GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Network Monitoring and Threat Detection In-Depth (Security 503)"
at http://www.giac.org/registration/gcia

HTTP header heuristics for malware detection

GIAC (GCIA) Gold Certification

Author: Tobias Lewis, tobaslouis@gmail.com
Advisor: Antonios Atlasis

Accepted: 24th December 2013

Abstract
Sophisticated malware, such as those used by Advanced Persistent Threat (APT) groups, will attempt to avoid detection wherever and whenever it can. However, even the stealthiest malware will have to communicate at some point, and when it does so, it provides an opportunity for detection. This paper looks at a number of techniques to identify the presence of malware which attempts to masquerade as legitimate web browsing activity, exploiting some of the occasionally inaccurate attempts to mimic the HTTP protocol. This should provide network defenders with greater opportunity to detect malicious activity, without the need for maintaining a corpus of virus specific signatures that are vulnerable to change.

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

1. Introduction

Signature based detection is one of the most fundamental techniques for identifying malicious activity on your network. However, these only really account for the so called "known, knowns" (Rumsfeld, 2002), and with numerous commercial offerings of threat indicators, it can be costly to maintain an up to date corpus of network signatures.

Behavioural, or *heuristic* based detection, provides a broader capability by attempting to identify malware from behaviour that is deemed to be, or at least associated with, nefarious activity – including that which has potentially never been observed before ("Heuristics", Virus Bulletin Glossary, n.d.). This paper discusses the use of heuristics in malware detection, focussing on the network traffic generated and specifically the attempts to masquerade as legitimate web browsing traffic using the Hypertext Transfer Protocol (HTTP).

HTTP is an application layer protocol that allows the transfer of data using the client-server model. Typically used for web browsing, clients issue a request to a server (such as a web server), which responds, either with the appropriate resource if available or some form of information or error message. The latest operational version (version 1.1) is defined in RFC2616 (Fielding et al., 1999).

Like most network protocols, HTTP makes use of headers to transfer metadata that provides the receiving entity with information on how best to treat the event. It may, for example, provide information on the browser being used to view a webpage, that tells the web server the best format to send back; or, which file types the client is expecting to receive as part of the request. Some of the more common header options, and those which I'll refer to in this paper, are as follows:

- User-Agent; used to describe the specifics of the software application making the HTTP request, for the purposes of ensuring compatibility and usability statistics

© 2013 The SANS Institute Author retains full rights.

© 2013 SANS Institute, Author retains full rights.

- Host; specifies the domain or IP address, where the requested resource is located, although for externally bound network traffic it is unlikely to see an IP address.
- Referer; a field used to indicate when a webpage visit is as a result of a hyperlink being followed, and will specifically contain the source of that link

A range of predefined headers (including those above) are listed in RFC2616 (Fielding et al., 1999), with most giving some indication of the expected format of the entry. For example, the User-Agent option requires the following format:

```
"User-Agent" ":" 1*( product | comment )
```

Which may look something like:

```
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

RFC2616 (Fielding et al., 1999) only explicitly requires that the Host field is present in a request; albeit with some edge cases, everything else is optional, including the ordering. It is also worth noting that the RFC does not define or describe the specific content of the header value, only the format or syntax it expects the individual options to be in. As a result there can be significant variation between implementations - although you would expect some consistency between products based upon the elements of the same source code, for example between different versions of Internet Explorer.

At a basic level, HTTP offers the good majority of the functionality required by malware, specifically referring to the ability to upload and retrieve data. Furthermore, due to its fundamental use in web browsing, HTTP is one of the more common protocols observed in networks both big and small. As a result, not only are source code libraries and modules for HTTP widely available, HTTP traffic is often enabled by default on security devices and network gateways.

Researching cyber crime botnets in 2009, it was identified that "...the majority of...bots use HTTP to communicate with their C&C [command and control] server..." (John, Moshchuk, Gribble & Krishnamurthy, 2009) and within the recent "APT1" report (Mandiant, 2013) which discusses the use of malware by a specific, sophisticated

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

"Advanced Persistent Threat" (APT) group, over 30 out of nearly 50 tools appear to communicate using HTTP-like protocols.

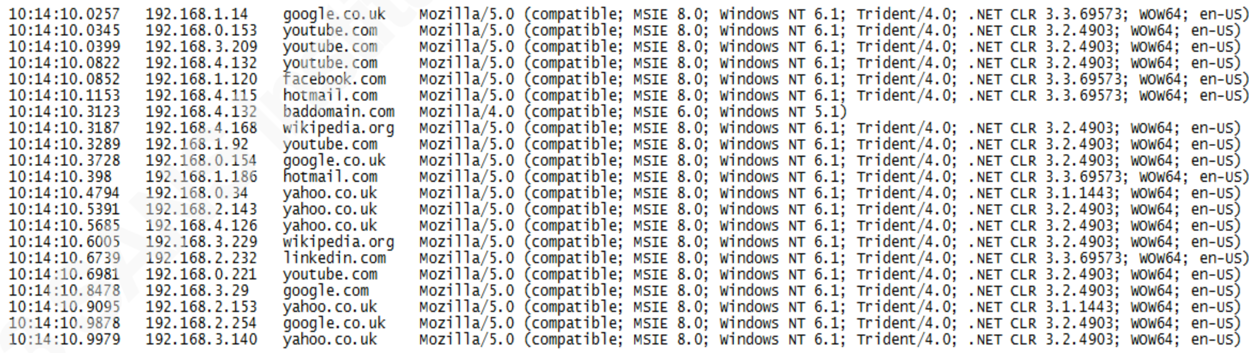
2. Analysis of HTTP heuristics

2.1. General heuristics

2.1.1. User-Agents

In enterprise environments, it is common for IT infrastructure to be centrally coordinated and managed. As a result, you could expect a fairly static IT build across the estate and thus minimal variation in the operating system and browser versions reported in the User-Agent. It could therefore be possible to rely on this relative predictability to help identify alien network traffic.

Figure 1 mimics what this could look like in a web proxy log. Whilst there is some slight variation amongst the other User-Agents, the anomalous entry is clear to see. Appreciating that in the specific example, the difference in line length makes it stand out; an observant network administrator should also be able to note the different operating system and browser version.



