do to defend yourself to make my life hard?"

DISCOVERY

The adversary is trying to learn about your environment.

Network Service Scanning
To determine which devices on a network are exploitable, and the services available on those devices, an intruder can employ active scanning. Active scanning methods include:

- Horizontal scanning: Sending connection requests to a specific port across many IPs to see which IPs respond. For example, scanning across many devices on port TCP/22 typically reveals devices running an SSH server. Scanning across many devices on port TCP/445 can effectively enumerate Windows infrastructure.

- Vertical scanning: Sending connection requests to a single IP address across many ports to see which ports respond. This method lets attackers infer services available from that IP address.

Each of these methods can be performed using a free or commercially-available vulnerability scanner. These products often add other logic to check service availability, version information, and if services are vulnerable to known exploitation techniques.

If an intruder uses one or more of the above methods to attempt service discovery, the byproduct is a failed or rejected connection. In Corelight data, these are recorded in the conn log as connections with a conn_state of S0 (initiated, and ignored) or REJ (initiated, and rejected), and typically have a history field where there is no 'D' (post-syn data from the initiator). To look for network service scanning internal to the network:

1. Search for entries in the conn log where conn_state is S0 REJ.
2. Filter for records where local_orig=true and local_resp=true.
3. Group and count the results by the id.orig_h, and the number of unique id.resp_p, to assess the horizontal/verticalness of the scan.
4. Inspect the list, starting with the records that have the highest count of id.resp_h or id.resp_p.
5. Identify the originator (id.orig_h) and review the list of responders (id.resp_h) and ports (id.resp_p).
6. Determine whether the behavior is acceptable based on the identity of the source, the ports involved, and the destinations.

Not all items on the list are malicious. DHCP servers, for example, are commonly configured to ping an IP address to confirm if the address is in use before assigning it from the pool. Print servers with a large number of print queues attempt SNMP and/or network printing services to printers, even if those printers are offline. For this reason, print servers can cause large numbers of S0 connections. Of course, software that scans legitimately, such as a corporate-sanctioned vulnerability scanner or an inventory management system, might appear in the list. Finally, network engineers conduct ad-hoc network scanning for troubleshooting purposes. If you run across network scanning, modify the original query to omit the records that are known to be benign, then resume hunting.

Network Share Discovery
The most common network sharing protocol abused by attackers is SMB, the standard for Windows file sharing. SMB is supported by every modern operating system. High-value documents that store PII, trade secrets, network diagrams, and other sensitive data, typically live on SMB shares in enterprises of all sizes.

Scanning for and discovering shares on an SMB server is typically done using a DCE/RPC command on TCP port 445. Specifically, a connection to the "srvsvc" pipe — which shows up in the dce_rpc logs as an endpoint by the same name — is followed by a call to the NetShareEnumAll or NetShareEnum functions (called "operations" in the Zeek log). These function calls are used for legitimate file-sharing purposes, and taken alone they are insufficient indicators of malicious intent. However, in combination with other indicators of lateral movement, they illustrate how an attacker moved laterally within a network. Prime targets for further investigation are ones that generate a large number of DCE_RPC function calls across a large number of hosts in a short period.

Network Sniffing (X-reference)
- See *Credential Access: Network Sniffing*

Remote System Discovery
The same principles for detecting Network Service Scanning apply to detecting Remote System Discovery. See this section for more information.

LATERAL MOVEMENT

Lateral movement is what adversaries use to enter and control remote systems on a network.

Remote Desktop Protocol
The Microsoft Remote Desktop Protocol (RDP) is used to remotely control a Windows endpoint. This protocol can be abused by an attacker to gain unauthorized access to your network (see Initial Access: External Remote Services). Once an intruder is inside, they can use RDP to move laterally among devices.

RDP is one of the many protocols parsed by Corelight. For some environments, the presence of RDP, or its presence on specific systems, is sufficient to trigger an investigation. For networks where RDP is permitted, the Zeek RDP log is rich in information that helps establish whether a connection is legitimate, for example, recording data like keyboard layout, encryption levels, or client name for a connection.

When hunting with the rdp log:

1. Focus on the id.orig_h, id.resp_h, id.resp_p, and cookie fields. The cookie field can contain any arbitrary value sent by the RDP client to the server, but it often contains the username sent by the RDP client.
2. Aggregate the records based on these four fields and show a count for each unique set.
3. Iterate through the set and identify the origin and destination of each connection (e.g., you can use the records from the DNS and DHCP logs).
4. Some RDP connections will use a non-standard keyboard layout. To look for this, examine the keyboard_layout field. Count the number of instances of each value and apply data stacking to look for outlying or rarely occurring values.
5. Identify the origin and destination and determine whether the non-standard keyboard layout is expected, for instance, if the origin user is known to have a non-English language as their primary language, and that language is the requested language in the RDP connection.

