Onion-Zeek-RITA: Improving Network Visibility and Detecting C2 Activity

Dallas Haselhorst
Onion-Zeek-RITA: Improving Network Visibility and Detecting C2 Activity

GIAC (GCIA) Certification

Author: Dallas Haselhorst
Email: dallas@treetopsecurity.com / Twitter: @oneoffdallas
Advisor: David Hoelzer
Accepted: 2 January 2019

Abstract

The information security industry is predicted to exceed 100 billion dollars in the next few years. Despite the dollars invested, breaches continue to dominate the headlines. Despite best efforts, all attempts to keep the enemies at the gates have ultimately failed. Meanwhile, attacker dwell times on compromised systems and networks remain absurdly high. Traditional defenses fall short in detecting post-compromise activity even when properly configured and monitored. Prevention must remain a top priority, but every security plan must also include hunting for threats after the initial compromise. High price tags often accompany quality solutions, yet tools such as Security Onion, Zeek (Bro), and RITA require little more than time and skill. With these freely available tools, organizations can effectively detect advanced threats including real-world command and control frameworks.
1. Introduction

The mean time to identify (MTTI) a breach, often referred to as dwell time, is defined as “the period between when a malicious attack enters your network and when it is discovered” (Mallon, 2017). Measuring MTTI varies from one reporting agency to the next. Even so, the general consensus is that dwell time has either remained mostly unchanged or has even decreased from one year to the next. Improvements are always welcome, however, the 2017 Cost of a Data Breach Report from the Ponemon Institute shows the average attacker dwell time remains over 190 days (Ponemon Institute, 2018).

Infiltrating a network means very little without a method for the attacker to interact with a compromised host. In most cases, the infected host communicates with a command and control server or C2. The C2 channel is the communication conduit between the compromised system(s) and the attacker’s C2 server. The compromised host periodically checks with the C2 server to carry out the next attacker-directed task. These instructions can include a wide range of malicious activities such as ransomware, exfiltrating data, or denial of service (DoS). To avoid triggering any alarms, attackers attempt to bypass traditional defenses en route to gaining a system foothold. It is no surprise that along with a lengthy dwell time, 44% of threats go undetected by standard tools (Crowd Research Partners, 2017). Security Operation Centers (SOCs) also recognize new and unknown threats as difficult to detect (Figure 1).

![Figure 1: Detection & dwell time](Image source - Security On-Demand)

<table>
<thead>
<tr>
<th>They get in</th>
<th>They expand and hide</th>
<th>They are tough to detect</th>
</tr>
</thead>
<tbody>
<tr>
<td>44%</td>
<td>6 MONTHS</td>
<td>70%</td>
</tr>
</tbody>
</table>

of Threats go undetected by automated security tools¹
Average time attackers dwell on networks until discovered
of SOCs top challenge is detection of hidden, unknown, and emerging threats²

---

Dallas Haselhorst, dallas@treetopsecurity.com

---

© 2019 The SANS Institute

Author retains full rights.
Several factors contribute to these statistics. First, organizations incorrectly view security as a problem to be solved. Instead, security should be identified as a process and not a destination. Even with an endless security budget, an organization cannot achieve total security. Investing in meaningful prevention will always provide improvements to security, but a zero-attack surface is an impossible goal. Furthermore, if an organization matched its needs with ideal solutions, security is a diminishing return. Thus, a small investment in an immature security program may significantly improve the security posture of an organization while the same investment in a mature security program may provide little impact.

Attacks are not entirely preventable, but where prevention fails, detection prevails. Prevention is often another “black box” plugged into the datacenter promising to end security woes. Detection is viewed similarly, although to a lesser degree. The truth is that neither of them truly fit the “set and forget” mold. Proper detection requires people and continuously improving processes, i.e., perceptive results require perceptive individuals. In his book, Good to Great, Jim Collins best described this fallacy when he said, “thoughtless reliance on technology is a liability” (p. 159). This concept is no different in information security. The information security field may even compound this idea by requiring more talent to run more tools effectively. At a fundamental level, products can and should augment talent, not replace it. This concept is the opposite of traditional management philosophies which view product costs as offsets or justifications for reductions in full-time employees (FTEs).

