StealthWare - Social Engineering Malware

Joey Dreijer [1]

Abstract
This research focuses on testing different social engineering malware frameworks on their practical use and to find detection methods. By analyzing network traffic and behaviour of existing tools we were able to create signatures for Intrusion Detection System to detect the use of social engineering malware on a network. Furthermore, different theoretical aspects of network monitoring were discussed to provide insight in signature-less detection of advanced social engineering malware. Finally, an advanced proof-of-concept malware client was written to bypass basic network policies and evade basic detection rules. This tool can be used for testing network restrictions and advanced detection rules in real-life.

Keywords
Social Engineering — Malware — Detection — Security

[1] *Student System and Network Engineering, University of Amsterdam, the Netherlands*

Contents

1. Introduction

Pentesters, security professionals and grey/black-hats often use software beacons (ie. command and control malware) for social engineering assignments. Security professionals use these beacons to identify possible security vulnerabilities from inside the network. These moles are often deployed via methods such as social engineering or spearphising; sending e-mails with malicious attachments or handing out specialized USB-sticks. The development of specialized malware for social engineering assignments is not done on a large scale and the current frameworks offer limited functionality (for the researched use-case) and/or ask for a substantial amount of licensing costs. During my research I investigated the effectiveness of current social engineering malware and developed my own proof of concept client (based on an existing framework). This research focuses on the usability within company environments and the detectability when using commonly found security software/hardware such as firewalls, network authentication, intrusion detection systems and virus-scanners.

1.1 Research question

Is it possible to stealthy use social-engineering malware for security assessments?

- What existing frameworks already exists and what functionality do they offer?
- What common security policies are often found within company networks?
- How can these common security policies be bypassed?
- How effective do the already existing frameworks work in realistic company environments?
- Can these frameworks be further extended and/or improved to minimalize detection chances for advanced security audits?

1.2 Scope

This research focussed on investigating the usefulness of already existing social engineering malware frameworks and possibilities to find possible detection methods. Furthermore, attempts were made to further improve the existing frameworks to demonstrate the use of specialized social engineering malware that can be deployed in company networks. This report also contains a proof-of-concept design involving new evasive techniques can best be used to make these frameworks as stealthy as possible on a network/-transport level. Three existing frameworks that were tested during this research:

- Cobalt Strike by Strategic Cyber LLC - www.advanced-pentest.com;
- MetaSploit (Interpreter) by Offensive Security - www.off-ensive-security.com;
- ThrowBack by Brady Bloxham (SilentBreak)- github.com/-silentbreaksec/Throwback.

Cobalt Strike and Throwback make use of their own unique beaconing mechanisms. Besides that, both frameworks can use Metasploit to set up an interactive (Metasploit) meterpreter session to interact with the infected client. More about specific functions of these frameworks will be discussed in section 3.4. There are different methods how malware is injected on a client during social engineering assignments. Some common scenarios involve sending a phishing e-mail with malicious attachment or giving away promotional USB sticks that automatically execute a malicious file. This research will focus on the malware itself and only at a minimal level about exploitation (ie. how the malware is executed by a client). The hosts that are being infected are all Windows-based.

2. Break-down Social Engineering malware attack

There are two publicly available social engineering malware Frameworks; Cobalt Strike and ThrowBack. Cobalt Strike [1] is a commercial penetration testing tool to replicate advanced threats. Cobalt Strike offers advanced/additional functionality on top of Metasploit. Cobalt Strike offers an easy to use interface to interact with Metasploit, deploy custom payloads and act as a command and control server for beacons. A trial version (valid for 21 days) was used during this research which includes identical functions as that of the licensed version. For this research, only the beaconing / command and control functionality was tested. ThrowBack is a new Open Source framework to provide beaconing functionality with a command and control server. ThrowBack was first presented during the DefCon Conference in 2014 [2] and contains many default functions that are also implemented in the Cobalt Strike beacons; executing commands, interactive shell via a Metasploit, changing the communication time-outs. Figure two displays a basic example of a Cobalt Strike beacon deployment scenario.

1. The attacker crafts a social engineering email via the Cobalt Strike client. The email contains malware that should provide control over the infected client
2. Once the client is infected, a reverse connection is opened with the Metasploit server.
3. The attacker can gain access to the client via the Metasploit meterpreter. The Cobalt Strike application interacts with the Metasploit server to get an interactive shell on the infected client.
4. When configured, the client can also communicate (without a connection to Metasploit) with the Cobalt Strike via beacons. These beacons are periodic requests that do not have a continues active session opened. Cobalt Strike can 'queue' commands that are executed by the client the next time a beacon is sent/received.

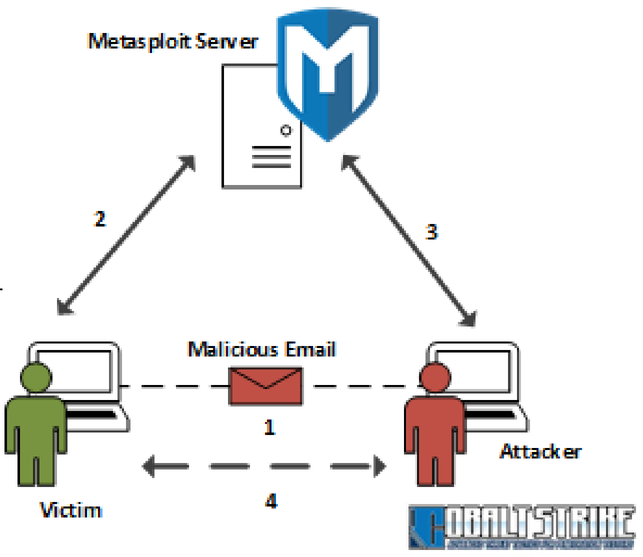


Figure 1. Example of Cobalt Strike infection and control. **Note:** The number displayed in the image do not necessarily imply the order of activity or execution.

The goal of social engineering malware is to gain control over a specific target and to maintain control over an extended amount of time. Social Engineering frameworks make use of these periodic beacons (instead of a continuous open connection) to be as stealthy as possible. However, even these beacons can be detected over time. Sections 3.5 and 3.6 will display how Cobalt Strike and ThrowBack beacons can be detected via non-complex detection rules.

2.1 Beaconing

The term "Beacons" has been mentioned multiple times in this report already. But what exactly is beacon traffic? Beacons are periodic signals an infected client sends to a command and control server. The command and control server will receive this signal and report that a new infected client is available. These beacons not only produce a method of indicating client uptime and activity, but also provide a mechanism to execute new commands on the infected client.

When an attacker infects a client, the client unknowingly runs a process that sends periodic signals to a command and control server. Cobalt Strike offers the ability to send these beacons via regular HTTP(S) requests, DNS requests and SMB traffic. At the date this report was written, Throwback only offers beaconing over Hypertext transport protocol secure (HTTPS). Since ThrowBack is an open framework, additional transport methods can be implemented by the community.

