DIMVA 2018

Update State Tampering

: a Novel Adversary Post-compromise Technique on Cyber Threats

Sung-Jin Kim, Byung-Joon Kim, Hyoung-Chun Kim
@ National Security Research Institute of South Korea
Dong Hoon Lee
@ Korea University

Topics

This presentation contains the following

- Blind spots on the Windows update management
- Update state tampering attacks
  - A reinfection scenario after system recovery
- Ways to monitor the blind spots
  - A PowerShell script for detections

Contents

- Introduction
- Background
- Problems of Windows Update Management
- Attack: Update State Tampering
- Countermeasures
- Discussion & Conclusion

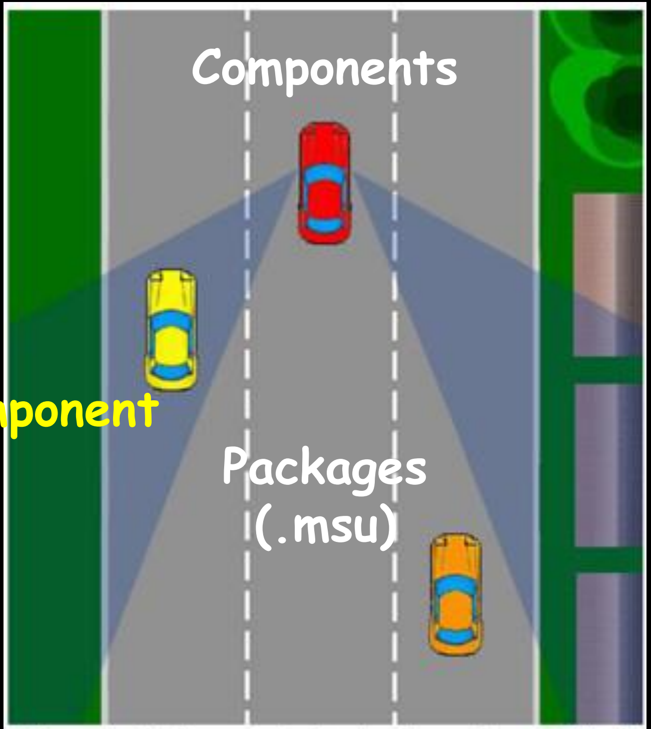
Introduction

What happens when update management mechanism does not work properly?

- *A package-component mismatch* may occur!

Can we notice or correct the problem after things going?

- *Nope...*
- *Blind spots* on the windows update management



Components

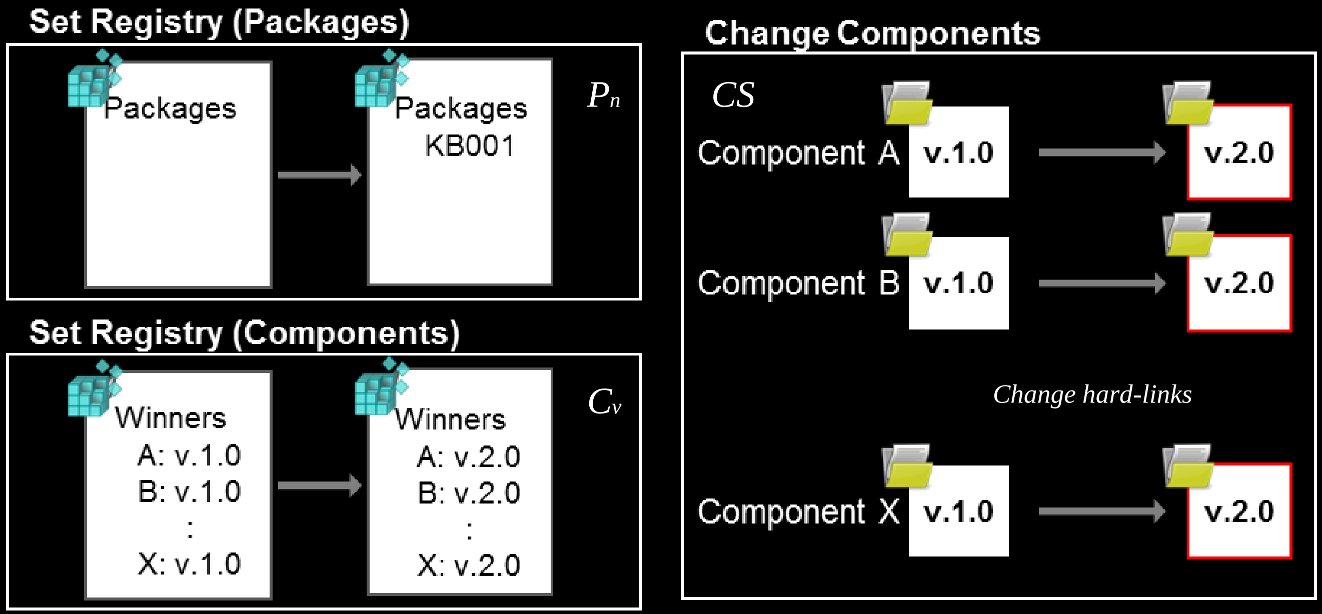
Package-Component Mismatch

Packages (.msu)