Another contributing factor to the detection and high dwell time statistics is the lack of logging. Logging should be considered a crucial component to enhancing visibility and detection. Instead, logging often receives a negative stigma because of its alignment with compliance requirements. Compliance does not equal security. Likewise, logging alone does nothing to improve security. An endless number of logged security events provide little to no benefit without follow-up. Security only improves through logging, careful analysis of the collected data, followed by action. When properly scrutinized, logging has real-world benefits and is much more than a checkbox for regulatory standards.

2. Basic Detection

Dallas Haselhorst, dallas@treetopsecurity.com
A common misconception in information security is that attackers only need to find one vulnerability to succeed and defenders need to ensure there are none. This statement is only valid when defenders are not looking for significant security events. If a security-related event is missed, it is either a security blind spot or it is one relevant event in a sea of false positives. Regardless of the reason, the attackers only need to find one way in if they attack where defenders are not actively monitoring. Even the most basic prevention tools can augment detection efforts, however, the issue is that most system administrators are not looking at these tools or logs for security-related activity.

One such prevention tool is a firewall with properly configured egress filtering. Egress filtering can generate logs when an attacker tries to communicate outbound on blocked ports. Although the attacker or malware may eventually find an open port, malware may test several outbound ports during the initial compromise. In Figure 2 below, malware triggered multiple firewall blocks when attempting to communicate with an external C2 server. The malware in this example communicates over port 443, the standard port used for HTTPS traffic. The firewall blocked this communication because this environment uses a web proxy. Thus, secure web traffic communicates with the proxy over an entirely different port. Simply stated, this particular malware was not proxy-aware.

![Figure 2: Firewall egress blocks](image)

Correlating security events with alerts from other fundamental prevention tools such as endpoint protection can significantly enhance the overall security of an environment. Endpoint protection is under-valued in this regard and is instead considered an automated, configure-once security tool. Many administrators believe if the endpoint protection blocks malware from executing, that is where its usefulness ends. Endpoint protection can provide much more and endpoint notifications can even indicate an attack is underway. Similar to egress blocking, attackers may make several attempts to bypass endpoint protection or disable the endpoint
protection before succeeding in their endeavor. Without any modifications to the default C2 configuration, the attack in Figure 3 failed because endpoint protection blocked it from execution. After the initial attack fails, an attacker may modify the C2 configuration to bypass known endpoint protection signatures.

![Trojan:Win32/Skeezyah.A!rfn](image)

**Figure 3:** Windows Defender detecting a default PoshC2 payload

### 3. More Advanced Tools

As discussed, even standard prevention tools can enrich detection capabilities. In most instances, however, those techniques only work for commodity malware or attackers unfamiliar with an environment. The value of data from traditional defenses is undeniable, yet more sophisticated attackers and malware could bypass those defenses with little effort. A security-conscious organization would require tools and techniques beyond signature-based, defensive capabilities.

One tool examined to enhance the detection of security events is Security Onion. From the website for the distribution, “Security Onion is a free and open source Linux distribution for intrusion detection, enterprise security monitoring, and log management” (Security Onion, 2018). Security Onion is essentially a Network Security Monitor (NSM) with a wide range of capabilities in a single, easily-deployed package. Installation instructions and the latest ISO image are on the Security Onion GitHub page (Installation, 2018). The capabilities of Security Onion include packet capturing, intrusion detection, logging, and numerous analysis tools. Once integrated into an environment, the distribution becomes a “swiss-army knife” for threat hunting.
and traffic analysis. There are numerous ways to get data into Security Onion, but this research focused primarily on log collection and full packet captures. Elastic Stack (formerly ELK stack or Elasticsearch, Logstash, and Kibana) and Zeek (formerly Bro) analyzed the collected data along with other Security Onion tools.

RITA (Real Intelligence Threat Analysis), a tool not installed by default with Security Onion, was added to the lab setup. The installation of RITA is straightforward with the help of the guide on the Security Onion GitHub page (RITA, 2018). RITA is a threat hunting framework that ingests Zeek logs. In turn, RITA uses statistical analysis and the k-means clustering algorithm to aid in searching logs for indicators of compromise. The framework excels at identifying network abnormalities such as long connections and beaconing activity. The beaconing module uses Discrete Fast Fourier Transform (DFFT) to analyze connection logs. DFFT can identify connection frequency to show how often communication occurs (Strand, 2015). This type of traffic analysis is useful even if the traffic is encrypted.