```
10:14:10.0257   192.168.1.14    google.co.uk    Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.3.69573; WOW64; en-US)
10:14:10.0345   192.168.0.153   youtube.com     Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.2.4903; WOW64; en-US)
10:14:10.0399   192.168.3.209   youtube.com     Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.2.4903; WOW64; en-US)
10:14:10.0822   192.168.4.132   youtube.com     Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.2.4903; WOW64; en-US)
10:14:10.0852   192.168.1.120   facebook.com    Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.3.69573; WOW64; en-US)
10:14:10.1153   192.168.4.115   hotmail.com     Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.3.69573; WOW64; en-US)
10:14:10.3123   192.168.4.132   baddomain.com   Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
10:14:10.3187   192.168.4.168   wikipedia.org   Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.2.4903; WOW64; en-US)
10:14:10.3289   192.168.1.92    youtube.com     Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.2.4903; WOW64; en-US)
10:14:10.3728   192.168.0.154   google.co.uk    Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.2.4903; WOW64; en-US)
10:14:10.398    192.168.1.186   hotmail.com     Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.3.69573; WOW64; en-US)
10:14:10.4794   192.168.0.34    yahoo.co.uk     Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.1.1443; WOW64; en-US)
10:14:10.5391   192.168.2.143   yahoo.co.uk     Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.2.4903; WOW64; en-US)
10:14:10.5685   192.168.4.126   yahoo.co.uk     Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.2.4903; WOW64; en-US)
10:14:10.6005   192.168.3.229   wikipedia.org   Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.2.4903; WOW64; en-US)
10:14:10.6739   192.168.2.232   linkedin.com    Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.3.69573; WOW64; en-US)
10:14:10.6981   192.168.0.221   youtube.com     Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.2.4903; WOW64; en-US)
10:14:10.8478   192.168.3.29    google.com      Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.2.4903; WOW64; en-US)
10:14:10.9095   192.168.2.153   yahoo.co.uk     Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.1.1443; WOW64; en-US)
10:14:10.9878   192.168.2.254   google.co.uk    Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.2.4903; WOW64; en-US)
10:14:10.9979   192.168.3.140   yahoo.co.uk     Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; .NET CLR 3.2.4903; WOW64; en-US)
```

Figure 1 - Sample web log showing an infected host

However, when we look at the network traffic generated by just a single host, it is possible for a large number of User-Agents to be present. From Windows Services to browsers built into applications such as iTunes, they all act to raise the noise floor. It could also be argued that this would not be an unexpected result given the propensity for legacy systems to remain on enterprise networks for support, contractual or backward

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

compatibility reasons. The increasing popularity of “Bring Your Own Device” (BYOD) and also guest users, could increase the likelihood of apparently anomalous User-Agents even further.

Despite this, it may still be possible to detect suspicious activity. Whilst we can look to defeat or white list User-Agents associated to Windows services such as Microsoft Update, these applications are subject to change and could lead to false positives. What may be more practical is looking for User-Agents that simply contain incorrect or false information and regardless of the browser or HTTP client being used; the Operating System for a *specific platform* should remain constant. This is a reminder that heuristics work best when fine-tuned to their environment.

Whilst the data in figure 1 is presented in a log format, network administrators could deploy SNORT based signatures to identify this behaviour. Taking into account some of the additional requirements to reduce the amount of False Positives, two sample SNORT rules are provided below. The first will hit on activity that doesn't present the User-Agent of a possible standard build (in this case, Internet Explorer version 9.0 on Windows 7), the second will hit on HTTP activity that doesn’t contain the correct Operating System (Windows 7):

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "HTTP activity
using non-standard User-Agent"; flow:to_server,established;
content:"User-Agent: Mozilla”; http_header; content:!"User-Agent:
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)";
http_header; classtype:bad-unknown; sid:1000000; rev:1;)
```

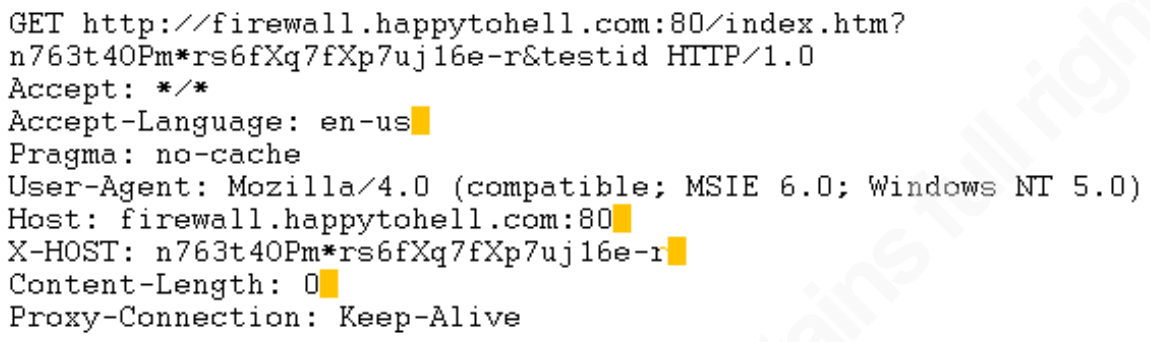
```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "HTTP activity
using User-Agent with non-standard Operating System";
flow:to_server,established; content:"User-Agent: Mozilla”;
http_header; content:"Windows"; http_header; content:!"Windows NT
6.1;"; http_header; classtype:bad-unknown; sid:1000001; rev:1;)
```

2.1.2. Typographic Errors

Much of the malware examined for this paper appears to use explicitly hardcoded header options and these could be prone to simple typographic or syntactical errors which can be used to identify malicious activity.