An infected client sends a new beacon periodically (the exact time is configured by the malware operator) to verify if a new task is available. The operator can add new tasks to a command queue hosted on the command and control server. If no task is available in the queue, the infected client will repeat the beaconing process and send another beacon after a static time-out. If the attacker added a new command to the queue, the command and control server replies to a beacon with a new task hidden inside the response traffic. Replying with a new task is done via the same transport protocol as which the beacon used for transmission. This beaconing process is displayed in figure 4. The tasks that

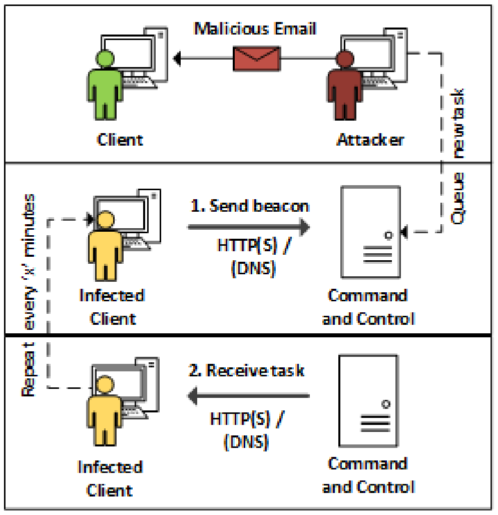


Figure 2. Receiving commands via beacons

can be queued by the operator varies. There are several default functions available by Cobalt Strike and Throwback, a few of these are (not limited to):

- Download a file from the internet;
- Send a list of running processes;
- Send a summary of the user's network activity;
- Send a dump of the hashed passwords;
- Open an interactive session with the Metasploit interpreter;
- Execute a system command.

As explained in the introduction, beaconing should offer the capability to secretly send and receive periodic commands to/from the command and control server. To fully interact with an infected client, the frameworks offer the capability to open an interactive session via the Metasploit interpreter. Once the client initiates a beacon and asks the command and control server for a queued task, the server will respond to open an active connection to the Metasploit server. However, this is very intrusive and can easily be detected with existing network monitoring tools. Hence the reason beacons are the primary communication method instead of a permant open connection with the command and control channel.

3. Research

3.1 Common network policies

For beaconing malware to succesfully establish a connection with a command and control server, different network policies and limitations have to be taken into account. Accessing the internet (ie. visiting your daily news-websites and sending your e-mails) involves many different protocols/transport methods. A company environment can decide to limit specific websites, ports and/or applications for their employees for whatever reason seems fit. There are a handful of generic network configurations you may run into during a social engineering assignment. These can include:

- No limitations: No proxys or captive portals are used;
- An unauthorized web-proxy: Websites are accessed via a configured proxy server. Authentication is not required
- An authorized web-proxy: Websites are accessed via a proxy server. Authentication is required. Widely used authentication mechanisms are Basic and NTLM. Digest, Negotiate and OAuth can also be used;
- A transparant proxy: An inline proxy that requires no client configuration. The client is usually unaware of a proxy located on the network.
- A captive portal: A service that redirects all web-requests to a landing (web) page where the user/employee has to log in.

Apart from the limitations mentioned above (which are primarily aimed for Hypertext transport protocol (HTTP) traffic), other protocol traffic can be blocked by different firewall policies. Think of a scenario as:

- Filtered internet: The policy includes a "reject all, except.." policy on their network. Employees are only able to communicate via specific ports/protocols that are deemed required by the organisation. Example: Drop everything except for port 80/443 for web-traffic.

Deploying Social Engineering malware often involves defining your outbound connection beforehand: What protocol/transport method should the malware use? If your assignment involves a large enterprise, different physical or organisational departments may have different internet policies. The IT department may be restricted in their activities if a proxy is explicitly required and/or specific ports are blocked. To maximize the success rate of social engineering malware, most (if not any) type of company environments should be taken into account when executing your assignment. The framework being chosen to craft your malware should support different transport mechanisms.

3.2 Common network defences

Apart from the specified policies, a company could also deploy defensive mechanisms to prevent protocol abuse, detect anomalies and perform deep-packet-inspection. Some products are able to decrypt SSL/encrypted sessions and perform analysis on the plain-text content of your connection. Companies may also deploy a Intrusion Detection System (IDS)/Intrusion Prevention System (IPS) to detect and/or prevent traffic behaviour and content on the network. Think of products as:

- Blue Coat: Stripping SSL Sessions for outbound web traffic;
- Snort/SourceFire: IDS/IPS to detect malicious traffic based on network signatures;
- Bro: Analysing protocols to detect possible mis-use;
- Lancope: Analysing NetFlow for traffic (non-content) anomalies.

The malware of your choice should also try to mask beacon traffic to prevent being blocked from the network and risk possible detection. The following sections will further discuss the popular frameworks and in what way they take company internet policies and detective/preventive measures into account. The proof of concept discussed in section 3.7 will demonstrate how social engineering malware can be further improved compared to the already existing frameworks.

3.3 Simulating network environments

Field-testing was performed together with setting up a lab-environment to simulate outbound malware connectivity in real-life use. This lab environment was configured to install, analyze and detect social engineering malware. A server located inside the OS3 lab hosted two different clients (Windows 7, fully updated) instances that act as control server for different types of malware. A third virtual machine would be configured as an intrusion detection system to detect command and control traffic from the infected VMs to any of the command and control servers inside the OS3 lab. The method which malware uses to communicate to their control server depends on the framework being used. Details about network traffic will be discussed in sections 3.4 and 3.5. The infected laptop was connected to a (large) company network, lab environment, a home network (ie. telecom ADSL + NAT), university network, public network (stations/coffeshops) and semi-public network (ie. paid behind

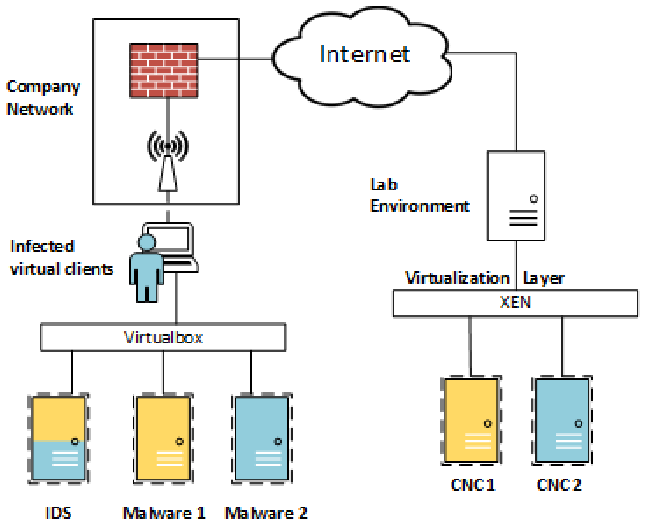


Figure 3. Basic overview lab environment

captive portals). The malware used during this research did not form a threat to other clients within the tested networks since the user has absolute control over the malware's functionality and activity. These field-tests do not necessarily indicate whether the malware will successfully operate on a company network since different policies and equipment is used within company networks compared to some of the public places I visited. To still give an indication what limitations are found within a variety of enterprises and smaller/medium sized companies, a few interviews were held with security professionals to ask about their experiences performing penetrations tests on-site. More about this will be discussed in section 3.4. Some of the different environments I visited:

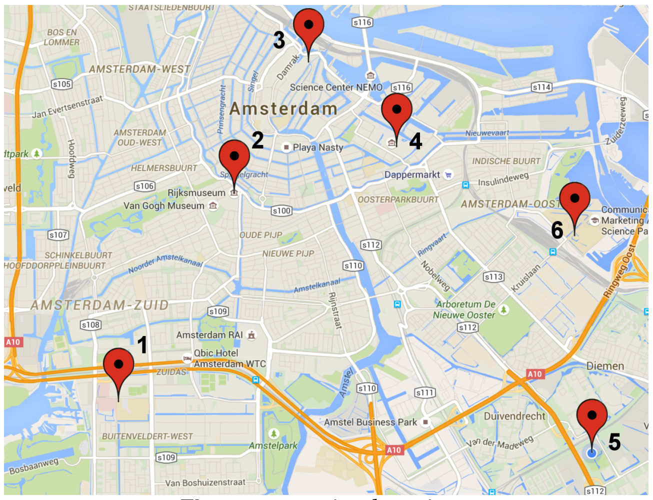


Figure 4. Testing locations

1. A company network providing 'guest' wireless (captive portal) and regular employee network;
2. A cafe free wireless network

3. Wireless inside the train. Requires a valid login inside a captive portal
4. My own home network (no filtering)
5. University network (SURFnet)
6. A campus network WiFi network

To simulate possible detection/defensive capabilities, an IDS was configured in the lab to monitor our beacon traffic. An IDS is a network (or software) component that monitors network and/or system activity for possible malicious behaviour. This research focussed on signature-based IDSs to detect Cobalt Strike and ThrowBack beacons. Different non-signature based detection methods can be used as well but are currently not tested in a lab environment. A theoretical approach to detect Cobalt Strike and ThrowBack beacons can be found in section 3.8.

Based on a market research done by the SANS Institute [4], the most common network monitoring IDSs are maintained by Cisco, SourceFire (which is now a Cisco company) and McAfee. The SourceFire product is a further developed and commercially maintained version of the open-source Snort IDS. Many non-Cisco IDSs (such as HP TipingPoint and McAfee IntruShield) either offer native Snort signature support or offer out-of-the box conversion tools. Since Snort (and it's signature format)is one of the most common network security tools, this report wil discuss possible detection methods based on the principle of Snort's core engine; intrusion detection via signatures. For a complete overview of the Snort Rule syntax, please refer to the 'Writing Snort Rules' [5] wiki

3.4 Malware features and network effectiveness

As discussed in the previous section, there are different network policies and detection methods that can be found in company environments. An important aspect of both Cobalt Strike and ThrowBack is the chosen beacon communication channel.

Cobalt Strike offers three different transport methods for beacon communication while ThrowBack has only implemented one as of date.

Channel	Cobalt Strike	ThrowBack
HTTP	Yes	No
HTTPS	Yes	Yes
DNS	Yes (TXT)	No

A security professional crafting the beacon configures a predefined communication channel. When the malware is deployed in a social engineering campaign, the beacon will always use the configured channel. This will involve pre-research on the target's network environment to determine what limitations are in place. Depending on the method of deployment, this may lead to issues regarding back-end reachability.

Example Scenario: A USB-Drop campaign is organized. Several USB sticks containing malware are given away or dropped at a client location. The security professional configured the malware to communicate over a direct HTTPS channel. One of the client's employees inserts the USB stick in his/her computer and executes the file. What the security professional did not know, is that all the employees make use of a NTLM-based proxy nor captive portal to access the internet. The malware will not be able to reach the command and control server and the security professional does not know whether clients were infected or not.

Implementing a fall-back method is not yet implemented in Cobalt Strike and/or Throwback. This will reduce the changes of successful malware deployment, but not significantly. Based on interviews held with several security professionals on their experience at client locations, about 75% of the **internal** networks required no manual proxy authentication or limited access HTTP and/or HTTPS access on any way. The networks involved belong to around 40 different insurance companies, industries, governments and small/medium sized businesses. Related to network configurations, this can mean two things:

1. The company has no proxy configured on the network;
2. The company has a transparent proxy configured on the network (ie. not visible to the user);
3. The company uses automatic proxy configuration without authentication.

The percentage of the different options found on company networks is unknown. For the effectiveness of malware research, it is only important to know if manual configuration and authentication is required for beacons to reach the external network. This number is purely indicative and doesn't provide any guarantees. It is highly dependant on the factors such as the company sector and network segment of deployment (ie. different departments and policies). The table below provides an overview of different network limitations and the operational status of the tested malware frameworks:

Configuration	Cobalt Strike	ThrowBack
Authenticated (ex. NTLM) proxy	Supported	Not supported
Non-Authenticated configured proxy (ex. IE proxy settings)	Supported	Partially supported
Transparent proxy	Supported	Supported
Captive Portal*	Supported	Not supported

Note: During the performed field tests it seemed that I did not have any issues using an external DNS server (port 53 outbound allowed) while a captive portal was enforced. This enabled Cobalt Strike's DNS beacons to still successfully reach the back-end server.

There are several reasons why the different malware frameworks show different results regarding outbound web connectivity.

Issue 1: A common (and important) issue is NTLM authentication. Common proxies that make use of NTLM for authentication require a valid challenge/response session to be initiated. There are libraries available to enable NTLM-enabled proxy traffic, but your application/malware will need the user's (hashed) password to generate the correct headers.

Issue 2: Cobalt Strike beacons work behind transparent proxies and proxies configured inside the Internet Explorer connection settings. ThrowBack does not work with the default Windows proxy settings because of the library being used. There are two main Windows API's available for applications: WinHTTP and WinINet. The WinHTTP API is being used by ThrowBack but does not automatically read the Windows registry for the configured (unauthorized) proxy settings. ThrowBack implemented it's own method to parse and apply proxy configuartions (based on individual user settings). However, this function did not work consistently and is dependant on the Windows version being used. The reasoning behind choosing WinHTTP over WinInet is discussed by Brady Bloxham during his DefCon talk [3]; the WinHTTP API is meant for windows services while the WinINet API is not (it can prompt the user for credentials). One of ThrowBack's core features is to run stealthy as a service, hence the decision for choosing the WinHTTP API and implementing custom proxy verification.

3.5 Analysis: Cobalt Strike

3.5.1 Beaconing traffic

Cobalt Strike uses three default methods of Beacon communication. The default beacon channels are HTTP, HTTPS and DNS. The primary beaconing mechanisms that will be discussed is the HTTPS beacon; reasoning that this is the only channel where no clear detection methods are available due to encryption standards. The HTTPS beacons makes use of TLS1.2 with EC Diffie-Hellman. The key exchange is performed the first time when the beacon is activated; the upcoming signals to/from the command and control server don't have to redo the entire handshake again. Each following beacon (when no command is set in the CnC queue) has the exact same behaviour and packet size:

- Each beacon session sends/receives 18 packets total and includes a single request and single response;
- If no command is put in the CnC queue, the response will always be of 197 bytes in size;
- Each beacon session closes with a Reset (packet) (RST)/ Acknowledgement (packet) (ACK). The server sends a FIN/ACK TCP packet while the clients replies with a RST/ACK.