Another great feature of RITA is the ability to compare older, collected data against newly discovered blacklisted IP addresses and URLs. For example, a compromised system might have previously communicated with an internet-based IP address. No alerts were triggered because the IP address was thought to be benign last week. This week, however, that same IP address is now on five different threat intelligence feeds because it was used for botnet activity and distributing malware. When combined with Security Onion and Zeek, RITA is an effective addition to any threat hunting toolkit.

4. Lab Setup

The lab environment for this testing resided on a network segment separate from all production traffic. The end user system was a clean Windows 10 Professional install with the latest Windows Updates applied. The Security Onion distribution was installed on a physical server with two network interfaces. One network interface was used for system management and accessing the web interface for queries. The second interface on the Security Onion server was used solely for full packet captures. Packet captures were configured via port mirroring on a managed switch. The port mirror duplicated all traffic from the end user system to the packet...
capture interface on Security Onion. Security Onion also received firewall logs via syslog to potentially assist with event correlation. The diagram seen in Figure 4 below shows the general lab configuration including installed applications and major components. The image also depicts the relative location of the end user to the attacker and the internet.

![Lab network diagram](image)

**Figure 4: Lab network diagram**

The network segmentation ensured outside activity would not taint collected data. Segmentation inevitably made detection easier, but gaining some familiarity with the environment proved to be a valuable first step before introducing the C2 traffic. Not unlike a standard production environment, understanding “normal” is critical to identifying abnormal activity and anomalous behavior. This observation period also allowed time to ensure the necessary features of Security Onion were working correctly.

### 5. Post Setup
Zeek immediately started analyzing the captured network packets and generating activity data. In addition to providing simple aggregation data such as SSL and HTTP connection counts (Figure 5), Security Onion also offers domain name frequency analysis. Though this feature was not used for detecting these C2 channels in a lab environment, malware often uses randomly generated domain names. The malware will then change to a new random name at predetermined dates and times. The domain frequency analysis feature could prove extremely useful when hunting for that type of activity. As shown below in Figure 6, a random domain name was easily identified using this method.

![Figure 5: HTTP log count](image1)

![Figure 6: Domain name frequency analysis](image2)

The overall goal was to determine if these more advanced detection tools could effectively identify malware widely considered to be difficult to detect when traditional defense mechanisms fail. PoshC2 and PowerShell Empire are both PowerShell exploitation frameworks. The Windows operating system has PowerShell built-in and both frameworks rely on it extensively. By using PowerShell, detection is somewhat more problematic because the
frameworks are “living off the land” where additional software components are not necessary for either of the C2s to work.

Both PoshC2 and PowerShell Empire employ numerous techniques to help evade detection. One method of evasion is encrypting all network communications. Encryption makes on-the-wire detection exponentially more difficult because the network traffic is not readable. Thus, many of the signatures found in standard intrusion detection systems (IDS) such as Snort or Suricata never trigger. Also, both frameworks employ jitter by default. Jitter randomizes the interval which the compromised host communicates back to the C2 server. Instead of communicating every three minutes, the system may communicate at two minutes and thirty-four seconds followed by four minutes and one second the next time.

These C2 frameworks can use encryption although attackers may fail to configure a valid SSL certificate from a certificate authority. By default, both exploit frameworks use self-signed certificates. Although Zeek is unable to decrypt the traffic, it can uncover this activity relatively easily. In Zeek notices, self-signed server certificates are easily recognized under the “SSL::Invalid_Server_Cert” notice type. Zeek will also display an “SSL Certificate validation failed with (unable to get local issuer certificate)” message to further explain why the certificate was invalid.

![Figure 7: Invalid SSL certificates](image)

When a user visits a website, their browser provides a signature to the web server they are visiting. This signature, called a user agent string, helps the server determine relevant content and client capabilities. C2 channels may spoof this string in an attempt to blend in with normal traffic (See Appendix A). In many cases though, the user agent string is unchanged from the C2 default values. Much like other C2 defaults, looking for the default user agent string can be extremely helpful in hunting. An organization can further differentiate from the C2 user agent string by designating only one, accepted browser type. Additionally, the organization could use a
custom user agent string for the entire organization. The custom user agent string could then be pushed out via group policy for easier management (Set Chrome, 2018). Hunt teams could easily search for non-standard or disallowed user agent strings since the C2 would likely be unaware of these site-wide policies.

