Hookin' Ain't Easy

BeEF Injection with MITM

70 W. Madison Street, Suite 1050 Chicago, IL 60602 www.trustwave.com

Table of Contents

Overview

The ability to modify content transferred between two hosts using Man In The Middle (MITM) attacks is a well known network attack vector and is frequently used for sniffing network traffic or modifying traffic in transit. This attack is limited to conversations initiated by the client, leaving an attacker only able to view content that the target is accessing. This paper discusses a method to extend access to network resources by combining network attacks and browser based attacks to leverage the browser to extend an attacker's access to network resources.

What is Man In The Middle?

Man-In-The-Middle (MITM) is class of attack that allows the attacker to insert themselves into the middle of a communication stream between two or more parties. This attack can leverage a number of different technologies in order to accomplish this task, including ARP, DNS, DHCP, 802.11, and even WAN protocols such as BGP. Man-In-The-Middle attacks often lead to greater compromise, since the a successful attempt gives an attacker access to privileged traffic – a concern indeed in cases where encryption is absent.

One of the most common methods, known as Address Resolution Protocol (ARP) Poisoning, uses spoofed ARP packets to maliciously make changes to victim systems' ARP Cache. The ARP Cache maintains a Machine Address Code (MAC) address to IP mapping. By informing another local device, such as the default gateway, that the attacker's MAC is now hosting the client's IP address, and then in turn telling a local client that the gateway IP address is now being served by the attacker, both client and gateway will send traffic destined for each other to the target system.

With the traffic flowing through the attacker's system, traffic can be inspected and often manipulated. This type of attack can be used to steal credentials or other information via unencrypted channels. This is not a a new attack: tools such as sslstrip have been leveraged in the past to downgrade SSL requests in order to sniff and manipulate traffic that would typically be SSL encrypted. Indeed, ARP Poisoning is almost as old as the Ethernet switch, which as a side benefit acted as a deterrent to sniffing traffic that would have otherwise been broadcast by a hub.