After you have this information ask several questions:

- Does the cookie value match the expected user at the source or destination machine?
- Is there a legitimate reason for the originator to be using RDP?
- Are there any users using RDP where you wouldn't expect that for their job function?

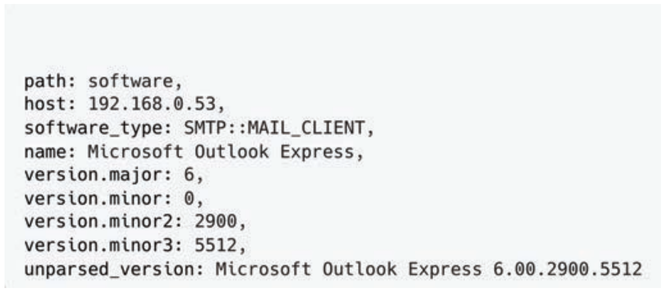
Remote Services
Exploitation of a software vulnerability occurs when an adversary takes advantage of a programming error to execute adversary-controlled code. This exploitation can happen in a program, service, or within the operating system software or kernel itself. A common goal for post-compromise exploitation of remote services is for lateral movement.

Given the complexity of today's enterprise networks, a variety of third-party and external services are often in

```
Path: http,
uid: CEeVS92Ljnr9jbW2J5,
id.orig_h: 54.235.163.229,
id.orig_p: 41855,
id.resp_h: 192.168.0.2,
id.resp_p: 80,
trans_depth: 1,
method: OPTIONS,
host: host-90-236-3-35.mobileonline.telia.com,
uri: *,
version: 1.1,
```

use. These services allow attackers to gain initial access or to move laterally. All connections are logged within conn.log, however, more details may be available within protocol-specific logs depending on the nature of the remote service under attack. For example, you can monitor the http.log file for suspicious and unexpected HTTP requests (such as OPTIONS requests).

Additionally, Corelight extracts information about software observed on the network into the software.log. This file provides defenders valuable data to monitor for unexpected or unauthorized servers, vulnerable or out-of-date services, and unpatched client software.

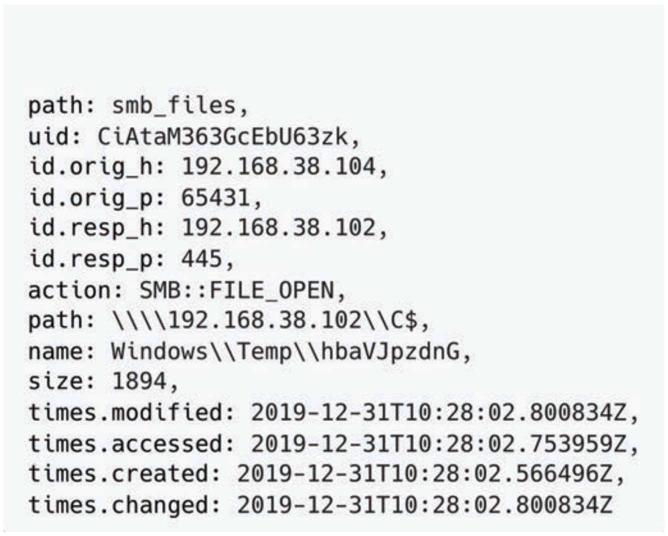


```
path: software,
host: 192.168.0.53,
software_type: SMTP::MAIL_CLIENT,
name: Microsoft Outlook Express,
version.major: 6,
version.minor: 0,
version.minor2: 2900,
version.minor3: 5512,
unparsed_version: Microsoft Outlook Express 6.00.2900.5512
```

Windows Admin Shares
Windows systems have hidden network shares that are accessible only to administrators and provide the ability for remote file copy and other administrative functions. Example network shares include C$, ADMIN$, and IPC$.

Attackers often use SMB to connect to administrative shares on Microsoft Windows workstations and servers. They may want to learn more about the target, extract sensitive files, upload malicious payloads, or authenticate so that further tools and attacks can proceed.

Corelight monitors SMB traffic, including authentication attempts, allowing defenders to log and notice patterns of administrative authentication attempts as well as monitor SMB traffic to extract transferred files. The example demonstrates the action FILE_OPEN being performed using the hidden admin share, and includes MAC information. Corelight logs the action performed including Open/Rename/Delete/Write.



```
path: smb_files,
uid: CiAtaM363GcEbU63zk,
id.orig_h: 192.168.38.104,
id.orig_p: 65431,
id.resp_h: 192.168.38.102,
id.resp_p: 445,
action: SMB::FILE_OPEN,
path: \\\\192.168.38.102\\C$,
name: Windows\\Temp\\hbaVJpzdnG,
size: 1894,
times.modified: 2019-12-31T10:28:02.800834Z,
times.accessed: 2019-12-31T10:28:02.753959Z,
times.created: 2019-12-31T10:28:02.566496Z,
times.changed: 2019-12-31T10:28:02.800834Z
```

COLLECTION

The adversary is trying to gather data to achieve their goal.

