Analyzing Man-in-the-Browser (MITB) Attacks

GIAC (GCFA) Gold Certification

Author: Chris Cain, cicain08@gmail.com

Advisor: Dominicus Adiyanto

Accepted: December 22nd 2014

Abstract

The Matrix is real and living inside your browser. How do you ask? In the form of malware that is targeting your financial institutions. Though, the machines creating this malware don’t have to target the institution, rather your Internet browser. By changing what you see in the browser, the attackers now have the ability to steal any information that you enter and display whatever they choose. This has become known as the Man-in-the-Browser (MITB) attack. No one is safe from a MITB once it is installed, which easily bypasses the security mechanisms we all rely on. By infecting the browser and changing what is displayed we now have to wonder what world we are living in? Take the Red Pill and learn how this attack occurs to better allow you to hide from malware that target us every day.
1. Introduction

Malware today has become the method of choice to attack financial institutions. With the ease of use and ability for criminals to cover their tracks, this has been the way to rob banks without the need for a getaway car. Attackers are finding new and complex methods in which to carry out attacks. One of these vectors is a Man-in-the-Browser (MITB) attack.

Man-in-the-Browser (MITB) attacks have been around for some time and are utilized through trojan malware that infects an Internet browser. This attack is dangerous because of its ability to hide from anti-virus software and steal information a user types into the browser. MITB is able to see information within the browser. Since no encryption occurs within the browser, security controls used by financial institutions are ineffective. Two-factor authentication may also be ineffective if the malware has access to user account settings. Anti-fraud technologies that banks use to detect malicious activity are ineffective because the transactions occur from the user’s workstation. Many banks have added additional layers of security for wire transfers using notifications such as SMS texts. Though, if an attacker is able to steal users’ credentials then an attacker may have the ability to change notification settings in the user’s bank account.

Due to how MITB attacks work many network level devices such as web application firewalls, IDS and IPS systems have difficulty detecting this attack since it occurs locally on the client side. Decrypting SSL banking sessions may be a solution, but could create a backlash from users and management who require privacy.

What makes Man-in-the-Browser attacks popular is the ease to which it can be deployed to many systems at once via phishing links or through compromising legitimate sites. By clicking a link, trojan malware can be installed with add-ons into a browser that has not been properly secured. More attackers are moving away from the traditional Man-in-the-Middle (MITM) attack to the Man-in-the-Browser (MITB) attack for these reasons.

The difference between Man-in-the-Browser (MITB) and Man-in-the-Middle (MITM) attacks is in their operation. Man-in-the-Middle (MITM) attacks use a proxy between two systems that perform a transaction. Using a proxy an attacker can fool a user to enter their credentials into the attacker’s site, in turn giving away their sensitive information. Figure 1

Chris Cain, cicain08@gmail.com
illustrates a Man-in-the-Middle (MITM) vs. Man-in-the-Browser (MITB) attack. One important difference is that MITM operates at the network layer, while MITB operates at the application layer, i.e. the browser.

Figure 1

The reason Man-in-the-Middle (MITM) attacks have become less popular is due to the ability to mitigate the attack with the use of Session ID’s. If a bank is able to determine the number of session ID’s involved in a transaction, a bank can determine if there was a malicious user involved in the transactions between the systems. This would then give the bank a way to determine if a fraudulent attempt occurred and cancel the transaction. There are methods in which banks can also track user’s transactions by utilizing unique ID’s. By giving the customer’s device a unique ID, the bank can then use algorithms to analyze and link the multiple user sessions from where they typically perform their banking (Eisen, 2012). Man-in-the-Browser attacks go beyond intercepting or piggybacking traffic via a proxy page to fully taking over a user’s websites and controlling the browser in an effort to trick the user into thinking that everything is normal. By slightly altering web views and account balances, attackers can steal money without a user’s knowledge. Once the user logs in they can also redirect any sensitive traffic to an attacker’s system, while keeping the original SSL/TLS protections intact (Trusteer, 2013).
2. Man-in-the-Browser