The content itself can be decrypted if the private key is obtained from the Cobalt Strike beaconing server. This research focusses on detectability of social engineering malware; in a realistic scenario we would not have the private key and we should only able to able to create detection methods based on HTTPS meta-data and behaviour characteristics.

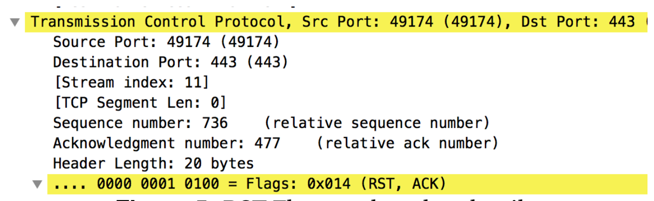
The other default beaconing mechanism over HTTP involves clear-text communication. The beacon displays the following characteristics:

- A HTTP request is sent to http://[ip||domain]/ptj;
- The beacon puts its request in the HTTP cookie field;
- When no command is present in the queue, the server replies with **Content-Type: Application/octet-stream** but does not send any data (specifying **Content-Length: 0**);

A third beaconing mechanisms consists out of a Domain Name System (DNS) tunnel. Cobalt Strike will query the beaconing server with a DNS request. The beaconing server will reply with command hidden in TXT fields. Tunnelling your traffic over DNS can be very slow and it's easy to detect with the most common monitoring tools. The developers of Cobalt Strike implemented this feature for a very valid reason; if a client is successfully infected and moves to a network where no outbound web-connectivity can be established, DNS (port 53) is often a channel that is still open.

3.5.2 Intrusion Detection

HTTPS Beacons: As explained earlier, the response from the Cobalt Strike server is always the same in size (without a queued command) and the client always closes the connection with a TCP Reset. Another thing noticed is that Cobalt Strike does not (by default) offer the availability to change the public key used during the beacon communication. This makes it easier to write specific detection mechanisms that trigger on the specified behaviour and public key. Since the



```
▼ Transmission Control Protocol, Src Port: 49174 (49174), Dst Port: 443
    Source Port: 49174 (49174)
    Destination Port: 443 (443)
    [Stream index: 11]
    [TCP Segment Len: 0]
    Sequence number: 736    (relative sequence number)
    Acknowledgment number: 477    (relative ack number)
    Header Length: 20 bytes
  ▼ .... 0000 0001 0100 = Flags: 0x014 (RST, ACK)
```

Figure 5. RST Flags and packet details

HTTPS Beacon's traffic is encrypted we are unable to look at the content of the session. However, based on Cobalt Strike's network behaviour, the following Snort Signature could be made for HTTPS Beacon detection:

Snort Signature: Cobalt HTTPS Beacons

```
alert tcp any 443 -> any any (msg:"C&C - CobaltStrike
     Response DataSize match (pre-rule)"; dsize:197;
     flags:PA; sid:1233340; flowbits:set, cobaltbeacon;
     priority:3; rev:3;)
alert tcp any any -> any 443 (msg:"C&C - CobaltStrike
     Beacon Verified Activity"; ttl:=128; flags:AR; sid
     :1333340; flowbits:isset, cobaltbeacon; priority:10;
      rev:3;)
```

The first rule will trigger if an IDS finds a response from source port 443 with a datasize of 197 bytes. The datasize matches the amount of bytes in the (application data) content including the amount of bytes in the TCP header (20 bytes). This by itself is not necessarily an alert, but this rule will set the flag (slight mis-use o the fowbits function, feedback is welcome) 'cobaltbeacon' that allows another rule to be triggered. The second rule will trigger if the first rule enabled the 'cobaltbeacon' flag and the clients sends back a response with a Time to live (TTL) value of 128 while the **Acknowledge** and **Reset** TCP flags are set.

HTTP Beacons: Opposed to the Cobalt Strike HTTPS beacon, the HTTP beacon's content can be read in clear-text. The **/pj** Uniform resource identifier (URI) is requested for every request. Monitoring for the specified URL can be a trigger to raise an alarm. However, the url/page name can be used by any website on the internet. The beaconing server shows default behaviour by responding with no content but still specifying an application stream in the HTTP headers. The following rule was able to detect the HTTP beacon:

Snort Signature: Cobalt HTTPS Beacons

```
alert tcp any any -> any any (msg:"C&C - CobaltStrike
     HTTP Beacon URL"; content:"|2F|ptj"; offset:4; depth
     :10; flowbits:set, cobalthttp; sid:143340; rev:1;)
alert tcp any any -> any any (msg:"C&C - CobaltStrike
     HTTP Beacon Verified Activity"; content:"Application
     /octet-stream"; dsize:>200;)
```

The first rule will trigger if an IDS finds the **/ptj** keyword between bytes 4 and 10 of the (application) data content. This is specifically refered to as the first header (ie. GET HTTP1.) of a HTTP packet. The first rule will hen set a 'cobalthttp' flag that allows the second rule to be triggered. The second rule checks that the HTTP headers specify **Application/octet-stream** as the content-type without actually sending any data back (ie. **Content-Length: 0**)

3.6 Analysis: ThrowBack

3.6.1 Beaconing traffic

ThrowBack only has a single HTTPS beacon channel for communication. The HTTPS beacons makes use of an encrypted TLS 1.2 connection. As compared to the Cobalt Strike HTTPS Beacon; the key exchange is performed every time the beacon attempts to contact the command and control server. Each beacon (when no command is set in the queue) has the exact same behaviour and packet size:

- Each beacon session sends/receives 18 packets total and includes a single request and single response;
- If no command is put in the CnC queue, the response will always be of 688 bytes in size. The request size differs per client;
- The client closes it's connection with a default FIN/ACK.

Compared to Cobalt Strike, the client closes it's connection according to the official TCP handshake; where Cobalt Strike always sends a RST packet.

The ThrowBack beaconing mechanism itself can be discussed in more detail since the entire project is made open-source (client and backend). The ThrowBack client only has a single beaconing mechanism that operates over HTTPS and communicates with a ThrowBack back-end webserver. This means that, as opposed to Cobalt Strike, no DNS tunnel is available when an infected client is located on a strict network with proxies and/or captive portals. The ThrowBack backend accepts a pre-defined POST format as input. The original ThrowBack beacon to the back-end server has the following format:

```
enc=123spec!alk3y456&hn=joeydreijer-VirtualBox& num
     =10.0.2.15&id=joeydreijer-VirtualBox8796759603609&pp
     =0&vn=2.50
```

The above beacon data consists out of several different elements.

1. enc: The 'command' key to verify a legitimate response/request from the command and control server;
2. hn: The client's (configured) hostname;
3. num (changed): The IP address of the network interface that is configured with to communicate via the default gateway;
4. id: A unique ID of the client. Consists out of the hostname + MAC Address of the interface;
5. vn: Malware version number (static);
6. pp: Flag (1 or 0) that checks if a client is behind a proxy or not.