![Figure 8: C2 unchanged user agent strings](image)

Domain name frequency, custom user-agent strings, and invalid SSL certificate tests can work for a majority of attacks. However, these analysis methods could potentially fail if used in a highly targeted attack. With additional time and planning, a sophisticated attacker could take extra measures to prevent C2 detection. Using SSL certificates from a trusted certificate authority would clear up any SSL-related certificate errors; even a free SSL certificate from Let’s Encrypt would satisfy this need. A valid SSL certificate further blends the encrypted C2 traffic in with other encrypted traffic. Custom C2 configurations could also bypass standard endpoint protection and intrusion detection signatures. The default settings for PowerShell Empire C2 are well-known and defenders can tune their systems to look for this activity (Long, 2017).

However, what if these C2 channels are configured with non-standard options? Can the advanced detection tools still aid in detecting these C2 channels?

### 6. C2 Traffic Analysis

The two PowerShell C2 frameworks share numerous similarities. Surprisingly, they also have several differentiating characteristics. The most significant difference is their underlying methods of communication. This variation leads to two distinct hunting methodologies.

One difference is that PowerShell Empire opens and closes TCP connections for its C2 communications. As a result, the beaconing traffic has constantly incrementing source ports. PowerShell Empire allows for changing the landing page for beacons, although this would not
change analysis from the network perspective. After filtering traffic by source and destination IP address, the beaconing activity is easily recognizable. In Figure 9 below, the source port increments by one and the beacons occur every five seconds. The source ports change less consistently if there is additional C2 activity or other TCP communications occur between the hosts.

![Figure 9: PowerShell Empire beacons in Zeek](image)

If the initial IP addresses of the systems were not known, another hunting technique to find PowerShell Empire beaconing activity is via the Zeek connections data. Constantly opening and closing TCP connections creates excessive connections by both the source and destination IP addresses. Unless the beacon is actively transmitting data over the C2 channel, the connection size would also stay the same as shown in Figure 10.
Another method of finding PowerShell Empire beaconing activity is with RITA. After creating the dataset from Zeek data (See Appendix B), RITA successfully identified the C2 activity using different approaches. The first command shows the most visited URLs in the dataset. Due to the excessive number of connections, the three most visited URLs were all related to PowerShell Empire C2 activity (Figure 11). The RITA command in Figure 12, show-beacons, provides a score indicating the likelihood of whether the analyzed traffic is beaconing. RITA can also output to an HTML format for improved readability (Figure 13).

```
$ rita show-most-visited-urls dataset1
URL,URI,Length,Times Visted,IPs
[evil IP],/admin/get.php,24,50,[evil IP]
[evil IP],/news.php,19,28,[evil IP]
[evil IP],/login/process.php,28,28,[evil IP]
```

```
$ rita show-beacons dataset1
Score,Source,Destination,Connections,Avg Bytes,TS Range,DS Range,TS Mode,DS Mode,TS Mode Count,DS Mode Count,TS Skew,DS Skew,TS Dispersion,DS Dispersion,TS Duration
0.894783,192.168.251.53,[evil IP],105,2502.12,5,849,5,413,94,48,0,0,0,4,0.5
```

Figure 10: Excessive same-sized connections

Figure 11: RITA most visited URLs

Figure 12: RITA beacons

Figure 13: RITA beacons - HTML output

Dallas Haselhorst, dallas@treetopsecurity.com
Instead of opening and closing TCP connections as seen in PowerShell Empire, PoshC2 opens a TCP connection and leaves the connection open. Thus, all beaconing traffic communicates over the same source port. The “beacon” appears as a single TCP connection. If the endpoint can capture the traffic, the activity is easy to recognize. In the Wireshark capture found in Figure 14 below, all beacons use the source port of 1653 and ACKs for traffic. The activity displays in the Zeek connection logs once the connection closes (Figure 15).

![Figure 14: PoshC2 activity in Wireshark](image)

The open connections pose a problem for several tools since Zeek does not write to the connection log until the TCP connection closes. Thus, any tool that relies on the connection logs cannot perform analysis as it is unaware the connection exists. The display packet capture tool, capME, gives an error stating the “Second ES query could not

![Figure 15: PoshC2 connection log](image)

![Figure 16: capME open PoshC2 TCP connection](image)
find this ID” as seen in Figure 16. Fortunately, there are Zeek add-ons to assist with detection of these long connections although these scripts and packages are not installed by default (Long Connections, 2017).