Archive Collected Data

To conceal data, attackers may consolidate data into compressed archive files, such as Zip, RAR, TAR, or CAB files. To hunt for this obfuscation technique, use the files log.

To search for compressed files:

1. Search all files logs, retrieving the tx_hosts, rx_hosts, mime_type, total_bytes, and source fields.
2. Remove records with uninteresting mime_types from the results, for example:
   a. application/x-x509-*
   b. application/ocsp*
   c. image/*
   d. audio/*
   e. video/*
   f. text/*
   g. application/xml
   h. application/chrome-ext

Automated Collection

Attackers can deploy automated tools on a compromised host to monitor intranet services for sensitive data and corporate secrets. These tools can include scripts to search for (and copy) information such as file type, location, or name at specific time intervals. Intruders may use remote access tools to conduct automated collection.

For example, a custom tool may query an intranet web server or an internal email server, polling regularly for new content. Corelight monitors multiple protocols including HTTP, email, MySQL, FTP, and SMB traffic to provide insight into these queries.

When hunting for automated collection use, defenders can identify automated tools by watching for repetitive queries or regularly scheduled connections. For example, if an intruder is web scraping, there will be a large number of connections from a finite number of IP addresses. Additionally, you can use the SMB logs (smb_files or smb_mapping) to identify anomalous traffic patterns.

Data From Network Shared Drive

Network shared drives are a treasure trove of sensitive corporate documents. Most enterprise networks host share network drives using SMB, but some may rely on FTP, HTTP, or even RDP. Zeek can monitor access to shared network drives when protocols like SMB, FTP, or HTTP are used. Remote control protocols, like RDP, are also parsed in protocol-specific logs. Anywhere Corelight sees this traffic, it is monitored and logged in the protocol-specific log.

The following example demonstrates the ftp log. Corelight logs the command and arguments.

```
path: ftp,
uid: C0EeI73um1Aw3rrOib,
id.orig_h: 10.0.0.11,
id.orig_p: 45831,
id.resp_h: 119.74.138.214,
id.resp_p: 21,
user: 1,
password: <hidden>,
command: RETR,
arg: ftp://119.74.138.214/doc.exe,
reply_msg: Transfer OK
```

COMMAND AND CONTROL

The adversary is trying to communicate with compromised systems to control them.

Commonly Used Ports/Non-Standard Ports
Adversaries may use a commonly used port to avoid more detailed inspection.

Hunting for C2 channels over commonly used ports is difficult, but not impossible. To look for C2 channels, search for well-known ports that are being used with an uncommon service.

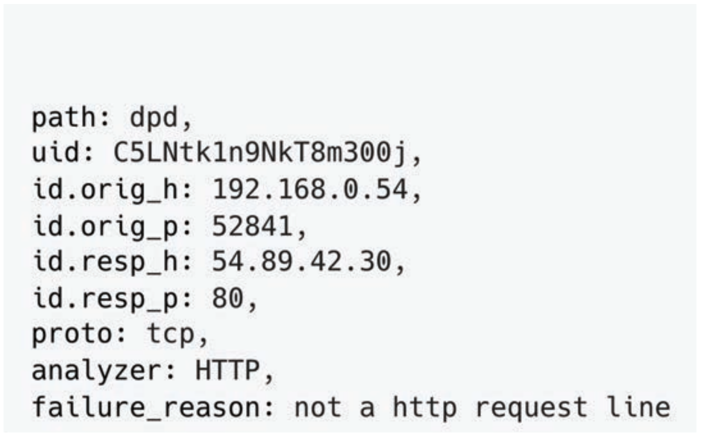
When hunting for C2 using commonly used ports:

1. Initially focus on the service field, and search the conn log for entries where the service field isn't what you would expect for the standard port (the service field could be either a '-' or another service).
   a. Start with the most common protocols:
      - TCP:80 (HTTP) TCP:443 (HTTPS)
      - TCP:25 (SMTP)
      - TCP/UDP:53 (DNS)

2. Corelight's Encrypted Traffic Collection contains a package titled Encryption Detection. Encryption Detection generates a notice when cleartext traffic is observed on usually encrypted ports. Observing notices for Viz::UnencryptedService highlights this behavior and helps you identify potentially malicious connections using common ports.