© 2013 The SANS Institute Author retains full rights.

In the MEDIANA sample below (Parkour, 2013), there is surplus white space (ASCII character 0x20) at the end of a number of options before the carriage return and line feed:



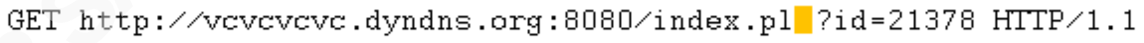
```
GET http://firewall.happytohell.com:80/index.htm?
n763t40Pm*rs6fXq7fXp7uj16e-r&testid HTTP/1.0
Accept: */*
Accept-Language: en-us 
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: firewall.happytohell.com:80 
X-HOST: n763t40Pm*rs6fXq7fXp7uj16e-r 
Content-Length: 0 
Proxy-Connection: Keep-Alive
```

This doesn't contravene RFC2616 (Fielding et al., 1999), which indicates that the presence of trailing "Linear White Space" in a header value can be removed without altering the meaning. However, for network efficiency, it would be unlikely for surplus "Linear White Space" to be included. This assumption is supported when looking at legitimate network traffic.

The following snort rule could be used to detect HTTP headers with similar superfluous white space:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "Extra white
space in HTTP Header"; flow:to_server,established; content:"|20
0d 0a|"; http_header; classtype:bad-unknown; sid:1000002; rev:1;)
```

The PROTUX (Parkour, 2013) sample also has surplus white space, in this case between the URL and the URL arguments shown and highlighted below:



```
GET http://vcvcvcvc.dyndns.org:8080/index.pl ?id=21378 HTTP/1.1
```

This is counter to the URL syntax prescribed by RFC2396 (Berners-Lee, Fielding, Irvine & Masinter, 1998), which treats white space in URLs as an "excluded character" and is thus disallowed.

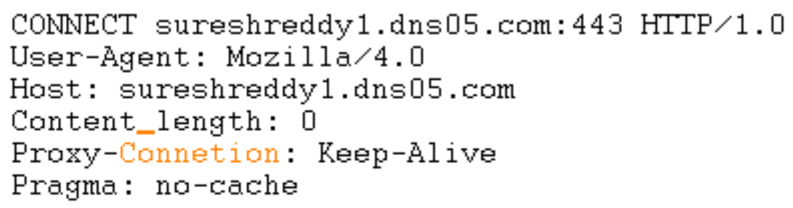
A sample as discussed by the Kaspersky Global Research & Analysis Team (branded as "GReAT") further demonstrates typographic errors in HTTP headers.

Author Name, email@address

© 2013 SANS Institute Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

QUARIAN (GReAT, 2012) is identifiable by incorrect[1] use of an underscore ("_") instead of a hyphen ("-") in the Content-Length header option, and further more by a misspelling of "Connection" in the Proxy-Connection option (although it's worth noting that this is not a header field predefined by RFC2616 (Fielding et al., 1999)):



```
CONNECT sureshreddy1.dns05.com:443 HTTP/1.0
User-Agent: Mozilla/4.0
Host: sureshreddy1.dns05.com
Content_length: 0
Proxy-Connetion: Keep-Alive
Pragma: no-cache
```

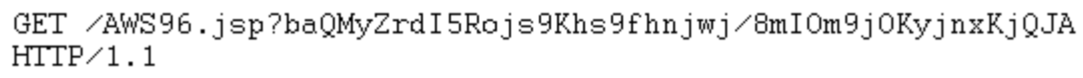
The PROTUX and QUARIAN samples both demonstrate errors with specific respect to the formal RFC. As a result, we can assume that this would not be present in HTTP traffic generated by a commercial browser, and that the risk of false positives would be low.

2.1.3. URL Complexity

When a user wishes to visit a specific website, they type the URL into the address bar of their browser and hit enter. It would be considered unlikely that a real user would be willing to type in a long or complex URL directly, although you might expect a more complex or long URL if it was being reached by the following of a link such as in the results of a search engine. In this case there should be a sensible referer field indicating this. RFC2616 (Fielding et al., 1999) doesn't formally limit URL length, although it does recommend that genuine web servers shouldn't rely on lengths greater than 255 characters, to allow backward compatibility with older clients.

There are a number samples, which by their inclusion of complex URL arguments and an absent referer field, would be deemed unlikely to have been manually typed in by a user:

IXESHE (Parkour, 2013):



```
GET /AWS96.jsp?baQMyZrdI5Rojs9Khs9fhnjwj/8mIOm9jOKyjnxKjQJA
HTTP/1.1
```

[1] Incorrect in terms of RFC2616 (Fielding et al., 1999), which specifically states that the field name should be of the format: "`Content-Length`"

© 2013 The SANS Institute Author retains full rights.

TAIDOOR (Parkour, 2013):

```
GET /gmzlk.php?id=031870111D309GE67E HTTP/1.1
```

However, there are some even more extreme examples, such as the MONGALL sample (Parkour, 2013):

```
GET
/3010850A0000F0FD0F0032313744374432453631363433383338004445
4C4C5854000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000
01000007014C61757261000000000000000000000000000000000000000
000000000000000000000000000000000 HTTP/1.1
```

Or worse, NETTRAVELER (Parkour, 2013):