Before this data is sent to the backend, the data is encrypted (RC4 legacy by default, AES improvement optional) and encoded with BASE64. All of the encrypted/encoded data is sent over a TLS1.2 session to a central ThrowBack server. The beacons show similar behaviour as that of CobaltStrike: al beacon is sent with static intervals and the content size of the beacon is always the same (when no command from the queue is executed). The beacon displays the following characteristics:

- A HTTP POST is sent to http://[ip||domain]/cp/index.php;
- The beacon data consists out of different encrypted and encoded variables as discussed earlier;
- The server replies with **<hidden stup1fy 0/////** > if no command is available.

3.6.2 Intrusion Detection

As explained in the network traffic subsection, the response from the Cobalt Strike server is always the same in size (without a queued command). Since the client does not perform any non-default behaviour (such immediately sending a RST packet), other detection flags should be taken into account. A rule that only triggers on a specific amount of bytes as response may generate false-positives. However, the response-size can be combined with a rule that checks for the relative TCP sequence number. The absolute TCP sequence will always differ, but Wireshark (and Snort) are able to calculate the relative sequence which is always the same for each command and control session. HTTPS Beacon detection:

Snort Signature: Cobalt HTTPS Beacons

```
alert tcp any 443 -> any any (msg:"C&C - ThrowBack
    Response DataSize match"; dsize:654; seq; 1470;
    flags:PA; sid:1233390; flowbits:set, throwbackbeacon
    ; priority:3; rev:3;)
```

The rule will trigger if an IDS finds a response from source port 443 with a datasize of 654 bytes. The datasize matches the amount of bytes in the (application data) content including the amount of bytes in the TCP header (20 bytes) with the relative sequence number of 1470.

3.7 Analysis: Improving malware

The previous sections discussed possible detection methods for already existing social engineering malware. In order to bypass detection and strict network policies a proof of concept was developed based on the original ThrowBack client. The ThrowBack client was rewritten in Python with additional (evasive) functions. The Python-based client is functional on OS X, Linux and Windows (with Py2Exe). The client functions as a proof-of-concept since the executable is quite large: Around 5MB with Ultimate Packer for Executables (UPX) compression. The client adds some additional features, such as:

1. Add additional and random length padding to encrypted requests to bypass static IDS signatures;
2. Add additional and random timing to beacon frequency (ie. 10 minutes + somewhere between 1-80 percent) to evasive possible beacon detection;
3. Implement different DNS channels: Response data is now included in Resource record digital signature (RRSIG) records (less obvious than TXT and compliant to the format) to bypass the already existing TXT tunnel detection;
4. Command and Control traffic via trusted and often-visited domains. Thus bypassing communication to newly registered domains and blacklisted domains;
5. Provide multiple different channels for outbound communication (in case of blocked ports/domains).

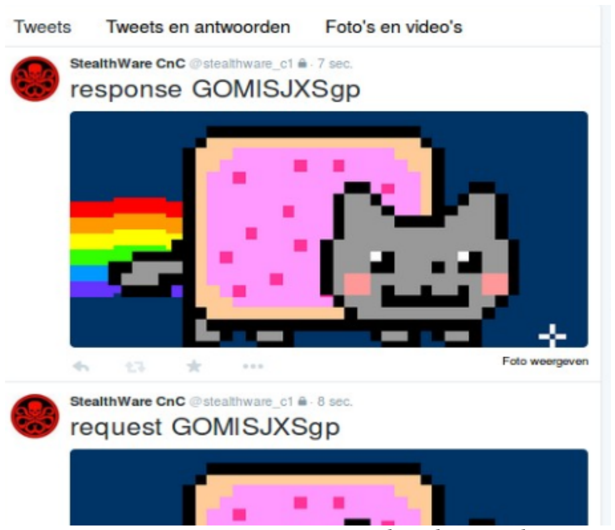
A new communication channel was added to ThrowBack to bypass different DNS detection methods (more about DNS malware detection in section 3.8). In short, malware detection via domains was based on three points: The age of a domain, the structure of a domain and possible lookup failures. To bypass these detection methods, the new ThrowBack client makes use of Social Media (Twitter) for communication. This works in the following steps:

- A client is infected with our malware;
- The malware uses creates its original requests (ie. the POST values);
- Instead of sending it over the default HTTPS channel, the data is encoded inside a Portable network graphics (PNG) image;
- The PNG image is uploaded to a twitter timeline with a specific identifier. The identifier consists out of 1) The type (ie. request) and 2) the unique identifier (a string of random characters);
- The back-end server listens to a twitter feed realtime. When an image is uploaded with the correct type and identifier, the image content is decoded and forwarded to the webserver (according to the original HTTPS channel specification);
- The back-end response is received and, in a similar fashion, encoded inside an image;
- The new PNG is uploaded to the twitter timeline with 1) a new type (ie. response) and 2) the same identifier as the request;
- The client extracts the content from the response and the process repeats itself.

In comparison, the proof-of-concept includes the same common channels plus the previously discussed twitter beacon channel:

Channel	Cobalt Strike	ThrowBack	ThrowBackPy
HTTP	Yes	No	No
HTTPS	Yes	Yes	Yes
DNS	Yes (TXT)	No	Yes (RRSIG)
Social Media	No	No	Yes

Apart from the evasive measures, a few minor functional adjustments have been made to the original ThrowBack POST parameters to give the malware operator more insight in the client's network limitations. The original ThrowBack client included 7 default POST parameters. The new format includes two new parameters and 3 adjusted parameter values. The adjusted beacon to the ThrowBack back-end looks as follows:


Tweets Tweets en antwoorden Foto's en video's

StealthWare CnC @stealthware_c1 · 7 sec.
response GOMISJXSgp

Foto weergeven

StealthWare CnC @stealthware_c1 · 8 sec.
request GOMISJXSgp

Figure 6. Twitter command and control

```
enc=123spec!alk3y456&hn=joeydreijer−VirtualBox&us=
        joeydreijer&num=10.0.2.15&id=joeydreijer−
        VirtualBox8796759603609&pp=0&vn=1.00&lolrandom=
        aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

1. enc (unchanged): The 'command' key to verify a legitimate response/request from the command and control server;
2. hn (unchanged): The client's (configured) hostname
3. **us (new)**: The client's username (original username that executed the malware);
4. **num (changed)**: The IP address of the network interface that is configured with to communicate via the default gateway;
5. id (unchanged): A unique ID of the client. Consists out of the hostname + MAC Address of the interface;
6. vn (unchanged): Malware version number (static);
7. **pp (changed)**: Originally states of the client is behind a proxy. This now states what type of channel is being used for command/control communication (ie. 1=Twitter, 2=HTTPS, 3=DNS, 4=HTTPS+Proxy);
8. **lolrandom (new)**: Additional padding for the request. This adds between 1-300 bytes of extra data that is not being parsed by the back-end server.

Next to the minor adjustments mentioned earlier, the new client is able to find it's own way out of the network. As explained in section 3.4, the default Cobalt Strike and ThrowBack beacon clients have a pre-defined channel that will be used if deployed (ie. communicate over HTTPS). If this channel is blocked due to network policies, the malware will not be able to reach it's command and control server. To fix this, the new client uses a modular set-up based on prioritized channels to communicate with the back-end server. An example of the new architecture can be seen in Figure 6. Another issue found specifically for ThrowBack beacons involve Windows proxy settings. ThrowBack could not (natively) read the Windows proxy settings due to the WinHTTP API being used. The urllib2 libraries being used in Python will read the Windows registry for proxy settings (Internet Explorer proxy settings) and apply the same configuration to the beacon client. Based on the improvements and rewritten code a new operational comparison can be made between Cobalt Strike, ThrowBack (original) and ThrowBack (Python-edition):

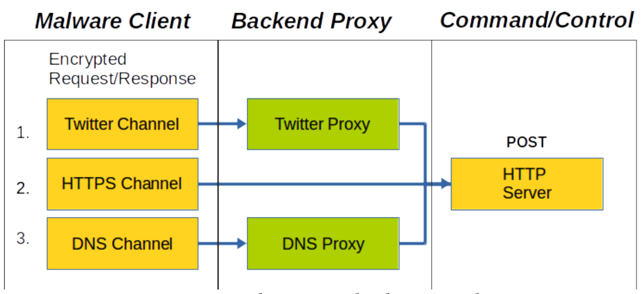


Figure 7. New ThrowBack client architecture

Configuration	Cobalt Strike	ThrowBack	ThrowBackPy)
Authenticated (ex. NTLM) proxy	No	No	No
Non-Authenticated configured proxy (ex. IE proxy settings)	Yes	No	Yes
Transparent proxy	Yes	Yes	Yes
Captive Portal	Yes	No	Yes

The (unfortunately) only remaining issue remaining involves NTLM authenticated proxies. During my research I was not able to find any methods that could include a valid NTLM token inside our request headers. This could be part of future research (or part of a future code-update during the summer).

The goal of the developed proof-of-concept malware is for security professionals to test their defensive capabilities. The malware can be run on systems to test different types of monitoring capabilities and verify their effectiveness. Furthermore, the new malware client can be used by security professionals as a method of entry security assessments.

3.8 Advanced detection methods

There are other alternative detection methods next to the signature-based approach discussed on the previous subsection. Detection malware via signatures is only effective for known-threats; unknown threats will not be detected. The static signatures to detect Cobalt Strike and ThrowBack would not work with dynamic malware behaviour as shown in the proof-of-concept developed for this research. The detection methods discussed below provide an indication how beacons can possibly be detected in a theoretical way without analyzing content with signatures. These could provide detectability for unknown threats. The methods described have not been practically tested but may provide a subject for future research.

3.8.1 Beacon detection

The current malware frameworks being used during this research use beacons as their primary form of communication. This means that a signal will be sent periodically to a (de/central) server to provide heartbeats and ask for commands. Research done by Leendert van Duijn [6] provides a proof-of-concept method to detect these beacons inside PCAP files. Van Duijn demonstrates that malware beacons can be detected by analysing network traffic statistics. Van Duijn mentions that it is indeed possible to identify beacons that are distinguishable from typical user traffic. Van Duijn mentions that ”Due to limited data, no conclusions can be drawn on generic HTTPS traffic” regarding differentiating legitimate user beacons and malware beacons over HTTPS. The primary channels for the researched malware makes uses of HTTPS, which means beacon detection may deem more difficult. Next to that, the current detection method is based on visual representations. This requires human interaction and analysis is done on forensic-data; thus not real-time (as of yet). The malware used during this research could potentially be detected by applying van Duijns beacon detection methods, but will only work if another detection method that operates real-time triggered an alert. The technique demonstrates by van Duijn can then be used to analyse the trigger/incident and verify a possible breach.

3.8.2 Domain anomalies

The malware used during this research makes use of a central (or multiple) command and control server(s). Before any communication can be established to these servers, a DNS query has to be performed to retrieve the server’s IP address. Robert Lemos [7] wrote an article that demonstrates three malware detection methods based on monitored DNS queries. These methods include:

1. Domain age: Domains that have been registered recently (ie. a week ago) may indicate a malware domain;

2. Esoteric domains: Analysing the uniqueness of a domain (in amounts of visits). An alert may show up if a specific domain name is not being visited very often and/or only by a handful of people within a large organisation;

3. Lookup failures: A client that attempts to look up non-existing domains may indicate a malware infection.

Apart from the three items listed by Lemos, a common malware detection method is by importing domain blacklists. These blacklists are updated frequently but will not contain new malware domains that are not known to the public yet. Another detection method is based on the structure of the domain name. Research done by Barry Weymes from Fox-IT [8] demonstrates two methods to detect malware based on DNS queries. One is related to the lookup failures discussed earlier, the other is based on the characters and structure of the domain name. Weymes mentions that there are three domain characteristics that he looks for:

1. Domain length: Length of the domain name (in 6 categories);

2. Character make-up: Whether the domains contains only chatacters, but also digits and/or consonants;

3. Top Level Domains (TLD): Variations and often-occuring TLDs for malware.

A few examples of malicious domains that are triggered by Weymes detection method:

```
agng78sagdfDKJdtwa887.com
kt2syggf436dtag312.com
tcdjnkntkkreatlbtbuguyxdqx.biz
cykjxnzmrhaygajncyfmoyljdpb.biz
gfeoacjlvufodylcsnbordyxs.cm
nhnjnmlgofabeynimrtdyt.cm
```

However, most of the detection methods making use of DNS anomalies assume well-known ”evil” botnets and not professional social engineering malware or custom malware crafted for targeted attacks. It’s safe to assume that with social engineering malware (or targeted malware), the professional will register a valid domain for a specific assignment/service that has a legitimate looking structure. These domains will most likely not be present in blacklists, not contain a Russian TLD and have a series of random characters in them. The researched malware may be detected if monitoring is in place to detect recently registered domains and unique domains (ie. esoteric). However, chapter 4 will demonstrate how these detection methods can be evaded by use of commonly visited social networks.

3.8.3 Traffic anomalies

Network anomaly detection is a mechanism often found to detect DDoS attacks, network downtime and malicious behaviour in general. A survey performed by Ripe NCC [9] tries to answer what an anomaly means and how/when security professional report these. The definition of an anomaly is described as an outlier (ie. peaks) on a specific dataset. The security expects that participated in the survey use different data sources to find these outliers. These datasets include, from mostly used to less used (top to bottom):

- Netflow/Sfloc;

- Logs;

- Custom scripts;
- Intrusion Detection Systems (IDS);
- MTRG/Torrus;
- Nagios;
- Honeypots.

The topic of IDSs was already discussed in chapter 4.2, but gathering network data from NetFlow can be of use when detecting beacons. Network traffic is generated by our beacons with a specific interval. The malware used during this research will always attempt to reach the command and control server, even when the user is not actively using his workstation. In theory, security specialists could define policies and monitor user behaviour outside office hours and/or weekends. A theoretical is to define a time-policy in order detect malware when the client is not performing any network traffic him/self during the weekends or after working hours. Of course, this means that the client should have his workstation/laptop running when not performing any work-related activities. This also means that other background processes or idiomatically refreshing websites that are left open will also generate traffic that may lead to false-positives.

Since beaconing traffic generates a low amount of traffic, it can be very difficult to detect with classical anomaly detection methods. However, a topic not yet discussed is the command and control server's queue. When the malware operator wants a client to execute a specific command, the results are uploaded/transferred via the same channel. The operator can also decide to start an interactive session with the infected client or attempt to download a file located on the infected system. The main issue regarding detectability is that the common channels all transfer their data over HTTPS. It is not uncommon to see large amounts of data being transferred over port 443, since many popular websites (think of Facebook, Twitter) make use of a secure connection to upload/download files. Purely looking at the behaviour of connections on the used ports may not be enough, but correlation with other monitoring tools may improve alerting. Combining anomalies over common ports with DNS anomalies discussed in the previous subsection may trigger an alert that is much more relevant. Example: If a client visits an uncommon and newly registered domain and starts to send excessive amount of data over port 443. This may not directly indicate a true-positive, but can lead to further analysis via other forensic methods. The malware operator (of CobaltStrike/Throwback) is not able to change the default behaviour of command and control traffic. Which means evasive behaviour like rate-limiting your file uploads / command shell will not be possible. This improves the changes of detection via any of the previously mentioned detection theories.

4. Conclusion

What existing frameworks already exists and what functionality do they offer?
There are only two public social engineering frameworks that offer command and control functionality. These are Cobalt Strike and Throwback. They provide security professionals with the extensive ability to deploy and craft malware to gain full control over a client's system. The frameworks are out there and are actively being used around the globe. Where Cobalt Strike offers a lot of functionality and beacon transport mechanisms, ThrowBack still lacks different outbound channels.

What common security policies are often found within company networks?
The malware being used for social engineering assignments have to take different types of network policies into account. Based on field testing and practical experience at client environments, there are several network configurations you may encounter during a social engineering assignment: Workstations behind a NTLM-authorized proxy, workstations behind an in-line proxy, workstations that make use of captive portals and different firewall policies that allow all outgoing traffic, only allow port 80/443 traffic or allow limited access but include services as external DNS, IMAP/SMTP. Next to the policies, different monitoring services may be in place to detect malicious activities. These can vary from IDSes that make use of signatures, NetFlow anomaly detection engines and/or packet inspection after stripping the SSL layer via appliances as Blue Coat.

How effective do the already existing frameworks work in realistic company environments?
The frameworks vary in operations. Cobalt Strike is a proven product with many different beaconing options, whereas the original ThrowBack client only supports communication to a central HTTPS server. There is one major flaw that influence operations: Both frameworks do not make use of fall-back methods. If your malware has been configured to communicate over HTTPS, it will only be able to operate over that specific channel. If a domain or IP address is blocked or if the client is behind a proxy or captive portal, no outbound connection can be made. The malware will not function since no fall-back method to, for instance, DNS is implemented. However, HTTPS/HTTP traffic is commonly accepted within company environments without any authorization. Regarding detection: the current beacons can easily be detected via multiple different methods. A client can monitor for newly registered and unique domains and perform anomaly detection on the data transfer. A more straight-forward method is to implement an IDS and detect the beacons with signatures. Both encrypted channels of Cobalt Strike and ThrowBack can be easily detected by using Snort signatures that inspect specific behaviour aspects and data response sizes. Even though these signatures were not already made available by the community, this report demonstrate PoC signatures that can be freely used to detect social engineering malware on your network. It is important to note that these signatures rely on the default behaviour of Cobalt Strike and ThrowBack. As is the case with any signature-based detection engine, when the content of the response/request changes, the signature is no longer valid. This research only focussed on the default beaconing behaviour and did not use the custom templating engine that ThrowBack provides.

Can these frameworks be further extended and/or improved to minimalize detection chances for advanced security audits?
A proof of concept malware client was developed during this research which makes use of different evasive techniques. Random request/responses were generated, popular domains (social media) were used to generate command/control traffic, random beacon timeouts were implemented to prevent beacon detection and fall-back mechanisms were implemented to bypass network policies. An additional DNS channel was created that makes use of RRSIG records to transmit data instead of the traditional TXT tunneling technique. However, the new proof-of-concept does not take SSL stripping mechanisms into account. It is still possible to detect the malware based on content analysis if plainly readable. The improved malware can be used by security professionals to test their security monitoring capabilities.

Acronyms

ACK	Acknowledgement (packet). 6
DNS	Domain Name System. 6
HTTP	Hypertext transport protocol. 3
HTTPS	Hypertext transport protocol secure. 5
IDS	Intrusion Detection System. 4, 7
IPS	Intrusion Prevention System. 4
NTLM	NT Lan Manager. 3
PNG	Portable network graphics. 10
RRSIG	Resource record digital signature. 10
RST	Reset (packet). 6
TTL	Time to live. 7
UPX	Ultimate Packer for Executables. 9
URI	Uniform resource identifier. 7

References

[1] Strategic Cyber. LCC. "http://www.advancedpentest.com", 2015.

[2] Brady Bloxham. Getting windows to play with itself. https://defcon.org/images/defcon-22/dc-22-presentations/Bloxham/DEFCON-22-Brady-Bloxham-Windows-API-Abuse-UPDATED.pdf, 2014.

[3] Brady Bloxham. Def con 22 - brady bloxham - getting windows to play with itself. https://www.youtube.com/watch?v=dq2Hv7J9fvk, 2015.

[4] SANS Institute. Intrusion detection faq: What are the top selling ids/ips and what differentiates them from each other? https://www.sans.org/security-resources/idfaq/top-selling-ids-ips.php, 2009.

[5] Cisco. Writing snort rules. http://manual.snort.org/node27.html, 2015.

[6] Leendert van Duijn. Beacon detection in pcap files. https://www.os3.nl/_media/2013-2014/courses/rp2/p73_report.pdf, 2014.

[7] Robert Lemos. Got malware? three signs revealed in dns traffic. http://www.darkreading.com/analytics/security-monitoring/got-malware-three-signs-revealed-in-dns-traffic/d/d-id/1139680?, 2013.

[8] Barry Weymes. Dns anomaly detection: Defend against sophisticated malware. http://www.net-security.org/article.php?id=1844&p=2, 2013.

[9] Jan Rejchrt. Network anomaly detection evaluation. https://labs.ripe.net/Members/jan_rejchrt/network-anomaly-detection-2013-survey-evaluation, 2013.

[10] S. et al Abraham. An overview of social engineering malware: Trends, tactics, and implications. *Proceedings of the 11th European Conference on Information warfare and security*, 2012.

[11] Trend Micro Corperation. Trends in targeted attacks. "http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp_trends-in-targeted-attacks.pdf", 2011.

[12] Guofei Gu et al. Botsniffer: Detecting botnet command and control traffic. "http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp_trends-in-targeted-attacks.pdf", 2008.

[13] Guofei Gu et al. Bothunter: Detecting malware infection through ids-driven dialog correlation. http://static.usenix.org/legacy/events/sec07/tech/fullpapers/gu/gu.html/, 2008.

[14] Michael Bailey et al. A survey of botnet technology and defenses. http://www.merit.edu/research/pdf/2009/catch09_botnets_final.pdf, 2009.

[15] Timothy Strayer et al. Detecting botnets with t ight command and control. http://isis.poly.edu/~kurt/fm/feb_15/StrayerWLL06.pdf, 2009.

[16] Original Wiki author; Casascius. Elliptic curve digital signature algorithm. https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm, 2015.

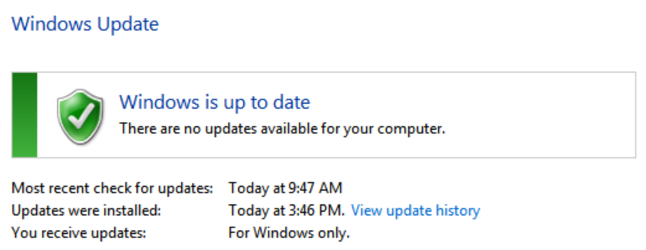
[17] Chris Jordan. Mcafee now highlighting snort signature integration. http://playingwithothers.com/2014/08/13/mcafee-now-highlighting-snort-signature-integration/, 2013.

[18] Cisco. Snort ids. https://www.snort.org/, 2015.

[19] SANS Institute. Detecting and preventing unauthorized outbound traffic. http://www.sans.org/reading-room/whitepapers/detection/detecting-preventing-unauthorized-outbound-traffic-1951, 2007.

5. Appendix

Image Material


Windows Update

Windows is up to date
There are no updates available for your computer.

Most recent check for updates: Today at 9:47 AM
Updates were installed: Today at 3:46 PM. View update history
You receive updates: For Windows only.

Figure 8. Fully updated Windows 7 instance that was running the malware. Displaying that updates will not prevent any execution limits

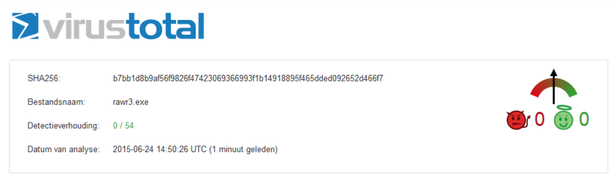
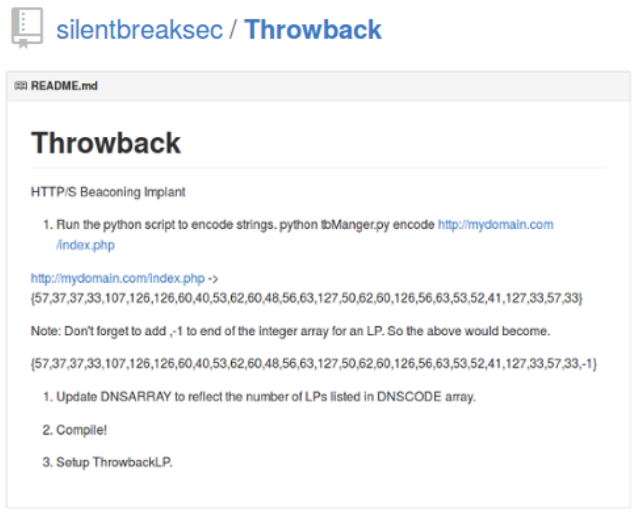

SHA256: b7bb1d8b9af56f9826f47423069366993f1b14918895f465dded092652d466f7
Bestandsnaam: rawr3.exe
Detectieverhouding: 0 / 54
Datum van analyse: 2015-06-24 14:50:26 UTC (1 minuut geleden)

Figure 9. VirusTotal report of a Python executable (Py2Exe) with UPX


silentbreaksec / Throwback

README.md

Throwback

HTTP/S Beaconing Implant

1. Run the python script to encode strings. python tbManger.py encode http://mydomain.com/index.php

http://mydomain.com/index.php ->
{57,37,37,33,107,126,126,60,40,53,62,60,48,56,63,127,50,62,60,126,56,63,53,52,41,127,33,57,33}

Note: Don't forget to add ,-1 to end of the integer array for an LP. So the above would become.

{57,37,37,33,107,126,126,60,40,53,62,60,48,56,63,127,50,62,60,126,56,63,53,52,41,127,33,57,33,-1}

1. Update DNSARRAY to reflect the number of LPs listed in DNSCODE array.
2. Compile!
3. Setup ThrowbackLP.

Figure 10. (Original) ThrowBack GitHub page @ https://github.com/silentbreaksec/Throwback

Snort signatures

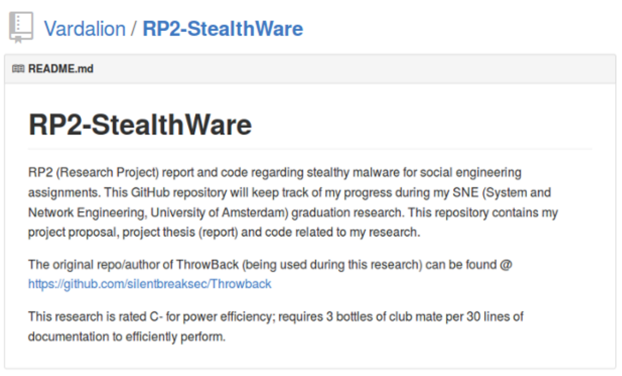
```