Man-in-the-Browser (MITB) attacks utilize various functions and features within a browser. MITB attacks occur based on information gathered and what can be stolen similar to keylogging, form-grabbing, snapping screenshots, spamming, HTML injection and other various exploit functions. This gives the attackers information on when to use MITB as part of a malware attack. Browser extensions are a browser feature that can be used to exploit the operating system given the privilege given to extensions. Browser extensions are typically used to enhance users' experience within the browser and while surfing the Internet. Browser extensions can include plugins, Browser Helper Objects (BHO), JavaScript and add-on features. Many types of malware have been known to use these features as part of a MITB attack; these include Zeus, URLZone, Shylock, Spyeye, Carberp and Sunspot to name a few. Other functions that MITB utilize include AJAX, Browser API Hooking, and DOM Object models.

The functions of MITB can be controlled via a configuration file or a web injection file, which are updated at certain time intervals as part of a botnet. These configuration files may be obfuscated with different types of encoding. The configuration file and web injection file allow an attacker to control sessions and inject custom code into HTTP traffic. They also allow the trojan to run when certain websites are visited such as banking institutions. These connections typically occur over SSL connections. Since browsers have high level privileges on a system, if an attacker is able to execute processes through the browser then those processes can be executed with high level privileges (Alcorn, Frichot, Orru, 2014).

2.1. Browser Helper Objects (BHOs)

Browser Helper Objects (BHO) are DLL (dynamic linked libraries) modules which can access DOM (Document Object Model) within a browser. Browser Helper Objects were created by Microsoft and run in the address space of the browser and embed the main window of the browser (Blunden, 2009). They are installed as add-ons to the browser for added functionality. The issue with Browser Helper Objects is their ability to run with SYSTEM level privileges on the operating system. Browser Helper Objects have long been a popular method for hackers to abuse due to their ability to hide from anti-virus software. MITB attacks can use browser helper objects to change a site, adding fields or removing fields as an example.
Chris Cain, cicain08@gmail.com

can even add registry entries to the system, which will load at startup when a browser is opened (Utakrit, 2009).

Add-ons have been known to use MITB attacks, such as JavaScript and ActiveX controls to control the browser. One add-on that is popular with Firefox is Grease Monkey. Grease Monkey (Monkey-in-the-Browser) for Firefox and Tamper Monkey for Chrome apply the same methodology to a Man-in-the-Browser attack in that their function is to change what is viewed when visiting websites, such as eliminating ads from the screen or changing the appearance of a website. These features are to improve the user’s experience rather than steal information, but the methodology is the same. This is done with user scripts, which are JavaScript applets that can be shared within the community. User scripts used within add-ons are much more powerful than traditional JavaScript programs, because they can manipulate and retrieve private data in a user’s browser without Same-Origin Policy (SOP) restrictions (Acker, Nikiforaki, Desmet, Piessens, Joosen, 2011). Malware such as Zeus that utilize MITB features use configuration files to update scripts for the browser to use.

2.2. DOM Module Interface

The main method for MITB to work is through the DOM Module Interface. The steps that occur during this process are as follows. Once the trojan is installed it will install an extension into the browser configuration. This will cause the extension to reload when the browser starts back up. When the extension is loaded it registers a handler for every page load. So whenever a page is loaded, the URL of the page is searched by the extension against a list of known sites. Once the handler object detects a page it is loaded from the list and it registers an event button handler. Then once a page is submitted, the extension extracts all data from the form fields through the DOM interface in the browser, and remembers the values. The extension then tells the browser to continue to submit the form to the server. The server receives the modified values in the form as a normal request, which the server cannot differentiate between the original value and the modified values. The server performs the transaction and generates a receipt. The browser also receives a receipt of the transaction. The extension then detects the receipt URL, scans the HTML for the receipt fields and replaces the modified data in the receipt with the original data that was remembered in the HTML. The user then thinks that the original transaction was received by the server intact and authorized correctly (OWASP, 2009).
2.3. JavaScript & AJAX