```
GET /fly/2013/2011/nettraveler.asp?
hostid=E81B9088&hostname=DellXT&hostip=172.16.253.130&filename
=travlerbackinfo-2013-1-14-29.dll&filestart=0&filetext=begin::
tCvUBC2vGMy3Gu300GKz1EXQaCuRHQgIhFJhMLBUmNNhrtTsN9yhTLJTKhFJs4
STgtWw1lvSDEbjIXDEbC4ZDUZ3IzGEsWTZa73Z61ZqzmhNNJj0Rv0ZvMx0EBXN
JuMhv05zEBYKAgKFTMx10ZzNJvNxhNgskp10R2qJ1shhMZrEAzEAoIR1NJsMgg
GQ2GAwFIKkRaMVisxftVe0STmu305FnKodDMckchCx1KNpuDxnRkN6qFFbDYcx
4MxKKaPIZQKtMkcym6hDcZSSbjoZTYIDw0BsICMYTbQp5EFvk03mTNoajicrZ0
BwtQziAc5EDkZk0kFqZL0d7AdrEbS0c2pnwg4xGExSbKtfsmD3EXhvkfkDMAbj
AXDYbax6eFtevhZsxgtBYu3AWIBAIBgIC2Uzyc56G3TJHiNXNJGk8uKn1hGY4n
iczbV4XGU8GXLdIquZj0Zy8buM50HefBsN5njb1iRvmR0fJ07xA3G1gjYczD4Y
ccjCsx2NsfDoaDYc09/Dq0x2MZoiod0-k0Kn-dGIxGVyPZ7xv703P6Mvd7Rq0R
kIKLThMucpq0osG2wgcDeb5ujt1H891EZy0sLhsNgzg461LHanK0myYZ05kxpJ
zTMbfBmtg8gwpHk2TV9Dn1RFEXtEeH7P-ZTWcu6HGeTYajj23wzG1VRtMht6tA
ajabg7mSoQz9R9MfXL7zcNBrRqVCgQTrX4Q6hjcU6re6zyIobzPzjUHuPZC-Y4
Y42DMeesoG2WV44aZagus6pisxMdbfWBDfFyNhhj50GD5zsAzGusD3rwzGeUgd
D1bYc7a7Se4-NrMo4zhU5NPzy2p4AAbdj2LRJhjzSzMaT0dbjTpg2Z2mefix56
t6pIysBATo4oeRdfNvzd/N4fZgKQ7TZgGvF6cVk0xy5StACcfFn0pmninigV7v
x8oDk7B1zRDycPrfKVTcazd07153c0cd-UjfNI0fBFg3GI2GWcB8EVKIP1Gwrk
knFPSsHigx-LIIiZKrqD0pqgt HTTP/1.1
```

However, we can identify examples of long and complex URLs in legitimate traffic. For example, events generated by the Microsoft-CryptoAPI service, are often significantly longer than you would expect a legitimate user to be willing to enter – often over 100 characters in length with no Referer entry. Equally as prevalent are events associated to in page banner advertisements that your browser will send separate direct (non-referred) requests for.

Without any form of fine-tuning, in many cases this heuristic would only be sufficiently reliable when looking for URLs that exceed the recommended 255

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

characters. However, given knowledge of the environment, the ability to defeat events associated with certain User-Agents and/or domains, it may be possible to reduce the threshold.

Snort rules could be written for this characteristic, making use of the "urilen" keyword, with the following example using a threshold of 255 characters in the URL:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "URL length
exceeds 255 characters"; flow:to_server,established; urilen:>255;
classtype:bad-unknown; sid:1000003; rev:1;)
```

2.2. Using p0f for generic malware detection

2.2.1. Introduction to p0f for malware detection

Assuming that network traffic generated by malware effectively calls on the TCP library of the host operating system, we can use the structure of the OSI model to identify malicious software.

Using a passive fingerprinting tool such as p0f (Zalewski, 2012) we can look to characterise the host operating system by transport layer artefacts such as Time to Live (TTL) values, Window Size and Sequence numbers and referencing it to a known fingerprint library. We can then compare this result with the Operating System value as reported in the malware controlled application layer, such as the User-Agent field within a HTTP header.

The latest version of p0f goes further in attempting to identify the client used to generate the HTTP request, based upon the inclusion, exclusion and ordering of certain HTTP header options. This can be used to additionally identify fake User-Agent information, when, for example, an application purporting to be Microsoft Internet Explorer version 6.0 doesn't present the expected header options of a real IE6.0 browser.

2.2.2. Demonstration of p0f for malware detection

To demonstrate the effect of this, traffic was generated using two methods:

1. A Windows version of the popular Unix tool wget.
2. Mozilla Firefox v24.0 with a plug-in allowing custom User-Agents

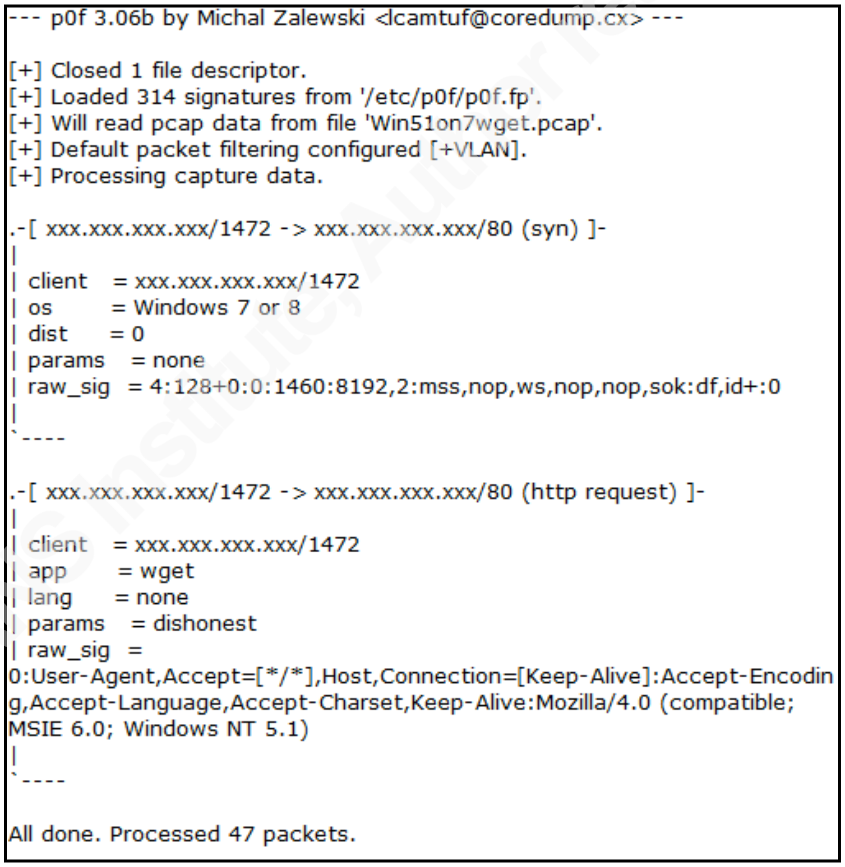
© 2013 The SANS Institute Author retains full rights.