## Cobalt Strike HTTPS Beacons

alert tcp any 443 -> any any (msg:"C&C - CobaltStrike
     Response DataSize match (pre-rule)"; dsize:197;
     flags:PA; sid:1233340; flowbits:set, cobaltbeacon;
     priority:3; rev:3;)
alert tcp any any -> any 443 (msg:"C&C - CobaltStrike
     Beacon Verified Activity"; ttl:=128; flags:AR; sid
     :1333340; flowbits:isset, cobaltbeacon; priority:10;
      rev:3;)

## Cobalt Strike HTTP Beacons
alert tcp any any -> any any (msg:"C&C - CobaltStrike
     HTTP Beacon URL"; content:"|2F|ptj"; offset:4; depth
     :10; flowbits:set, cobalthttp; sid:143340; rev:1;)
alert tcp any any -> any any (msg:"C&C - CobaltStrike
     HTTP Beacon Verified Activity"; content:"Application
     /octet-stream"; dsize:>200;)


## ThrowBack HTTPS Beacons
alert tcp any 443 -> any any (msg:"C&C - ThrowBack
     Response DataSize match"; dsize:654; seq; 1470;
     flags:PA; sid:1233390; flowbits:set, throwbackbeacon
     ; priority:3; rev:3;)
```


Vardalion / RP2-StealthWare

README.md

RP2-StealthWare

RP2 (Research Project) report and code regarding stealthy malware for social engineering assignments. This GitHub repository will keep track of my progress during my SNE (System and Network Engineering, University of Amsterdam) graduation research. This repository contains my project proposal, project thesis (report) and code related to my research.

The original repo/author of ThrowBack (being used during this research) can be found @ https://github.com/silentbreaksec/Throwback

This research is rated C- for power efficiency; requires 3 bottles of club mate per 30 lines of documentation to efficiently perform.

Figure 11. RP2 Report and adjusted code GitHub page. Includes: PCAP Files of the Beacon traffic - Reports (Thesis+Proposal) - Proof of Concept SourceCode of adjusted ThrowBack @ https://github.com/Vardalion/

Sourcecode