If the connection is closed, finding long connections becomes somewhat trivial. As shown in Figure 17 below, a Kibana query with the “max duration” column allows for sorting based on connection times. RITA does not explicitly identify the activity as beaconing because of the single TCP connection (See Appendix C). However, the show-long-connections functionality of RITA is ideally suited for finding this type of C2 activity.

![Figure 17: Kibana and max duration to detect PoshC2](image)

### 7. Potential Issues

Numerous tools identified the beaconing activity found in PowerShell Empire. RITA, for example, would undoubtedly still find the C2 activity even if the attacker increased the beacon interval. It is important to keep in mind that with a high enough beacon interval, RITA might require substantially more data – gathered across days, weeks, and months – in order to identify beaconing activity. This change would likely decrease the usefulness of the C2 to the attacker unless maintaining access for future activity was the primary focus. Also, if PowerShell Empire intermittently went dormant, this could circumvent some of the statistical analysis occurring in the background.
PoshC2 uses HTTP/2 for communications as shown in Figure 18 from capME. Similar to other HTTP/2 traffic, this leaves a single TCP connection open and all subsequent communications occur over the initial connection. Aside from the issue with open connections, HTTP/2 is a binary protocol instead of plain-text as found in the prior standard, HTTP 1.1. Thus, HTTP/2, by its very nature, is not human-readable (HTTP/2 FAQ, 2018). These changes along with security improvements in Version 1.3 of the TLS protocol (Fruhlinger, 2018) could make the SOC analyst role far more difficult for even the most seasoned professional.

Identifying HTTP/2 traffic related to PoshC2 in the lab or even a production network might not be overly difficult today. However, the global shift to HTTP/2 is unquestionably going to make detection far more problematic. An analyst or hunt team can investigate a handful of long connections, but this practice is not feasible as more connections shift to the new standard. Also, minor changes to the C2 configuration could hinder the ability to locate the long TCP data streams. These changes could help the C2 traffic blend in with standard HTTP/2 traffic. Similar to PowerShell Empire C2, an attacker may choose to keep a low profile on the system after the initial compromise. Modifications to beaconing intervals and jitter could also stifle hunt team efforts if combined with the turtle feature already built into PoshC2.

8. Conclusion

Although often ignored, basic security tools can augment the detection of C2 activity. Despite their usefulness, these tools alone cannot keep environments secure. Security Onion and

Dallas Haselhorst, dallas@treetopsecurity.com
Onion-Zeek-RITA

Zeek excel in aggregating, presenting, and correlating network data. RITA further enhances these capabilities by providing an entirely different level of analysis. These tools can supplement current defensive toolsets. More importantly, these freely available utilities in the hands of skilled defenders can significantly improve the odds for the blue team.

Attackers change their tactics only to find that defenders are a step ahead of them. Defenders improve only to find the attackers have shifted the battle to a new front. Attackers can often evade detection by avoiding common mistakes and configuration defaults. Still, the core tenets for successful detection have not changed for even the most skilled adversaries. Security Onion, Zeek, and RITA can enhance the ability to analyze the number of connections, the length of connections, and the amount of data. Thus, even though the game has changed significantly, these same tenets will remain crucial to hunting both now and in the future.

It is only a matter of time before the capabilities found in these tools become the new baseline for every organization to stay secure. Even then, no security tool is a fix-all and the combination of these tools should not be mistaken as a panacea. Understanding how these tools work and how they can help detect real-world command and control is important. Perhaps equally important is understanding their limitations such as where gaps still exist or even where new gaps might develop. The game of cat-and-mouse between attackers and defenders will continue. Breaches will undoubtedly continue. Defenders must stay up-to-date on current trends, current toolsets, and follow ever-changing best practices to protect the data we value most.
References


Long Connections (2017, September 29). Retrieved December 10, 2018 from
https://github.com/corelight/bro-long-connections


Nettitude. PoshC2 Overview. Retrieved October 2, 2018 from


https://securityonion.net/#about

Set Chrome Browser Policies on Managed PCs. Retrieved October 10, 2018 from
https://support.google.com/chrome/a/answer/187202?hl=en