© 2013 SANS Institute, Author retains full rights.

In both cases, this was conducted from a Windows 7 host operating system, using a fixed User-Agent that claimed to be Microsoft Internet Explorer 6.0 running on Windows XP:

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

Data was captured using Fakenet (Honig & Sikorski, 2012) to avoid unnecessary interaction with external web servers. As you can see below, p0f was able to correctly identify both the real host operating system and the application used to generate the application layer content.

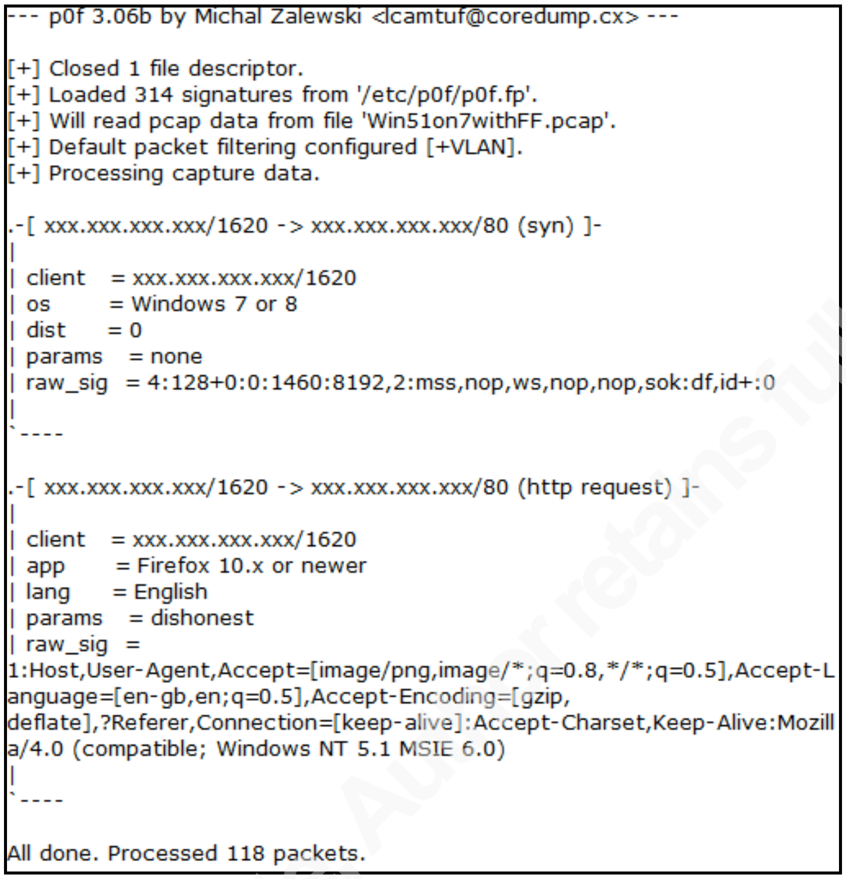


```
--- p0f 3.06b by Michal Zalewski <lcamtuf@coredump.cx> ---

[+] Closed 1 file descriptor.
[+] Loaded 314 signatures from '/etc/p0f/p0f.fp'.
[+] Will read pcap data from file 'Win51on7wget.pcap'.
[+] Default packet filtering configured [+VLAN].
[+] Processing capture data.

.-[ xxx.xxx.xxx.xxx/1472 -> xxx.xxx.xxx.xxx/80 (syn) ]-
|
| client   = xxx.xxx.xxx.xxx/1472
| os       = Windows 7 or 8
| dist     = 0
| params   = none
| raw_sig  = 4:128+0:0:1460:8192,2:mss,nop,ws,nop,nop,sok:df,id+:0
|
`----

.-[ xxx.xxx.xxx.xxx/1472 -> xxx.xxx.xxx.xxx/80 (http request) ]-
|
| client   = xxx.xxx.xxx.xxx/1472
| app      = wget
| lang     = none
| params   = dishonest
| raw_sig  =
0:User-Agent,Accept=[*/*],Host,Connection=[Keep-Alive]:Accept-Encodin
g,Accept-Language,Accept-Charset,Keep-Alive:Mozilla/4.0 (compatible;
MSIE 6.0; Windows NT 5.1)
|
`----

All done. Processed 47 packets.
```

Figure 2 - Results using WGET
(sanitized and snipped to show pertinent results)