One of the goals of an attacker is to maintain persistence. Using the previously described methods, this can be very difficult due to how features within a browser are performed. AJAX or Asynchronous JavaScript and XML solve these hurdles as it works in the presence of X-Frame-Option headers or other Frame-busting logic. JavaScript has the ability to “hook” the browser and perform actions entirely invisible to an end user. Below is an example web injection script used by the famous Zeus malware.

Example script:

```
set_url https://www.yourbank.com/*
data_before
<div class='footer'>
data_end
data_inject
<script src='https://somescript.com/hook.js'></script>
data_end
data_after
</body>
data_end
```

These scripts are implemented within the configuration files that are used in botnets. Zeus was famous for implementing configuration files that would call the Command and Control servers to inject new fields into banking sites to steal additional information beyond just capturing the user’s password.

One feature of JavaScript is the ability to override prototypes of built-in DOM methods. Overriding built in DOM methods in the browser is the same as extending DOM objects with your own method. Such as creating various form methods or additional fields for a user to fill in. This allows an attacker to see any sensitive information entered, such as PIN numbers, Mothers Maiden Name, DOB, etc.

Chris Cain, cicain08@gmail.com
2.4. API Hooking

Man-in-the-Browser attacks use API Hooking to infect the browser. Once MITB is activated from the malware, it will attempt to hook the Internet Connect function in Wininet.dll. This allows the attacker to modify what a user sees in the browser. This is similar to how HTML rewriting works. Using methods of HTML rewriting the malware can change the sites a user browses and make it appear in a certain fashion even presenting information that is not truthful. Figure 2 demonstrates the method of Browser API Hooking used in MITB attacks.

![Browser API Hooking Diagram]

Wininet, which is a superset to WinHTTP, is an API within Internet Explorer that enables applications to interact with FTP and HTTP protocols to access internet resources. Many wininet functions are targeted by MITB including the httpsendrequest() and navigateto() functions. Some other popular functions that are injected include httpopenrequest(), httpsendrequest(), and the internetreadfile function.

Changes to settings within the browser which allow this attack to be successful will leave artifacts behind in the Registry. To avoid Browser security settings that may prevent a script from properly displaying via an I-Frame or on a trusted site, malware may attempt to change security settings via the registry. Zone elevation within the browser is one of these methods. By lowering browser security settings more add-on controls and scripts will be able to run. A few dll’s that are a popular target of this type of malware include crypt32.dll and wininet.dll. Wininet.dll provides many functions for communication and is a target for malware since it allows the malware to access to privacy and security settings such as Zone preference settings.
and Cookie settings. Crypt32.dll implements many messaging functions in the CryptoAPI, such as the CryptSignMessage which also has the ability to digitally sign messages.

2.5. Registry Entries

For MITB maintain high level privileges, browser security settings are changed within the registry during exploitation. These registry changes can be monitored with host based intrusion detection systems, or analyzed after infection. Registry entries used in MITB attacks including the path for browser helper objects include:

- **HKLM\SOFTWARE \Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects.**
- **HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main**
  "NoProtectedModeBanner" = 1 - This turns on this function, which would disable Protected Mode in the Browser
- **HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed** – This is used to create randomly seeds for numbers in cryptography, quite possibly to hide malicious files
- **HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\3\"1406" (Miscellaneous: Access data sources across domains) = 3** - Sets the Zone Level to Low
- **HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\3\"1609" (Miscellaneous: Display mixed content) = 3** - Sets the Zone Level to Low
- **HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\3\"2500"** (Protected Mode) = 3 - Sets the Zone level to low
- **HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\"DisableCachingOfSSLPages" = "0"** - Turns this function off
- **HKEY_USERS\S-I-D\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\Random Number\**

3. Malware examples of MITB usage

Research into malware that utilize Man-in-the-Browser (MITB) as part of its exploitation was conducted to find the behavior of malware beyond the browser functions. Zeus was analyzed as well as a recent variant of the Shylock Trojan, both known to use MITB. Both

Chris Cain, cicain08@gmail.com
exhibit similar behavior since Shylock uses some of the Zeus source code features. Both use web injection files to inject into web fields and pages to steal banking credentials and perform wire transfers.