The Corelight Encrypted Traffic Collection package also has a feature that notifies you when a session uses instant encryption. The package looks for pre-shared keys or encrypted connections that begin without a traditional key negotiation. Observing notices for Viz::CustomCrypto highlights this behavior and helps you identify potentially malicious connections using common ports.

Additionally, you can use the Corelight dpd and weird logs to identify unexpected protocol behavior. These logs show debugging and parsing errors and identify out-of-specification usage of common ports and protocols — which might indicate malicious activity or covert use of known ports and protocols.



```
path: dpd,
uid: C5LNtk1n9NkT8m300j,
id.orig_h: 192.168.0.54,
id.orig_p: 52841,
id.resp_h: 54.89.42.30,
id.resp_p: 80,
proto: tcp,
analyzer: HTTP,
failure_reason: not a http request line
```

Encrypted Channel
See the Commonly Used Ports section for a description of Corelight's Encryption Detection package, the dpd log, and the weird log. These help you identify potential custom cryptographic protocols.

Fallback Channels, Multi-Stage Channels
Adversaries have been known to split communications between different protocols, using one for inbound C2 and another for outbound data. This allows for the communication to bypass firewall restrictions.

Malware that splits communication between two hosts for instructions and for exfiltration introduces a new challenge for defenders. Recognizing the linkage between suspicious control traffic and large data transfers is challenging, but Zeek provides packages and frameworks that synthesize data. For example, there is a package for determining the producer-consumer ratio for connections that identifies imbalanced, and possibly suspicious, data transfers. Additionally, the Intelligence Framework enables coordination with other defenders by identifying possible indicators of compromise (IP addresses, email addresses, and domain names) in Corelight data.

It's difficult to correlate attackers using different communication methods and channels but Corelight content, along with Zeek frameworks and packages can help. They allow defenders to identify the hidden channels discreetly, providing multiple opportunities for detection.

Beyond watching for the previously mentioned C2 communication mechanisms, here are some other signs available in Corelight data:

- Use conn.log to identify communication patterns that indicate additional channels (e.g., using orig_h and resp_h to narrow connections to a time window and observe connections between the hosts that include odd ports, failed or refused connections, or interesting/suspicious elements).
- Use Corelight (ETC), or self-developed content, in conjunction with connection log discovery to find potential relationships between overlapping, adjacent, or interesting connections.
- Search for sequences of connections to unrelated hosts using different protocols or events in the dpd and weird logs as described in Commonly Used Ports.

Ingress Tool Transfer
Intruders typically move files onto compromised systems — both tools that can assist with further lateral movement, and/or sensitive files designed for exfiltration. Those files will typically move over an HTTP(S), SSH, or SMB connection.

For files moving over plaintext HTTP, details like the remote host name and the name and mime type of the file being transferred can be useful indicators; users should also consult the files log for the hashes of files being moved, as many popular attacker tools have known crypto hashes that make identifying them easy. In the case of HTTPS, defenders can use the IP address of the remote system, as well as the certificate details noted in the ssl log (i.e., organization name, FQDN of the remote host from the CN, etc.) to look for anomalous connections.

Intruders copy files from one endpoint to another as they move laterally among compromised assets. Traditionally, file copies to or from Unix/Linux systems occur over the SSH protocol using the scp command. For Windows systems, remote file uploads or downloads typically happen over SMB, but also may use SSH via PUTTY.

Corelight Sensors with the ETC SSH inferences package enabled extend the ssh log. The extension includes an inferences field that adds inferred characteristics about the SSH traffic. For example, if the session is being used to move files, or if it is interactive:

- LFU: Large File Upload
- LFD: Large File Download
- KS: Keystrokes

To begin hunting for interesting SSH sessions use the inferences field in the ETC SSH package:

1. Identify sessions where the inferences field contains LFU, SFU, LFD, or SFD
2. Determine whether file activity via SSH is legitimate and expected

Corelight Sensors are preloaded with the MITRE BZAR (Bro/Zeek ATT&CK-Based Analytics and Reporting) package. MITRE BZAR identifies MITRE ATT&CK techniques for remote file copy, namely files being copied to C$ or ADMIN$ shares. This package generates entries in the notice log, as depicted here.

```
path: notice,
uid: CiAtaM363GcEbU63zk,
id.orig_h: 192.168.38.104,
id.orig_p: 65431,
id.resp_h: 192.168.38.102,
id.resp_p: 445,
fuid: FSeaVF4qnjl8cT3HF8,
file_mime_type: text/plain,
file_desc: Windows\\Temp\\hbaVJpzdnG,
proto: tcp,
note: ATTACK::Lateral_Movement_Extracted_File,
msg: Saved a copy of the file written to SMB admin file share,
sub: 2020-10-23/6f24ac6ce591baf02acd64684f596d2db0ec97c0,
src: 192.168.38.104,
dst: 192.168.38.102,
p: 445,
actions: [Notice::ACTION_LOG],suppress_for:3600.0
```

Even if you do not enable the MITRE BZAR package on your Corelight Sensor, Corelight still logs SMB share access in the smb_mapping log and file access and modification in the smb_files log.