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.



```
--- p0f 3.06b by Michal Zalewski <lcamtuf@coredump.cx> ---

[+] Closed 1 file descriptor.
[+] Loaded 314 signatures from '/etc/p0f/p0f.fp'.
[+] Will read pcap data from file 'Win51on7withFF.pcap'.
[+] Default packet filtering configured [+VLAN].
[+] Processing capture data.

.-[ xxx.xxx.xxx.xxx/1620 -> xxx.xxx.xxx.xxx/80 (syn) ]-
|
| client   = xxx.xxx.xxx.xxx/1620
| os       = Windows 7 or 8
| dist     = 0
| params   = none
| raw_sig  = 4:128+0:0:1460:8192,2:mss,nop,ws,nop,nop,sok:df,id+:0
|
`----

.-[ xxx.xxx.xxx.xxx/1620 -> xxx.xxx.xxx.xxx/80 (http request) ]-
|
| client   = xxx.xxx.xxx.xxx/1620
| app      = Firefox 10.x or newer
| lang     = English
| params   = dishonest
| raw_sig  =
1:Host,User-Agent,Accept=[image/png,image/*;q=0.8,*/*;q=0.5],Accept-L
anguage=[en-gb,en;q=0.5],Accept-Encoding=[gzip,
deflate],?Referer,Connection=[keep-alive]:Accept-Charset,Keep-Alive:Mozill
a/4.0 (compatible; Windows NT 5.1 MSIE 6.0)
|
`----

All done. Processed 118 packets.
```

Figure 3 - Results using Firefox with a User-Agent changing plug-in (sanitized and snipped to show pertinent results)

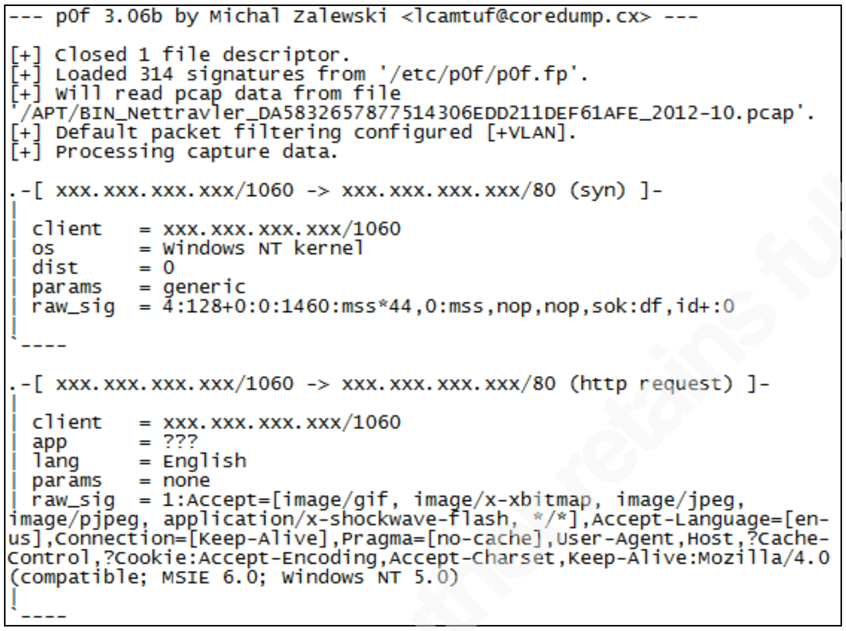
When we look at the full HTTP headers presented by samples from the Parkour (2013) dataset, p0f is unable to match the vast majority to the browser fingerprint they should have fired on. In fact we find that of the 11 samples that use a legitimate[2] looking User-Agent, 10 of them do not use or present the options as expected by p0f. In most cases, this is because the samples have an overly simplistic HTTP header, omitting a significant number of header options that would have otherwise have been included in a legitimate request.

Figure 4 shows the output of p0f when ran against the NETTRAVELER sample (Parkour, 2013). Whilst it is unable to identify the host operating system (instead only assessing that is using a Windows NT based Kernel), under the "app" field we can see

[2] Legitimate in that it appears to mimic a genuine operating system, starting with the Mozilla version token and then Operating System and Browser tokens followed in parentheses

© 2013 The SANS Institute Author retains full rights.

that the HTTP Headers do not fire on any of existing browser fingerprints, and certainly not Microsoft Internet Explorer version 6.0.