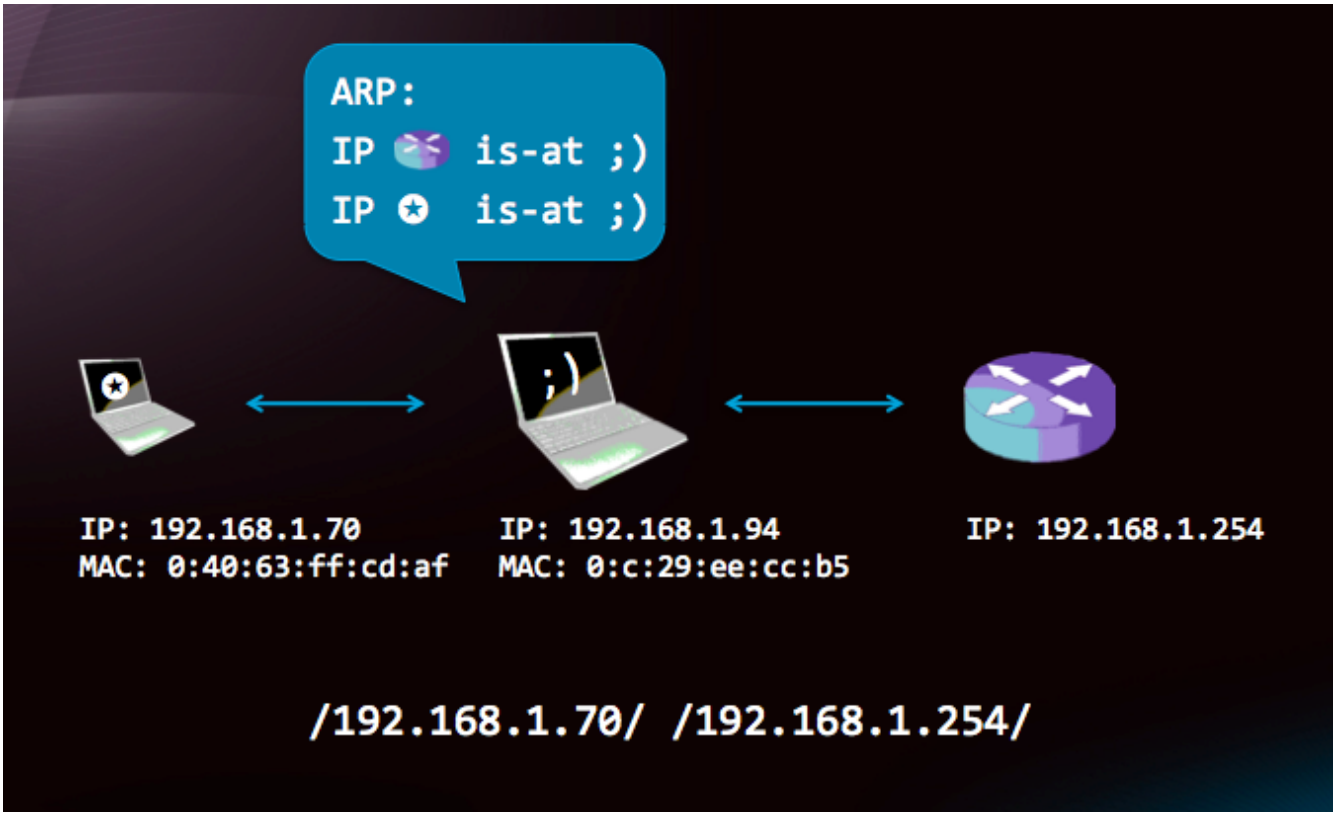

Figure 1. ARP Poisoning MITM attack

What is the Browser Exploitation Framework?

The Browser Exploitation Framework (BeEF) is a penetration testing tool written in Ruby and designed to both showcase browser weaknesses as well as perform attacks both on and through the web browser. BeEF consists of a server application that manages the connected clients, known as "zombies", and JavaScript "hooks" which run in the browser of target hosts.

Traditionally, the JavaScript hook is injected by the attacker into HTML code either through an attack such as Cross Site Scripting (XSS) or SQL Injection. Once the hook is processed by the browser, it beacons back home to the BeEF server, and will process JavaScript based commands sent from the BeEF server to the client.

The commands sent to the browser are triggered through modules running within the BeEF server. These modules send commands that do everything from fingerprinting browsers and plug-ins to allowing the attacker to proxy web traffic through the browser. Additional modules exist to perform tasks such as network scanning, browser keystroke logging, and cross protocol exploitation where HTTP requests can be sent to non-HTTP services with exploit payloads that will execute and return shells back to an attacker.

Current challenges with MITM

In addition to the common methods by which BeEF hooks are attached to browsers, this paper focuses on a third, less common approach: injecting the hook into a web page through MITM.

Current MITM tools are limited in their ability to influence traffic dynamically. Most tools that perform traffic modification do so either through filters, which will perform static transformations on packets, or through plug-ins that use task-specific code to perform those transformations. This limits the ability of an attacker to dynamically manipulate the packets that are traversing their own machine.

If a new task is desired, then a different plugin must be run, or a different filter must be created to perform a new transformation. In addition, the MITM tools are not state aware and as such, it's difficult to create rules that will only transform packets once or at regular intervals. For that level of functionality, custom code must be written.

The two most popular tools for ARP Spoofing are Ettercap and Cain and Abel. Ettercap is a multi-platform tool that utilizes two popular UNIX network libraries, libpcap and libnet, to interface with the network. Packet manipulation can occur in two ways, the first being through plug-ins written in C and compiled into a shared object that can be dynamically loaded at run-time. The second method is to create a filter through the built-in scripting language that then must be compiled into an Ettercap filter for usage at execution time. In order to modify filter functionality it must be modified and then recompiled.

Cain and Abel is a Windows tool suite that allows for sniffing traffic through MITM and then cracking the captured credentials. While Cain and Able will capture dozens of types of credentials, it does not have the ability to create filters or modify traffic, and as such, unless credentials are sent through the intercepted traffic, it has limited ability to coerce systems into disclosing credentials for other services.

