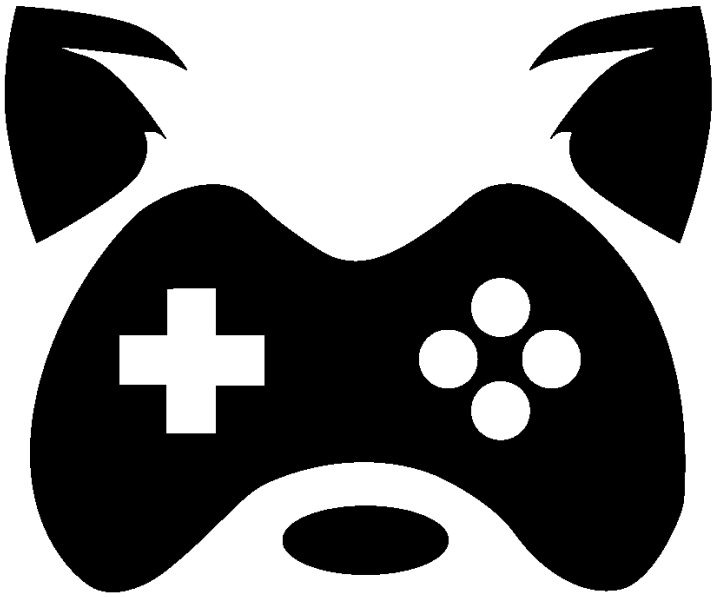

Raccoon Security Services

KENOBI PENETRATION TEST

October 1, 2021

TABLE OF CONTENTS

Executive Summary

Raccoon Security Services was contacted by Kenobi to conduct a penetration test against a new server in their production setup. The goal of this test was to gain remote root access and enumerate as much data as possible without disturbing the expected load on the server. The information given for the penetration test was the IP address of the server to be attacked and the mention of FTP and SMB services. The server of IP address 10.10.39.18 is accessible from the internet and does not require a VPN or access through a firewall. During the penetration test over a weeklong period, 4 vulnerabilities were found and exploited to gain root access, along with a few other potential vectors that lie unpursued only the dangerous of which are listed. The exploits used allow any unauthenticated remote connection to retrieve any sensitive data or make any changes with permissions as root.

Kenobi's server is vulnerable to primarily networking misconfigurations and outdated services. Among these the most critical is the outdated FTP service which allows for remote root control of copying data to more accessible locations. The lack of authentication required to the FTP and SMB services could indicate the absence of central network authentication such as a Windows domain or LDAP server. It is recommended to use such authentication methods to control what accounts are allowed access to services, and what requirements exist for password complexity across services.

The penetration test conducted uses vulnerabilities found by our internal testing team, and this report labels remedies for vectors regarding those findings, however the remediation listed should not be considered all the action needed to properly secure the server or the network/environment at Kenobi. At minimum it is recommended to update the services currently being used, require authentication to access FTP/SMB services, and restrict network share access to specific hosts.

Test Scope

The following IP addresses were authorized for enumeration and penetration testing:

- 10.10.39.18 – Internet Facing Server

The penetration test was conducted in three stages:

1. Information Gathering and Vulnerability Identification: where the internal testing team enumerated data and scanned for vulnerabilities of the remote server.
2. Attacking and Exploiting: where the internal testing team exploited found vulnerabilities and replotted their course once a foothold was achieved.
3. Cleanup: the internal testing team removes their tools and code from the targets in question or writes down what may be left and ought to be removed by the proper authority at Kenobi.

The following rules were set in place for the penetration test:

1. Raccoon Security Services is not to crash, DDOS, or otherwise affect the uptime or availability of the Kenobi server to be tested
2. Social engineering is off limits to any personnel working at Kenobi
3. No Physical Access is to be granted or permitted to conduct physical security analysis
4. Should any evidence of a previous breach be found notify the point of contact immediately and continue the test as planned
5. Any sensitive data found should be encrypted and stored on a media device to be physically given to the point of contact upon presenting the results. Keeping the data even if encrypted past the penetration test is not permitted and will be prosecuted to the fullest extent of the law
6. Once finished the point of contact is to be notified of its completion and if there is removal that needs to take place regarding a footprint from the testing team

Summary of Findings

The vulnerabilities found are only labeled to mirror the Test Result section. Though these vulnerabilities might have different severities, the section is only to convey the scale of the vulnerabilities found rather than the importance of prioritization.

Services and ports were found during the enumeration phase as depicted below:

PORT	Service Type	Identified Service	Additional Info
21	ftp	ProFTPD 1.3.5	
22	ssh	OpenSSH 7.2p2	
80	http	Apache 2.4.18	Image in webpage
111	rpc	RPCbind	NFS mounted /var
139	smb	SMB Samba 3.X – 4.X	
445	smb	SMB Samba 4.3.11	\anonymous & \IPC$ shares
2049	nfs	nfs_acl	
48337	rpc	mountd	

The following vulnerabilities were discovered during the Penetration Test:

#	Finding	Severity
1	Unauthenticated Remote SMB Share Access	Medium
2	Unauthenticated File Copying as Root	Critical
3	Unrestricted Network Share Access	Low
4	SUID Arbitrary Code Execution	Critical
5a	Memory Corruption/Remote Code Execution	Medium/Critical

Penetration Test Results

1. Unauthenticated Remote SMB Share Access

Severity: Medium

Details:

Our internal testing team was able to access the SMB shares on the server without any authentication. Inside the anonymous share was a log.txt file which contained information to further gain ground on rooting the server.