Since many malware have anti-sandboxing techniques a physical test machine was used. Various tools were used for the analysis, including win32dd and Dump-it as tools to extract a memory image of the system after infection. Volatility was used to examine the memory after it was dumped. Wireshark was used for packet captures and Regshot and Process Monitor were used to take a shot of the system before and after the infection. At one point a method was used to extract samples from remote systems that were live but unreachable via physical methods. To capture the memory remotely Kevin Neely found a method using psexec securely and win32dd/win64dd. The following is a sample of the method used. The account used to connect had appropriate permissions to execute win32dd/win64dd remotely (Neely, 2011).

- run cmd.exe as administrator
- net use \hostname\ipc$ - make sure command completes successfully
- copy c:\pathtowin32dd.* \hostname\c$ - copies win32dd.exe and the win32dd.sys driver
- c:\pathtopsexec.exe \hostname -e -w c:\c:\win32dd.exe /m 1 /r /a /f hostname-mem.raw
  - runs win32dd remotely, command will continue to run and will not give a status of completion. To verify it is complete run the following command and wait for the file size to stop growing. Please be aware of implications using psexec and credential passing that occur in cleartext.
- c:\dir \hostname\c$

### 3.1. Zeus

Zeus is a famous example of malware that utilize Man-in-the-Browser attacks. By use of a web injection file the malware is able to inject fields into designated websites that are entered into a file. So if a user visits www.bankofamerica.com the malware would use the web injection file to update the site and load the additional requested fields that are not legitimate. The following is an example web injection file used by Zeus.

```plaintext
;Build time: 14:15:23 10.04.2009 GMT;Version: 1.2.4.2
entry “StaticConfig”;
botnet “btn1” – Name of the botnet
timer_config 60 1 – Interval time for configuration file to be updated by bot in minutes
timer_logs 1 1 – Amount of time when bot will send data to the server
```

Chris Cain, cicain08@gmail.com
The malware also has the ability to clean itself from analysis including cookies and browser history to further hide itself from detection. This is to prevent support individuals being able to replicate the issue and stop it. This is one of the advanced features that show the capability and threat these malware can cause.

3.2. Shylock

Zeus has been a well analyzed over its lifetime and documented thoroughly once the source code was released many years ago. The Shylock Trojan that surfaced recently has caused harm to many organizations and individuals and has similar characteristics to Zeus yet with some
differences. Shylock was named after the famous Shakespeare play Merchant of Venice, because a few lines of the Shakespeare play were found in its code. Shylock based some of its source code from the Zeus malware, but added its own modules. Spyeye is another similar piece of malware that was based on the Zeus source code, but added its own modules, including one that would even delete the Zeus malware from a system.

Shylock has been known to run and create online chats when connecting to bank sites via advanced JavaScript. Many of the dropper files are named after chat programs such as Skype, Google talk, and Advantage. These files get dropped in the user’s folder under Application Data folder for Windows XP or the Roaming folder in AppData for Windows 7. Other modules that are included with the Trojan include VNC connectivity, spreading via network shares, separate drives or Skype sessions, as well as the ability as act as a proxy (Lennon, 2013).

The Shylock Trojan similarly to Zeus uses encoded web injection files in order to change websites. Several API’s are hooked including crypt32.dll and wininet.dll in the browser. It also uses fake digital certificates and SSL connections when communicating to the Command and Control servers.