The logs below illustrate the data contained in the Corelight family of SMB logs:

To hunt for lateral movement:

1. Start by searching the smb_files logs, and focus on the id.orig_h, id.resp_h, path, and name fields
2. Filter records where id.resp_h is a known file server, which reduces the results to potentially interesting connections
3. Review the path and name fields to identify which share the file was accessed from or written to, and determine if the behavior is suspicious.
4. For additional context about the remaining interesting records, you can pivot to the files log, using the UID to collect specific information about the file(s). For example, the MD5/SHA1/SHA256 hash(es) are automatically calculated and can be used to identify known malware in external systems, such as VirusTotal.
   a. There are also other fields and possibly logs available (e.g., pe log) that can be used to rule out uninteresting records.

```
path: smb_mapping,
uid: CiAtaM363GcEbU63zk,
id.orig_h: 192.168.38.104,
id.orig_p: 65431,
id.resp_h: 192.168.38.102,
id.resp_p: 445,
path: \\\\192.168.38.102\\C$,
share_type: DISK
```

```
path: smb_files,
uid: CiAtaM363GcEbU63zk,
id.orig_h: 192.168.38.104,
id.orig_p: 65431,
id.resp_h: 192.168.38.102,
id.resp_p: 445,
action: SMB::FILE_OPEN,
path: \\\\192.168.38.102\\C$,
name: Windows\\Temp\\hbaVJpzdnG,
size: 1894,
times.modified: 2019-12-31T10:28:02.800834Z,
times.accessed: 2019-12-31T10:28:02.753959Z,
times.created: 2019-12-31T10:28:02.566496Z,
times.changed: 2019-12-31T10:28:02.800834Z
```

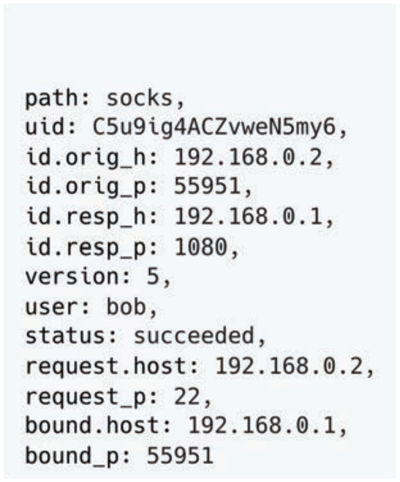
Non-Application Layer Protocol
Attackers often make use of a pair of techniques for hiding inside of legitimate traffic: sending their communications over a custom protocol on a commonly allowed port like 80, 443, or 53, and embedding their messaging inside of the structure of legitimate but typically less-monitored protocols like ICMP.

For the use of custom protocols on standard ports, see the Commonly Used Ports/Non-Standard Ports section for a description of Corelight's Encryption Detection package, the dpd log, and the weird log. These help you identify custom C2 communications that use non-standard encryption or violate traditional protocol specifications.

Malware sometimes employs standardized lower-level protocols like ICMP, UDP, and SOCKS to avoid detection as these protocols are rarely monitored. For example, malware authors might embed C2 instructions in an ICMP Echo Request ("ping") packet.

Corelight monitors all connections regardless of protocol, and stores connection data within the conn log. C2 channels that employ custom UDP protocols or TCP-based SOCKS protocols (but no standard application layer protocols) have conn log entries with no identifiable service field. These fields and logs provide visibility into traffic flows across the network — even ICMP, UDP, and SOCKS. For ICMP sessions, Corelight data contains more than just the source and destination, for example; packet counts, bytes transferred, and size of ICMP data for both the sender and recipient.

With this data, you have the information needed to discover abnormally large or frequent ICMP communications that can be indicative of C2. The log shown is a sample of the socks log.


```
path: socks,
uid: C5u9ig4ACZvweN5my6,
id.orig_h: 192.168.0.2,
id.orig_p: 55951,
id.resp_h: 192.168.0.1,
id.resp_p: 1080,
version: 5,
user: bob,
status: succeeded,
request.host: 192.168.0.2,
request_p: 22,
bound.host: 192.168.0.1,
bound_p: 55951
```