Recommendation:

If the SMB server is meant to be used by a stable few people those individuals should have accounts with passwords to access the SMB service. In the event the permanence of accounts is uncertain, Samba can be configured to allow access to specific accounts or clients designated by a network authentication server (LDAP, Windows Domain, etc.). In addition, the 'Null Passwords' global attribute can be set to 'No' to force users with blank password to be unable to log in.

Should more security be a concern Samba can be configured to allow access only from certain network hosts and certain interfaces on the server. The 'Bind Interfaces Only' global attribute can be set to 'Yes' and an 'interfaces' global attribute can be added and filled with the interface addresses you wish traffic to be restricted to. In a similar fashion the 'Hosts Allow' global attribute can be added and filled with allowed hosts to remotely connect.

2. Unauthenticated File Copying as Root

Severity: Critical

Details:

The FTP service ProFTPD 1.3.5 on port 21 contains an unauthenticated method to copy files from any location to another as root. These permissions extend to the .ssh directory in any home directory, allowing access to copy the private ssh key to an external media device or accessible network share. Our internal testing team performed the latter, using the built in CPFR and CPTO commands to move the ssh key to the mounted NFS share, then retrieve Kenobi's key by remotely mounting the NFS share on port 111. The following actions allowed for remote user access but could lead to sensitive data breach under the right circumstances.

Recommendation:

An update of ProFTPD version to the recent version 1.3.7 would fix this vulnerability. In general, keeping services, software, and operating systems updated to the latest versions available increases security by a considerable amount.

3. Unrestricted Network Share Access

Severity: Low

Details:

NFS shares mounted by the server can be remotely mounted by any host. This gives a remote client access to any files or information held within the NFS share. In Kenobi's case, /var is mounted on the NFS share which is a common place to hold website information, databases, and other data prone to change. If this server were to host a website, this share could be remotely mounted to read and retrieve important site files such as index.php.

Recommendation:

Restricting what IP addresses are allowed to mount the exposed shares is a change that should fix this vulnerability. If the NFS share is not required to perform root functions it is recommended to enable the "root_squash" option, though if it is uncertain that the share needs to use root permissions it could be left unchanged.

4. SUID Arbitrary Code Execution

Severity: Critical

Details:

The /bin/menu command improperly uses curl and other command line tools within the code. When writing custom SUID tools the commands to be used point directly to the directory they are held within. For example, if I wished to use python in my code I would point to /usr/bin/python to do so. Neglecting to point towards the directory results in an attacker having the opportunity to change the $PATH variable and run arbitrary code simply by naming their new command the same as one invoked within the SUID. Our internal testing team added a new $PATH location and made a custom curl command to gain a root shell.

Recommendation:

SUIDs that run as root should point directly to the command in question e.g., /usr/bin/curl to run curl in the case of the menu SUID.

5. Unused Potential Vulnerabilities

5a. Memory Corruption/Remote Code Execution

Severity: Medium/Critical

Details:

Left unused was the port 139 potential buffer overflow vulnerability. A buffer overflow can take the place of two types of attack: memory corruption/server crashing, and remote code execution (RCE). Memory corruption can affect availability and uptime but the danger from this vulnerability comes from the latter use. With advanced knowledge of the service used and some testing it is possible to find the length of the buffer in memory and write code into specific addresses. The way to test if this version is vulnerable to buffer overflow attacks is to fuzz the buffer until the service crashes. Actions such as these were out of the "without disturbing the expected load on the server" part of the scope.

Recommendation:

The concise way to determine if this vulnerability exists is fuzzing the buffer and attempting to crash the service. The following python script could be used to fuzz the buffer of the port in question on the server IP given:

```
#!/usr/bin/python3
import sys, socket
from time import sleep

buffer = "A" * 100


while True:
          try:
                    s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                    s.connect(('10.10.39.18',139))

                    s.send(('COMMAND ' + buffer))
                    s.close()
                    sleep(1)
                    buffer = buffer + "A" * 100

          except:
                    print("Fuzzing crashed at %s bytes" % str(len(buffer)))
                    sys.exit()
```

The blue COMMAND section is the command desired to check buffer overflow against.

Should the service crash upon running this command it is recommended to update the service listed to a more recent version likely unaffected by the innate problem. Though many ways exist to detecting and mitigating buffer overflow attacks, many are mostly useful when developing an application or service in-house, whereas using common services such as Samba limit the granularity of one's options to fight these vulnerabilities.