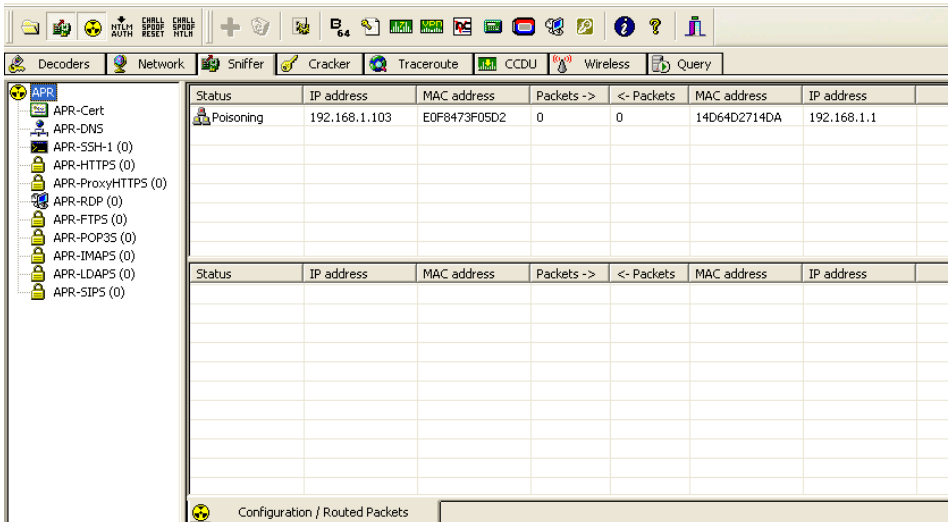

Figure 2. ARP Poisoning with Cain & Abel

The key to overcoming these problems is the ability to inject a dynamic element through the MITM attack. This will allow the attacker to both execute multiple secondary attacks selectively, as well as coercing web and network elements into disclosing credentials that have been browser cached.

Combining Network and Browser Attacks

As network applications move to web applications, the amount of information accessed by the browser is increasing. In corporate environments, the browser is even a more powerful asset as it has access to Windows Integrated Authentication credentials. By leveraging the browser functionality, an attacker can fingerprint resources to increase the likelihood that attacks will go un-noticed and take advantage of the powerful AJAX and HTML5 functionality, which in turn facilitates more sophisticated browser based attacks. To begin to leverage this functionality, an attacker must first hook the browser.

Introducing Shank

Shank is a Ruby based MITM tool that integrates tightly with BeEF in order to maximize the value of MITM attacks. Using PacketFu modules, Shank has the ability to target both systems that are currently on the broadcast network as well as new hosts that come online during poisoning. By interacting with BeEF to determine if a browser is currently hooked, Shank is able to selectively inject a hook into non-hooked resources. This decreases the chance that the browser manipulation will be obvious to the user browsing sites, and also prevents the BeEF server from processing duplicate hooks.

Shank has the ability to communicate to the BeEF REST API in order to dynamically retrieve the list of hooked browsers. By periodically repopulating that list, Shank can determine which hosts it should target for injection. Once a host has been targeted for injection, HTTP sessions are inspected and encoding is downgraded to ensure that content encoding such as gzip will not be present. This allows Shank to insert data into packets without having to gather multiple packets at a time, increasing speed and precision.

With the encoding downgraded, Shank targets HTML tags that indicate suitable insertion points for a BeEF hook. Shank searches for HEAD, BODY, and IMG tags to indicate that the page is designed to be viewed within a browser and then inserts a SCRIPT tag reference containing the URL to the BeEF JavaScript hook. The BeEF hook as well as the inserted code is designed to minimize the visual impact of the website in order to hide the presence of the hook and maintain control over the browser as long as possible.

Managing Browsers with BeEF

Once the JavaScript code is processed by a browser, a beacon is started between the browser and the BeEF server using the BeEF hook. When the initial BeEF hook is established, some basic information about the browser is gathered and BeEF stores this information in the local database. The browser is then added to the list of "Zombies", or controlled browsers. BeEF then maintains these hooks and waits for the attacker to send modules to individual Zombies.

This method works well for a small handful of Zombies. However, with larger numbers of Zombies, manually running desired modules against each one becomes tedious. The REST API of BeEF has come to the rescue to allow these Zombies to be managed automatically. BeEF only allows a single auto-run module to be set, reducing the ability of an attacker to create a series of modules that will run on new Zombies. Using the REST API, it is possible to create an autorun script that will be able to launch a series of modules against new Zombies.

Using the autorun script, the attacker can do multiple profiling steps and information gathering attacks on browsers as they initially become Zombies. Polling BeEF through the REST interface, as a new Zombie comes online, a chain of modules is run against a browser to determine plug-ins installed, current page, and then a number of hidden iframes are launched for additional information gathering through other tools. This includes using Metasploit to capture Windows Integrated Authentication credentials and potentially targeting specific browsers for Metasploit browser based exploits.