During the analysis, once the system was setup, the malware was downloaded from sites that had testing copies of the Shylock dropper files used by Shylock and Zeus. The files were run on a Windows XP machine with analysis tools capturing the events and artifacts created. Memory was dumped using the Dump-it utility. Once the memory dump was retrieved Volatility, Wireshark and Process Monitor were used for analysis.

Process Monitor is a tool that can be overwhelming to use with the amount of data received. In analyzing Shylock several filters were used. These included file attributes, files written, files deleted, noise reduction, registry values set, registry values deleted, registry keys deleted, and registry keys created. The Process Monitor filters that were used were created by Raymond Hodge and were downloaded from Lenny Zeltser’s blog (Hodge, Zeltser, 2011). These filters created a starting point in which to begin using other tools such as Volatility and Wireshark.

The Process monitor filters found several possible artifacts including the use of normaliz.dll, which is associated with the Internet Explorer browser. Many registry settings were changed and added as well. Wininet.dll was also used during initial infection.
Shylock has many modules beyond MITB that are included, such as propagating via file shares, hiding folders using shortcut links that point to more additional malicious files. In the analysis one of the files that was created during the process was “nKMuLt.exe”. This file had an association with the normaliz.dll, which Process Monitor was able to capture in Figure 3.

![Figure 3](image)

Process Monitor found registry keys created during the time the malware was run. A couple keys in particular were related to Internet Settings. This is represented in Figures 3 and 4 below. In Figure 5, wininet.dll appears to be targeted by the process “apwQivQu.exe” which was created during the infection process.
### Event Properties

**Event**
- **Date:** 12/7/2014 3:40:56 PM
- **Thread:** 3764
- **Class:** Registry
- **Operation:** RegCreateKey
- **Result:** SUCCESS
- **Path:** HKCUSOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings
- **Duration:** 0.0003358

**Desired Access:** Read/Write

### Figure 4

![Image of Event Properties Table]

### Figure 5

![Image of Event Properties Table]

### Figure 6

![Image of Event Properties Table]

---

Chris Cain, cicain08@gmail.com

---

© 2014 The SANS Institute

Author retains full rights.
During analysis many registry keys were noted while using Process Monitor. These registry keys were then used for further analysis with volatility. Using volatility it was possible to determine the value of these registry keys.

The hivelist command in volatility was able to pull the registry hives of the users in the memory dump. Figure 7 shows the results of running this command. User “cjones” was the user profile of interest during testing.

$vol.py –f profile=WinXPSP3x86 shylock.raw hivelist

Figure 7

$vol.py –f shylock.raw profile=WINXPSP3x86 printkey –o 0xe1088a00 –K ‘Software\Microsoft\Windows\CurrentVersion\Run’

This command revealed that an executable RmActivate_isv.exe was set to run at startup, which would be one artifact left behind from the malware. This is shown in Figure 8 below.

Figure 8

The Wireshark captures found connections to soks.cc, pqe.su and doks.cc domains (Figure 9 & 10). These sites certainly did not sound legitimate so recording their IP addresses

Chris Cain, cicain08@gmail.com
was done for further analysis. In Figure 11, IP 208.73.211.70 appeared abnormal in the connections it attempted to make. This IP was not resolvable via a “whois” lookup and was categorized as a parked domain, potentially a former malicious IP.

In Figure 12 volatility was used to show the process that was using this connection.

```bash
$vol.py –f shylock.raw profile=WINXPSP3x86 connscan
```

Volatility revealed a process ID of 1468, which was the explorer.exe process, which would be a suspect process in this case. Figure 13 shows the results.

```bash
$vol.py –f shylock.raw profile=WINXPSP3x86 psscan
```

Chris Cain, cicain08@gmail.com
Analyzing Man-in-the-Browser Attacks

Figure 13

Once explorer.exe was identified as the process in question the mutantscan plugin for volatility was used to check for mutexes within the process. A few mutant entries were found within wininet, which were identified in Process Monitor as well. The results are shown in Figure 14 below.