Appendix A
User Agent Strings Comparison

**PowerShell Empire (Default)**
Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko

**PoshC2 (Default)**
Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; Touch; rv:11.0) like Gecko

**Windows 10 - Internet Explorer**
Mozilla/5.0 (Windows NT 10.0; AGB45.8; Trident/7.0; MALNJS; rv:11.0) like Gecko

**Windows 10 - Microsoft Edge**
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134

**Windows 7 - Mozilla Firefox**
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:63.0) Gecko/20100101 Firefox/63.0

**Android Tablet - Firefox**
Mozilla/5.0 (Android 4.4; Mobile; rv:41.0) Gecko/41.0 Firefox/41.0

**Mac OS X - Google Chrome**
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_1) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/70.0.3538.77 Safari/537.36
Appendix B
RITA Commands - PowerShell Empire

$ rita import /nsm/bro/logs/current dataset1
[+] Importing /nsm/bro/logs/current
  [-] Finding files to parse
  [-] Parsing /nsm/bro/logs/current/dns.log -> dataset1
  [-] Parsing /nsm/bro/logs/current/conn.log -> dataset1
  [-] Parsing /nsm/bro/logs/current/http.log -> dataset1
  [-] Indexing log entries. This may take a while.

$ rita analyze
[+] Analyzing:
  [-] dataset1
[+] Analyzing dataset1
  [-] Running Unique Connections Analysis
  [-] Running Unique Hosts Analysis
  [-] Running Unique Hostnames Analysis
  [-] Running Exploded DNS Analysis
  [-] Running URL Length Analysis
  [-] Running User Agent Analysis
  [-] Running Blacklisted Analysis
  [-] Running Beaconing Analysis
  [-] Running Scanning Analysis
  [-] Running Cross Reference Analysis

$ rita show-most-visited-urls dataset1
URL,URI,Length,Times Visted,IPs
[evil IP],/admin/get.php,24,50,[evil IP]
[evil IP],/news.php,19,28,[evil IP]
[evil IP],/login/process.php,28,28,[evil IP]

$ rita show-beacons dataset1
Score,Source,Destination,Connections,Avg Bytes,TS Range,DS Range,TS Mode,DS Mode,TS Mode Count,DS Mode Count,TS Skew,DS Skew,TS Dispersion,DS Dispersion,TS Duration
0.894783,192.168.251.53,[evil IP],105,2502.12,5,849,5,413,94,48,0,0,4,0,0.5
0.111111,14,4,0.999396

$ rita show-beacons pse-modified
Score,Source,Destination,Connections,Avg Bytes,TS Range,DS Range,TS Mode,DS Mode,TS Mode Count,DS Mode Count,TS Skew,DS Skew,TS Dispersion,DS Dispersion,TS Duration
0.881719,192.168.251.53,[evil IP],2679,2094.84,56,52,51,413,64,918,0,-0.111111,14,4,0.999396
Appendix C
RITA Commands – PoshC2

$ rita import /nsm/bro/logs/current dataset2
[+] Importing /nsm/bro/logs/current
[-] Finding files to parse
[-] Parsing /nsm/bro/logs/current/dns.log -> dataset2
[-] Parsing /nsm/bro/logs/current/conn.log -> dataset2
[-] Parsing /nsm/bro/logs/current/http.log -> dataset2
[-] Indexing log entries. This may take a while.

$ rita analyze
[+] Analyzing:
[-] dataset2
[+] Analyzing dataset2
[-] Running Unique Connections Analysis
[-] Running Unique Hosts Analysis
[-] Running Unique Hostnames Analysis
[-] Running Exploded DNS Analysis
[-] Running URL Length Analysis
[-] Running User Agent Analysis
[-] Running Blacklisted Analysis
[-] Running Beaconing Analysis
[-] Running Scanning Analysis
[-] Running Cross Reference Analysis

$ rita show-beacons dataset2
Score, Source, Destination, Connections, Avg Bytes, TS Range, DS Range, TS Mode, DS Mode, TS Mode Count, DS Mode Count, TS Skew, DS Skew, TS Dispersion, DS Dispersion, TS Duration

$ rita show-long-connections dataset2
Source IP, Source Port, Destination IP, Destination Port, Duration, Protocol
192.168.251.53, 2743, [evil IP], 443, 2894.18, tcp
... other data ...