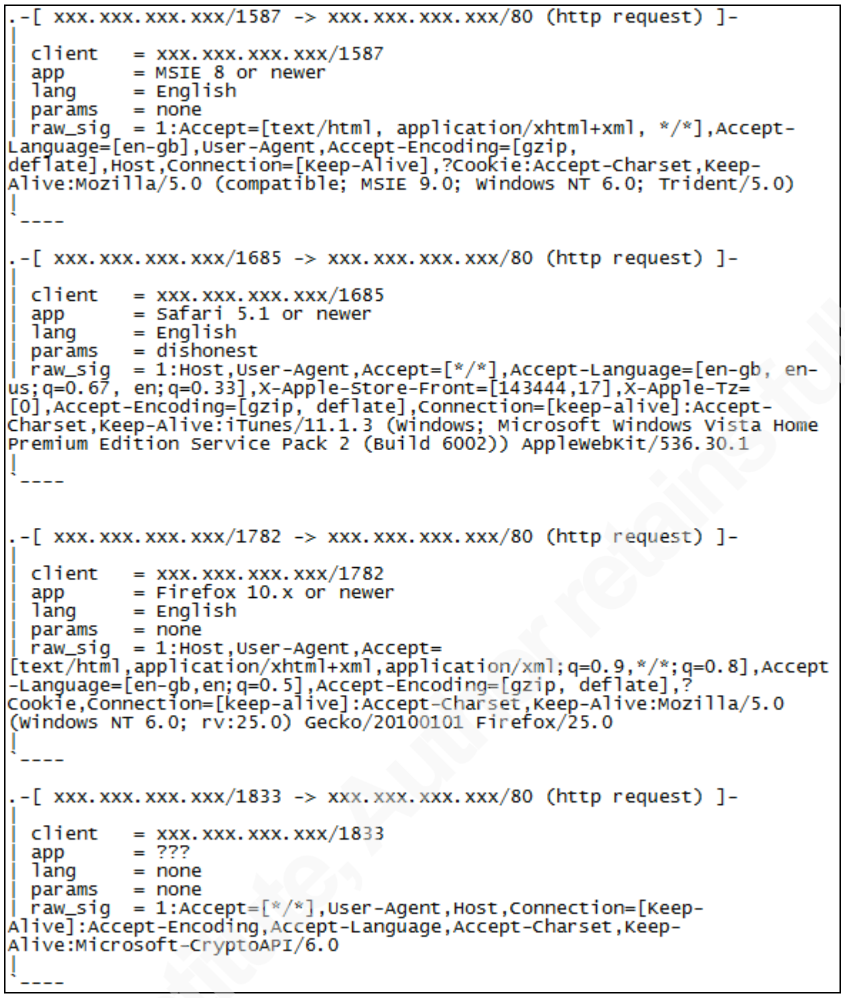


```
--- p0f 3.06b by Michal Zalewski <lcamtuf@coredump.cx> ---

[+] Closed 1 file descriptor.
[+] Loaded 314 signatures from '/etc/p0f/p0f.fp'.
[+] Will read pcap data from file
'/APT/BIN_Nettravler_DA5832657877514306EDD211DEF61AFE_2012-10.pcap'.
[+] Default packet filtering configured [+VLAN].
[+] Processing capture data.

.-[ xxx.xxx.xxx.xxx/1060 -> xxx.xxx.xxx.xxx/80 (syn) ]-
|
| client   = xxx.xxx.xxx.xxx/1060
| os       = Windows NT kernel
| dist     = 0
| params   = generic
| raw_sig  = 4:128+0:0:1460:mss*44,0:mss,nop,nop,sok:df,id+:0
|
`----

.-[ xxx.xxx.xxx.xxx/1060 -> xxx.xxx.xxx.xxx/80 (http request) ]-
|
| client   = xxx.xxx.xxx.xxx/1060
| app      = ???
| lang     = English
| params   = none
| raw_sig  = 1:Accept=[image/gif, image/x-xbitmap, image/jpeg,
image/pjpeg, application/x-shockwave-flash, */*],Accept-Language=[en-
us],Connection=[Keep-Alive],Pragma=[no-cache],User-Agent,Host,?Cache-
Control,?Cookie:Accept-Encoding,Accept-Charset,Keep-Alive:Mozilla/4.0
(compatible; MSIE 6.0; Windows NT 5.0)
|
`----
```

Figure 4 - Sample p0f response for NETTRAVELER (Parkour, 2013) (Sanitized and snipped to show pertinent results)

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.



```
.-[ xxx.xxx.xxx.xxx/1587 -> xxx.xxx.xxx.xxx/80 (http request) ]-
|
| client   = xxx.xxx.xxx.xxx/1587
| app      = MSIE 8 or newer
| lang     = English
| params   = none
| raw_sig  = 1:Accept=[text/html, application/xhtml+xml, */*],Accept-
Language=[en-gb],User-Agent,Accept-Encoding=[gzip,
deflate],Host,Connection=[Keep-Alive],?Cookie:Accept-Charset,Keep-
Alive:Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0; Trident/5.0)
|
`----

.-[ xxx.xxx.xxx.xxx/1685 -> xxx.xxx.xxx.xxx/80 (http request) ]-
|
| client   = xxx.xxx.xxx.xxx/1685
| app      = Safari 5.1 or newer
| lang     = English
| params   = dishonest
| raw_sig  = 1:Host,User-Agent,Accept=[*/*],Accept-Language=[en-gb, en-
us;q=0.67, en;q=0.33],X-Apple-Store-Front=[143444,17],X-Apple-Tz=
[0],Accept-Encoding=[gzip, deflate],Connection=[keep-alive]:Accept-
Charset,Keep-Alive:iTunes/11.1.3 (Windows; Microsoft Windows Vista Home
Premium Edition Service Pack 2 (Build 6002)) AppleWebKit/536.30.1
|
`----


.-[ xxx.xxx.xxx.xxx/1782 -> xxx.xxx.xxx.xxx/80 (http request) ]-
|
| client   = xxx.xxx.xxx.xxx/1782
| app      = Firefox 10.x or newer
| lang     = English
| params   = none
| raw_sig  = 1:Host,User-Agent,Accept=
[text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8],Accept
-Language=[en-gb,en;q=0.5],Accept-Encoding=[gzip, deflate],?
Cookie,Connection=[keep-alive]:Accept-Charset,Keep-Alive:Mozilla/5.0
(Windows NT 6.0; rv:25.0) Gecko/20100101 Firefox/25.0
|
`----

.-[ xxx.xxx.xxx.xxx/1833 -> xxx.xxx.xxx.xxx/80 (http request) ]-
|
| client   = xxx.xxx.xxx.xxx/1833
| app      = ???
| lang     = none
| params   = none
| raw_sig  = 1:Accept=[*/*],User-Agent,Host,Connection=[Keep-
Alive]:Accept-Encoding,Accept-Language,Accept-Charset,Keep-
Alive:Microsoft-CryptoAPI/6.0
|
`----
```

Figure 5 - Samples of legitimate HTTP activity
(Sanitized and snipped to show pertinent results)

In Figure 5, we see the results of p0f when compared to samples of legitimate traffic. Most notably is that p0f is not able to identify the application in all cases. This is ultimately because the p0f fingerprint library does not contain a fingerprint for every possible HTTP client (the Microsoft-CryptoAPI service in this example) – and nor should we expect it to. The way in which we can use p0f for malware detection is crucial to avoid unnecessary false positives.

Only where the User-Agent indicates a client or browser, for which p0f has an existing fingerprint for, we can rely on the results of the app field to identify cases where the User-Agent contains false or incorrect information and the event itself is likely to be malicious.

© 2013 The SANS Institute Author retains full rights.

Furthermore, there is nothing to stop network administrators from developing their own fingerprints for p0f, which adds an additional detection mechanism beyond SNORT based intrusion detection rules.