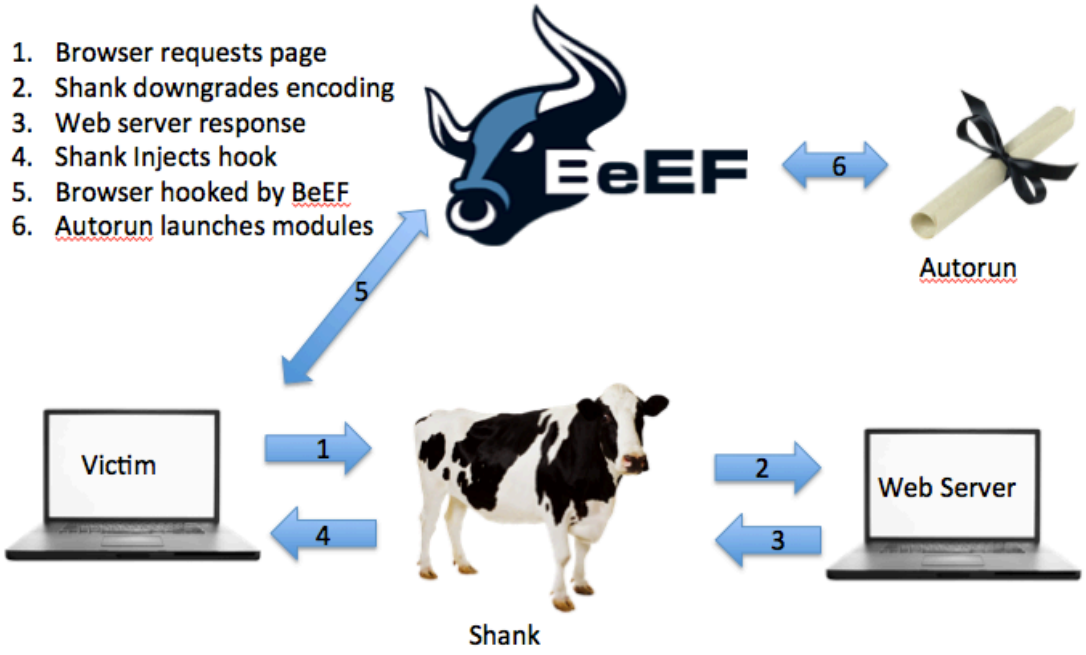

Figure 3. Attack Workflow

As this logic is all done using Ruby, automatic targeting can happen using common Java and other plug-in exploits to ensure that only browsers that may be vulnerable will be targeted. Based on the website visited,

additional plug-ins can be targeted for browser keystroke logging, submission target rewrites, or other types of attacks aimed at stealing credentials.

Once it is known what sites and credentials a browser has, it becomes a target for additional types of attacks. BeEF allows an attacker to use a Zombie as a proxy host. Once the host has been chosen, it can be used to fingerprint other systems using the integrated credentials within the browser. By chaining other tools such as Burp or ZAP proxy, scanning can be done of intranet sites that would normally require authentication. A browser can also be pointed directly at the proxy and internal files retrieved, or protected internal systems accessed by an attacker.

Prevention

The proof of concept presented here is meant to illustrate the extent of access that an attacker can gain via browser hooks. Browser attacks are generally thought of as having server ramifications only. XSS and session stealing all have a negative impact on the user, but the attack domain is actually focused on the server being connected to, and is limited to that site. In addition, these types of attacks usually require some form of social engineering to be successful. XSS requires a user to click on a link, as do many types of malicious browser payloads.

The combination of BeEF, which proves that browser-based attacks can persist beyond a single domain, and Shank, which eliminates the need for user intervention in order to compromise browsers, proves that this is an area that requires additional scrutiny.

SSL and other encryption methods such as VPN and other IPSec implementations such as DirectAccess have the capability to prevent this type of attack, if used consistently. As was the case with SSLStrip, this latest version of the MITM browser attack further demonstrates that the HTTPS URI should be used exclusively wherever possible. Sites that employ plaintext HTTP on any page, even pre-authentication, are subject to this type of attack vector and steps should be taken to protect users from this type of activity.

It is the hope of the authors that this tool is used by penetration testers to further demonstrate this attack method and can clearly demonstrate the importance of whole-site data encryption.

Conclusion

By utilizing a Man-In-The-Middle tool such as Shank alongside BeEF, an attacker can leverage web browsers on the local network to perform more complicated attacks. This technique can be used to gather Windows Domain credentials, capture browser keystrokes, and act as a proxy that an attacker can go through for attacking websites where a target has authenticated. This technique greatly extends an attacker's ability to manipulate traffic through MITM techniques -- from viewing and changing websites the target is browsing, to allowing an attacker to request web pages through the target, on the target's behalf.