$vol.py –f shylock.raw profile=WINXPSP3x86 handles –p 1468 –t Mutant --silent

Figure 14

Figure 15 shows process injections in explorer.exe. The malfind plugin for volatility is able to find a process injection since MZ is found in the header, which is a key that this was a process.

$vol.py –f shylock.raw profile=WINXPSP3x86 malfind –p 1468 | less

Chris Cain, cicain08@gmail.com
The yarascan plugin was used with volatility to find malicious IPs inside the explorer.exe process. Some links were found that attempted to reach a PHP file with the IP listed. Many Zeus variants have been known to run PHP scripts for updating their botnets. The results are shown in Figure 16.

From the analysis this malware has many characteristics that allow it to remain hidden from security software, while also having the ability to perform MITB style attacks. Shylock was found to have rootkit capabilities and have the ability to connect to malicious IP’s in an attempt to pull down configuration info from a central command server. The method of attack was to inject itself into the explorer.exe process and hide malicious processes.

Chris Cain, cicain08@gmail.com
4. Conclusion

There is no clear method in which to prevent MITB attacks beyond in-depth monitoring and prevention on the endpoint. Endpoint management that involves monitoring and preventing the browser from making changes to the system is one possibility to provide some defense against this attack. Many banks have even offered software that detects MITB type malware. Though, this is one layer to an attack that is continually evolving.

User education is mentioned as a method to prevent these attacks. In this case though user education isn’t enough. Trained security experts can be fooled just as easily as an end user by a well-crafted MITB script. Aside from not doing banking online there are many options that can be packaged together to lower the risk of this attack succeeding. A few educational topics to consider include configuring accounts with safeguards including secure notification options, checking account balances regularly, and using secure banks to do transactions.

Preventing browser extensions and scripting can also limit these types of attacks, or preventing scripts to run over SSL connections. There are methods in which to restrict browser extensions from running, though certain websites may not operate properly and restricting browsers is difficult in today’s age of multimedia operation. Banks have begun to use custom applications for banking on mobile devices to avoid any browser type intrusions. More of these apps may become popular as these attacks continue. Some banks have even offered to install anti-malware software on end users devices that would detect these types of attacks. This is debatable if this is good idea for banks to do, since attackers could use this as part of a phishing campaigns to install malware on users systems, posing as banks to install anti-malware software.

Transaction verification is also a popular method to counteract a Man-in-the-Browser (MITB) attack. This is also called Out of Band (OOB) transaction verification. Out of Band transaction verification is an additional method that verifies transactions such as a telephone call or an SMS text. This method has been known to get subverted as well if the verification information is stored in the user’s account online. If a user can change these details online then an attacker could change this information to a destination of their choosing without a user knowing. Many attackers have also begun using VoIP technologies to subvert Transaction verification via caller ID manipulation and cloned /recorded bank message alerts (Ollmann, 2008).

Chris Cain, cicain08@gmail.com
Three factor authentication using voice biometrics is another method banks have begun to use to further verify a transaction is valid (Hyderabad Hacker, 2011).

Banks have begun using Behavioral Analysis in their methods of defending against these attacks. Most credit card companies use this security feature to determine when potential fraud occurs in accounts currently. Detecting unusual wire transfers or transfers to international accounts typically throw up a red flag as an example of this type of detection.

Man-in-the-Browser attacks are not going to disappear anytime soon and will grow even more sophisticated. Potentially moving to mobile browsers as their use for banking is increased utilizing Man-in-the-mobile (MitMo) style attacks. Time will tell as the sophistication of these attacks not only target banking sites but other common sites that we have grown to trust.
5. References


Chris Cain, cicain08@gmail.com

Chris Cain, cicain08@gmail.com