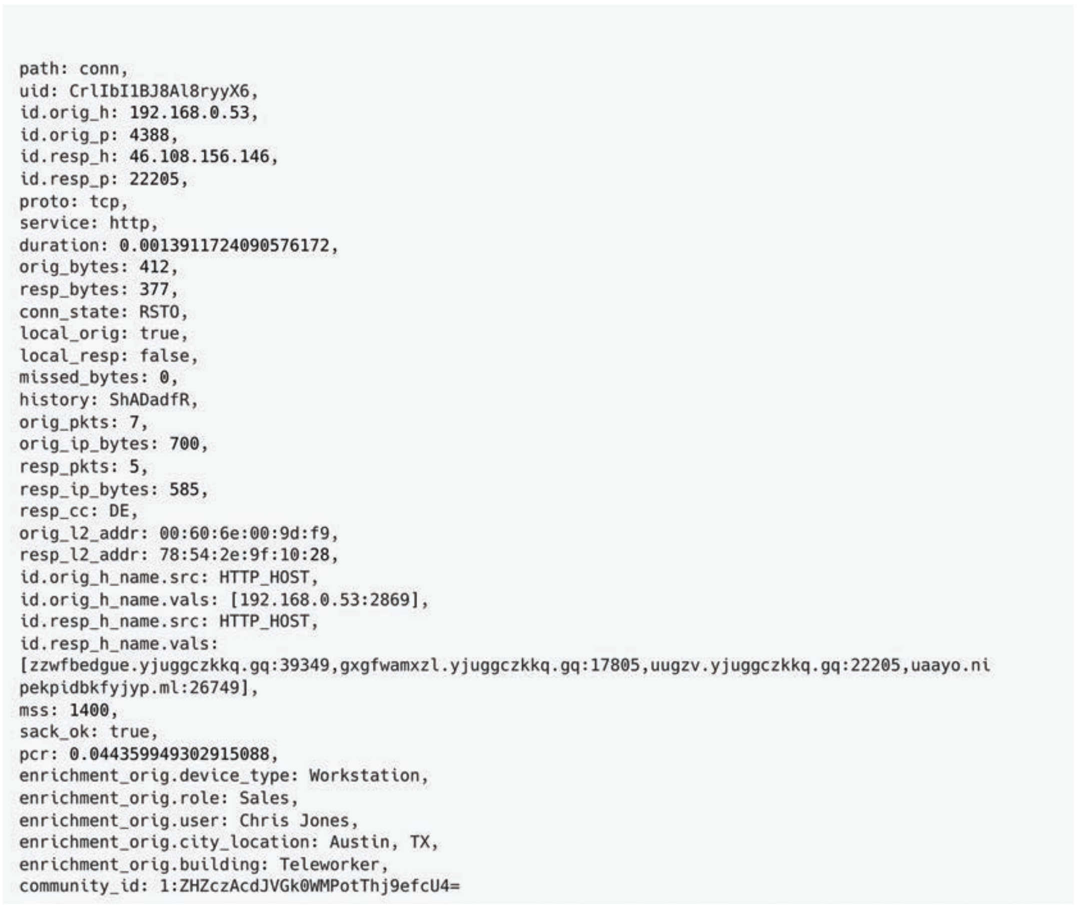
To hunt for an intruder using a standard non-application layer protocol to tunnel information:

1. Search the conn log for entries where the service field is blank, local_orig is true, and local_resp is false
2. Aggregate those results by id.orig_h, id.resp_h, id.resp_p and summarize by count
3. Filter 'normal' entries
4. Investigate any remaining items, focusing on the line items with the greatest count first

Non-Standard Ports
Every connection made in an environment monitored by Corelight is recorded in the conn log. After building a list of regularly used ports (e.g., 22/SSH, 25/SMTP, 80/HTTP, and 443/SSL), you can query the Corelight data to find connections to ports that aren't on that list.

If you encounter connections that appear on other non-standard ports, examine the Layer 7 service that Corelight observes and records in the conn log service field. Cases without a recognized service are the most suspicious, particularly if large volumes of data are being transferred or connection lengths are long.