2.2.3. Deployment options

There are a number of deployment options for p0f, and although I have used it in a purely offline mode, network administrators could deploy p0f to gateway devices passively sniffing live network traffic or could even incorporate it into a larger suite of network monitoring tools using a built in API.

2.2.4. Considerations to note when using p0f

Based upon expected behaviour and the inaccurate attempts to spoof it, we can use p0f to identify malicious behaviour based upon discrepancies in two areas - the application layer and the transport layer. However, they do not present equal opportunities for detection.

In the first case, we are reliant on the host operating system differing from that referenced in the User-Agent field; if the two were the same, it would clearly not be possible to identify malicious behaviour using this technique. We can attempt to quantify this, albeit simplistically, by looking at the frequency of certain operating systems in spoofed User-Agents, with the market share of that operating system. 9 of the 11 Parkour (2013) samples are for operating systems that account for over 30% of the current market share (NetMarketShare, 2013) - the other two refer to "Win32" which is not possible to categorise at this time.

Furthermore, in the samples presented, p0f struggled to reliably detect the host operating system beyond the core kernel (Windows NT in this case). This could be due to lack of sufficient network traffic for p0f to make a thorough assessment, but equally it could be due to the dynamic nature of an operating systems kernel over its lifetime, as patches and hot-fixes from the manufacturer are installed.

In the second method (the use of HTTP header options), this is entirely dependent on the malware itself, and more specifically how well the malware author is able to spoof genuine browser activity. As a result, this method has a broader application and would

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

have a higher chance of detecting malicious activity (around a 90% success rate, albeit based on the limited number of samples discussed in this paper).

3. Recommendations

This paper has focussed on the *content* of the HTTP header, but techniques involving the timing of human generated versus automated events could provide further opportunity for detection. Likewise with flow profiles, such as upload/download ratios, top talkers, variance of domains and URLs could give some statistical methods for malware detection. These will require a better understanding of the "norms" of your network, and so is something for individual network administrators and security teams to explore the merits of in their own enterprise.

Behavioural detection takes time to fine tune and some of the techniques shown will not work on all networks. Policies allowing Bring Your Own Device or Local Administration rights, not to mention networks with a significant number of application developers could reduce the signal to noise ratio and increase the amount of False Positives. This paper only discusses techniques in a relatively vanilla environment, and anyone wishing to implement them, should do so with the understanding of these risks.

Based on the techniques discussed in this paper and the experiences in implementing similar methods, the following steps are recommended to help detect malicious activity on a network:

1. Baseline your network – understand what is unusual and what stands out.
2. A clear, delineated and compartmentalised network, helps to simplify network activity.
3. Look to lower the "noise" floor wherever possible – if something doesn't need to be installed or connected to the internet, then make sure it isn't.
4. Heuristics work best when fine-tuned to the environment.
5. Above all, don't be reliant on a single method for detection – each has its weaknesses.

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

4. Conclusion

This paper has demonstrated how seemingly minor spelling, typographic and syntactical errors provide network administrators opportunities for the detection of malware. It has also illustrated how the lack of variance in a network can aid detection by making alien traffic stand out more and how consideration for real world user behaviour could be exploited as well.

One of the most successful techniques was the use of passive fingerprinting tools and the comparison with real world browser software, which the malware is ultimately trying to mimic. Presenting error free headers is one thing. Presenting them in the right order, with the correct inclusion and omission of specific options is another thing altogether, and one that seems to be a fairly common trait across the samples tested.

No one technique was able to detect all samples, even in the limited collection. This should serve as a reminder that heuristics is only one tool in the armoury and should be used as a complimentary addition to other detection methods, such as signature based Intrusion Detection Systems or host based techniques. In combination, these will only increase the likelihood of being able to detect malicious activity so security teams can respond appropriately, but obviously, prevention is better than the cure!

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

5. References

Berners-Lee, T., Fielding, R., Irvine, U. C., & Masinter, L. (1998). Uniform Resource Identifiers (URI): General Syntax (otherwise known as RFC2396)

Fielding, R., Irvine, U.C., Gettys, J., Mogul, J., Frystyk, J., Masinter, L., Leach, P., & Berners-Lee, T. (1999). Hypertext Transfer Protocol (otherwise known as RFC2616).

GReAT (2012). A Targeted Attack Against The Syrian Ministry of Foreign Affairs. Available from https://www.securelist.com/en/blog/774/A_Targeted_Attack_Against_The_Syrian_Ministry_of_Foreign_Affairs

Heuristics (n.d.). Virus Bulletin Glossary. Available from http://www.virusbtn.com/resources/glossary/heuristics.xml

Honig, A. & Sikorski, M. (2012). FakeNet version 1.0c [computer software]. Available from http://sourceforge.net/projects/fakenet/files/latest/download

John, J. P., Moshchuk, A., Gribble, S. D., Krishnamurthy, A. (2009). Studying Spamming Botnets Using Botlab. Available from https://www.usenix.org/legacy/event/nsdi09/tech/full_papers/john/john.pdf

Mandiant (2013). APT1: Exposing One of China's Cyber Espionage Units. Available from http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf

NetMarketShare (2013). Desktop Operating System Market Share [live data as of 20th October 2013]. Obtained from http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0

Parkour, M. (2013). Collection of Pcap files from malware analysis. Available from http://contagiodump.blogspot.co.uk/2013/04/collection-of-pcap-files-from-malware.html

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.

Rumsfeld, D. (2002). US Department of Defense News Briefing, originally presented 12th Feburary 2002. Transcript available from http://www.defense.gov/transcripts/transcript.aspx?transcriptid=2636

Zalewski, M. (2012). p0f version 3.0beta [computer software]. Available from http://lcamtuf.coredump.cx/p0f3/

© 2013 SANS Institute, Author retains full rights.

© 2013 The SANS Institute Author retains full rights.