Through the optimizations described in this paper and demonstrated by the accompanying proof-of-concept tools, managing larger numbers of Zombies becomes more practical, and the ability to rapidly profile large groups of hosts also becomes possible. These tools are all using Ruby with readily available modules, making these open source tools accessible for penetration testers and security researchers to demonstrate the cumulative impact of browser and network vulnerabilities.

References

Browser Exploitation Framework

- URL: http://www.beefproject.com/
- Rest API Information: https://github.com/beefproject/beef/wiki/BeEF-RESTful-API

Metasploit Framework

- URL: http://www.metasploit.org/

Ettercap

- URL: http://ettercap.sourceforge.net/

Cain & Abel

- URL: http://www.oxid.it/cain.html

About the Authors

Ryan Linn

Ryan Linn is a Senior Consultant with Trustwave's SpiderLabs - the advanced security team focused on penetration testing, incident response, and application security. Ryan is a penetration tester, an author, a developer, and an educator. He comes from a systems administration and Web application development background, with many years of IT security experience. Ryan currently works as a full-time penetration tester and is a regular contributor to open source projects including Metasploit and BeEF, the Browser Exploitation Framework.

Steve Ocepek

Steve Ocepek serves as a Senior Security Research Manager for Trustwave's SpiderLabs division - the advanced security team focused on penetration testing, incident response, and application security. An innovative network security expert with an entrepreneurial spirit, Steve Ocepek has been a driving force in pioneering Network Access Control (NAC) technologies delivering comprehensive endpoint control for mitigation of zero attacks, policy enforcement, and access management, for which he has been awarded 4 patents with 1 patent pending.

With a reputation for preventing, intercepting, and resolving malicious attacks from malware, viruses, and worms, Steve has provided consultative testing, and made recommendations for remediation for Fortune 500 and government enterprises in financial, credit card processing, educational, healthcare, and high-tech industries. His testing of network penetration, use of Network Access Control (NAC), Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), Web Application Firewalls (WAF), Network Firewalls, and Encryption Solutions enable him to advise on new countermeasures improving security, saving clients millions of dollars in losses of intellectual property, client data, customer confidence, and litigation costs.

Steve has led the growth of SpiderLabs Security Research Department, more than doubling services providing solutions to meet the needs of clients worldwide in identifying, preventing, and solving network security threats and problems. He is known as a trusted resource and problem solver by chief information officers, directors of security, chief technical officers, chief operating officers, chief executive officers, and military and national security leaders.

About Trustwave

Trustwave is a leading provider of compliance, Web, application, network and data security solutions delivered through the cloud, managed security services, software and appliances. For organizations faced with today's challenging data security and compliance environment, Trustwave provides a unique approach with comprehensive solutions that include its TrustKeeper® portal and other proprietary security solutions. Trustwave has helped hundreds of thousands of organizations--ranging from Fortune 500 businesses and large financial institutions to small and medium-sized retailers--manage compliance and secure their network infrastructures, data communications and critical information assets. Trustwave is headquartered in Chicago with offices worldwide. For more information, visit https://www.trustwave.com.

About Trustwave SpiderLabs

SpiderLabs is the advanced security team within Trustwave focused on forensics, ethical hacking and application security testing for our premier clients. The team has performed hundreds of forensic investigations, thousands of ethical hacking exercises and hundreds of application security tests globally. In addition, the SpiderLabs Research team provides intelligence through bleeding-edge research and proof of concept tool development to enhance Trustwave's products and services. For more information, visit https://www.trustwave.com/spiderLabs.php.

Contact

Corporate Headquarters

70 West Madison St.
Suite 1050
Chicago, IL 60602

P: 312.873.7500
F: 312.443.8028

EMEA Headquarters

Westminster Tower
3 Albert Embankment
London SE1 7SP

P: +44 (0) 845 456 9611
F: +44 (0) 845 456 9612

LAC Headquarters

Rua Cincinato Braga,
340 nº 71
Edificio Delta Plaza
Bairro Bela Vista -
São Paulo - SP
CEP: 01333-010 - BRASIL

P: +55 (11) 4064-6101

APAC Headquarters

Level 26
44 Market Street
Sydney NSW 2000,
Australia

P: +61 2 9089 8870
F: +61 2 9089 8989