When you encounter well-known services on irregular ports, examine the details in the corresponding protocol log for additional clues. For example, in the HTTP log, make note of the name of the remote host, the client's User-Agent string, and the URI. Together, they might all contain clues as to the software that's generating the request on the uncommon port.



```
path: conn,
uid: CrlIbI1BJ8Al8ryyX6,
id.orig_h: 192.168.0.53,
id.orig_p: 4388,
id.resp_h: 46.108.156.146,
id.resp_p: 22205,
proto: tcp,
service: http,
duration: 0.0013911724090576172,
orig_bytes: 412,
resp_bytes: 377,
conn_state: RSTO,
local_orig: true,
local_resp: false,
missed_bytes: 0,
history: ShADadfR,
orig_pkts: 7,
orig_ip_bytes: 700,
resp_pkts: 5,
resp_ip_bytes: 585,
resp_cc: DE,
orig_l2_addr: 00:60:6e:00:9d:f9,
resp_l2_addr: 78:54:2e:9f:10:28,
id.orig_h_name.src: HTTP_HOST,
id.orig_h_name.vals: [192.168.0.53:2869],
id.resp_h_name.src: HTTP_HOST,
id.resp_h_name.vals:
[zzwfbedgue.yjuggczkkq.gq:39349,gxgfwamxzl.yjuggczkkq.gq:17805,uugzv.yjuggczkkq.gq:22205,uaayo.ni
pekpidbkfyjyp.ml:26749],
mss: 1400,
sack_ok: true,
pcr: 0.044359949302915088,
enrichment_orig.device_type: Workstation,
enrichment_orig.role: Sales,
enrichment_orig.user: Chris Jones,
enrichment_orig.city_location: Austin, TX,
enrichment_orig.building: Teleworker,
community_id: 1:ZHZczAcdJVGk0WMPotThj9efcU4=
```

Proxy
While the use of proxies doesn't itself prove the presence of an intruder, intruders can use proxies to "launder" connections to obscure the communication from defenders. There are many methods to observe this, including traditional analysis of the underlying connection (signature, anomaly, behavioral) and statistical analysis of connection properties. Specifically identifying proxied connections is critical for beginning hunting or investigation.

If you see a value in the proxied field of Zeek's http log, that means an HTTP connection was proxied. The http log captures proxy details from the http headers. Search for any records in the http log that have a non-empty proxied field.

- host: the domain name of the website
- id.orig_h: the IP address of the proxy or reverse proxy
- id.resp_h: the IP address of the web server
- proxied: identifies the proxy and the original IP address of the client

For example, a client at IP 219.90.98.8 initiated this HTTP request. The request was proxied via 172.16.1.30 to the web server at 172.16.2.95.

host: www.totallyfakedomain.com
id.orig_h: 172.16.1.30 //the proxy
id.orig_p: 53,828
id.resp_h: 172.16.2.95 //the web server

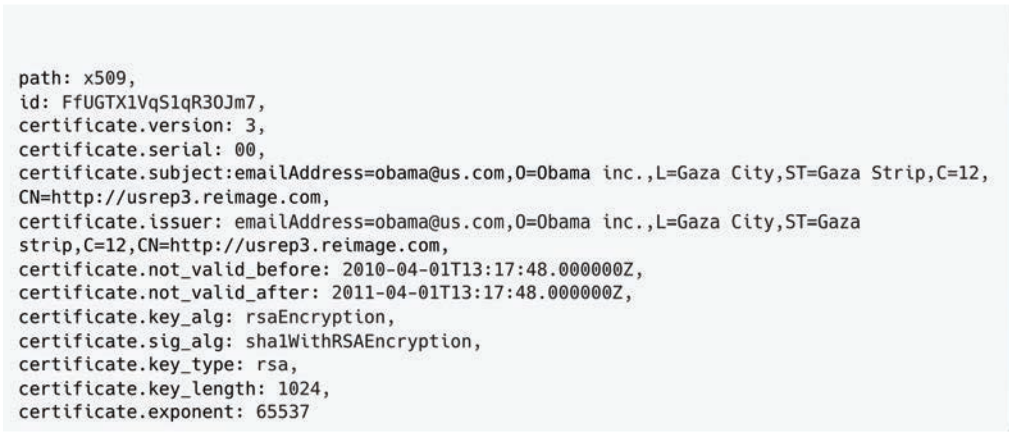
```
id.resp_p: 80
method: POST
post_body: dXNlcm5hbWU9cm9vdCZwYXNzd29yZD1tb25rZXk=
proxied: X-FORWARDED-FOR -> 219.90.98.8 //the real client
status_code: 200
status_msg: OK
uri: /xmlrpc.php
user_agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
log: http
```

Using this example, identify the proxy and determine whether it's internal or external. If it's external, evaluate the session and gain context, using Corelight data to decide whether or not to block it. If the proxy is internal, determine whether it's a legitimate piece of IT infrastructure, or if it is a rogue proxy set up to circumvent policy — shadow IT.

Additionally, SOCKS is a commonly used proxy protocol that Corelight Sensors natively parse. When SOCKS is encountered, a socks log is generated and records details on users and protocols. This information can be used to ensure that connections aren't malicious and comply with policy. In the socks log, focus on these fields:

- id.orig_h: the client IP address
- id.resp_h: the proxy IP address
- request: the domain or IP the client is attempting to access
- user: if it is an authenticated connection, the user using the proxy

Web Service
Web service is when attackers use a legitimate external web service to relay data to and/or from a compromised system. Attackers sometimes use well-known web services for C2 channels to hide in the noise. While this tactic makes identification more difficult, Corelight data — especially the http, ssl, conn, and x509 logs — helps you identify suspicious connections. Looking for IOCs including URI, hostname, or specific certificate details (like SNI or CN) is a good place to start. The following provides a few examples of certificate fields that might warrant an investigation:



```
path: x509,
id: FfUGTX1VqS1qR3OJm7,
certificate.version: 3,
certificate.serial: 00,
certificate.subject:emailAddress=obama@us.com,O=Obama inc.,L=Gaza City,ST=Gaza Strip,C=12,
CN=http://usrep3.reimage.com,
certificate.issuer: emailAddress=obama@us.com,O=Obama inc.,L=Gaza City,ST=Gaza
strip,C=12,CN=http://usrep3.reimage.com,
certificate.not_valid_before: 2010-04-01T13:17:48.000000Z,
certificate.not_valid_after: 2011-04-01T13:17:48.000000Z,
certificate.key_alg: rsaEncryption,
certificate.sig_alg: sha1WithRSAEncryption,
certificate.key_type: rsa,
certificate.key_length: 1024,
certificate.exponent: 65537
```

EXFILTRATION

Automated Exfiltration
If an attacker is using an automated means of exfiltration, data artifacts are captured in the Corelight data.

To look for exfiltration in your network, you can use the Zeek package developed to calculate Producer/Consumer Ratio (PCR). PCR values indicate whether flows are consumptive (download) versus productive (upload). PCR values range from -1 (consumptive) to +1 (productive). To hunt for exfiltration using this package:

1. Install and enable the PCR package.
2. Generate a table of id.orig_h, id.resp_h, id.resp_p, and pcr from the conn log.
3. Use local_orig is false or local_resp is true to filter the results.
4. Reduce the results by filtering where pcr <= 0.
5. For each host generating flows where pcr >= 0, consider whether that host is expected to transmit data, inside or outside the network.

Another option is to use a SIEM to calculate the PCR using the information available in the Corelight conn log. The following query creates a table organized by host that contains the originating and responding bytes and a PCR value.

```
index=corelight sourcetype=corelight_conn | stats sum(orig_bytes) as Total_orig_bytes, sum(resp_bytes) as Total_resp_
bytes by id.orig_h id.resp_h | eval PCR=(Total_orig_bytes-Total_resp_bytes)/(Total_orig_bytes+Total_resp_bytes) | fields
id.orig_h id.resp_h Total_orig_bytes Total_resp_bytes PCR
```

Data Transfer Size Limits
An attacker may attempt to transfer data or files by "chunking" them into smaller pieces, to avoid hard-coded data transfer limits or thresholds. We will present two methods to hunt for this technique.

The first method analyzes data leaving the network based on source and destination pairs and requires a data aggregation/visualization platform (unless you enjoy AWKing and GREPing through data):

1. Generate a table from the conn log including the id.orig_h, id.resp_h, id.resp_p, and sum(orig_bytes).
2. Sort the results by the largest sum (orig_bytes).
3. Examine each host and determine if there is a legitimate reason for uploads to that destination.

The second method analyzes the frequency, and sizes, of outbound transfers from each source:

1. Generate a table from the conn log including id.orig_h, id.resp_h, id.resp_p, and count(orig_bytes).
2. Sort the results by the largest count(orig_bytes).
3. Examine the results and determine the reason for all the connections with the same amount of data flowing from the source to the destination.

REFERENCES

1. https://attack.mitre.org
2. When used as an intel indicator IP is considered brittle, due to the ease with which adversaries can move to a new host or provider.
3. Not all versions of RDP assert the username in the cookie field. Some just assert nothing, or gibberish. In those instances, you would have to infer it from the NTLM or Kerberos log.
4. https://www.wired.com/story/untold-story-2018-olympics-destroyer-cyberattack/
5. Please visit https://packages.zeek.org/ for additional information about Zeek packages

CORELIGHT OPEN NETWORK DETECTION & RESPONSE PLATFORM

Get a demo

Easily deployed and available in on-prem and SaaS-based formats, Corelight combines the power of open source and proprietary technologies to deliver a complete Open Network Detection & Response (NDR) Platform that includes intrusion detection (IDS), network security monitoring, and Smart PCAP solutions.

https://corelight.com/products/demo

corelight

Corelight provides security teams with network evidence so they can protect the world's most critical organizations and companies. Our Open Network Detection and Response Platform enhances visibility and analytics, leading to faster investigations and expanded threat hunting. Corelight's global customers include Fortune 500 companies, major government agencies, and large research universities. Based in San Francisco, Corelight is an open-core security company founded by the creators of Zeek®, the widely-used network security technology.

info@corelight.com | 888-547-9497