Blue shaded areas indicate the red cars blind spot

Background

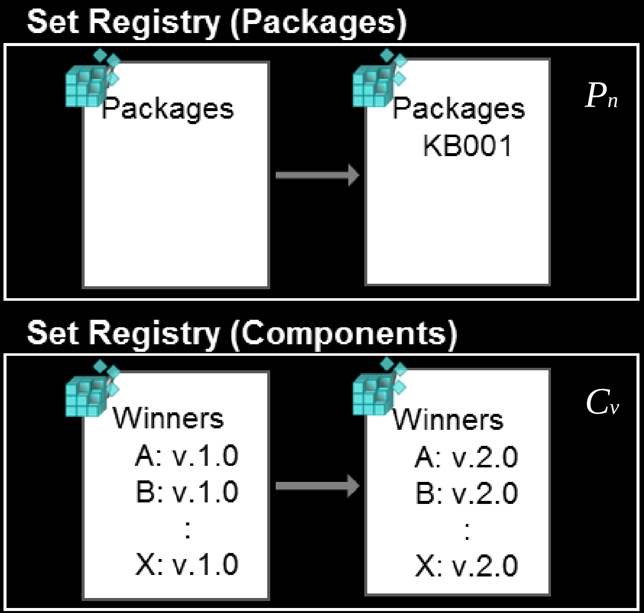
What Does Windows Update Do?



What Does Windows Update Do?

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Component based servicing\Packages\[*Package Name*]

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\SideBySide\Winners\[*Component Name*]\[*Windows Version*]\(default)



Set Registry (Packages)

Packages

Packages
KB001

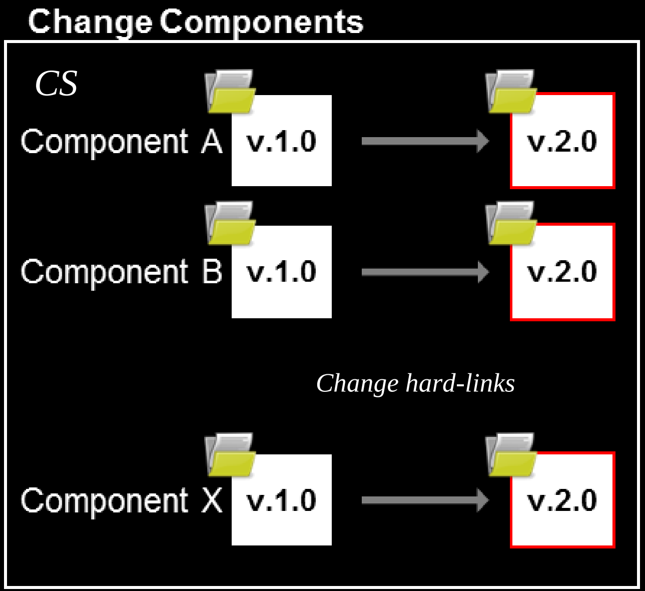
P_n

Set Registry (Components)

Winners
A: v.1.0
B: v.1.0
:
X: v.1.0

Winners
A: v.2.0
B: v.2.0
:
X: v.2.0

C_v



Change Components

CS

Component A v.1.0 v.2.0

Component B v.1.0 v.2.0

Change hard-links

Component X v.1.0 v.2.0

\Windows\WinSxS\[*Component Name*]\[*File name*]

Component-Based Servicing

Component

- The small grouping of files based on a feature area, functionality, and reusability

Servicing

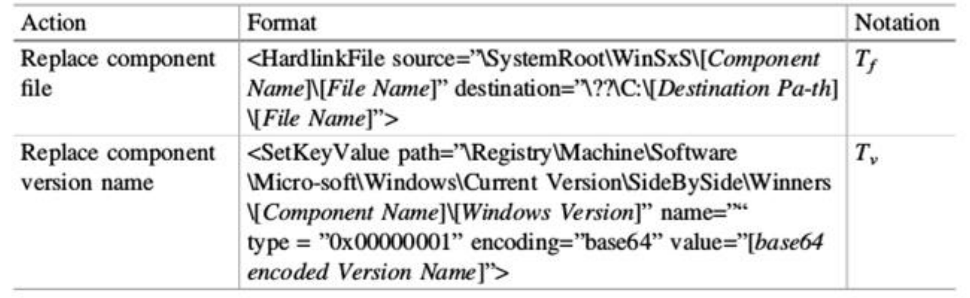
- The act of installing a role, feature, service pack or windows update against a Windows OS
- *TrustedInstaller.exe* is the servicing agent

The CBS provides a more robust installation process than the file-based servicing, while simultaneously mitigating against instability issues caused by improper or partial installation

PendingFileRenameOperations

In the last step of update..

- The system puts the replacement of kernel components at the point after the reboot event
- The TrustedInstaller.exe generates the *pending.xml* file
- *poqexec.exe* performs pending jobs that are listed on the pending.xml



Action	Format	Notation
Replace component file	<HardlinkFile source=”\SystemRoot\WinSxS\[*Component Name*]\[*File Name*]” destination=”\??\C:\[*Destination Pa-th*]\[*File Name*]”>	T_f
Replace component version name	<SetKeyValue path=”\Registry\Machine\Software\Micro-soft\Windows\Current Version\SideBySide\Winners\[*Component Name*]\[*Windows Version*]” name=”“ type = ”0x00000001” encoding=”base64” value=”[*base64 encoded Version Name*]”>	T_v

Problems of Windows Update Management

Package -Component Mismatch

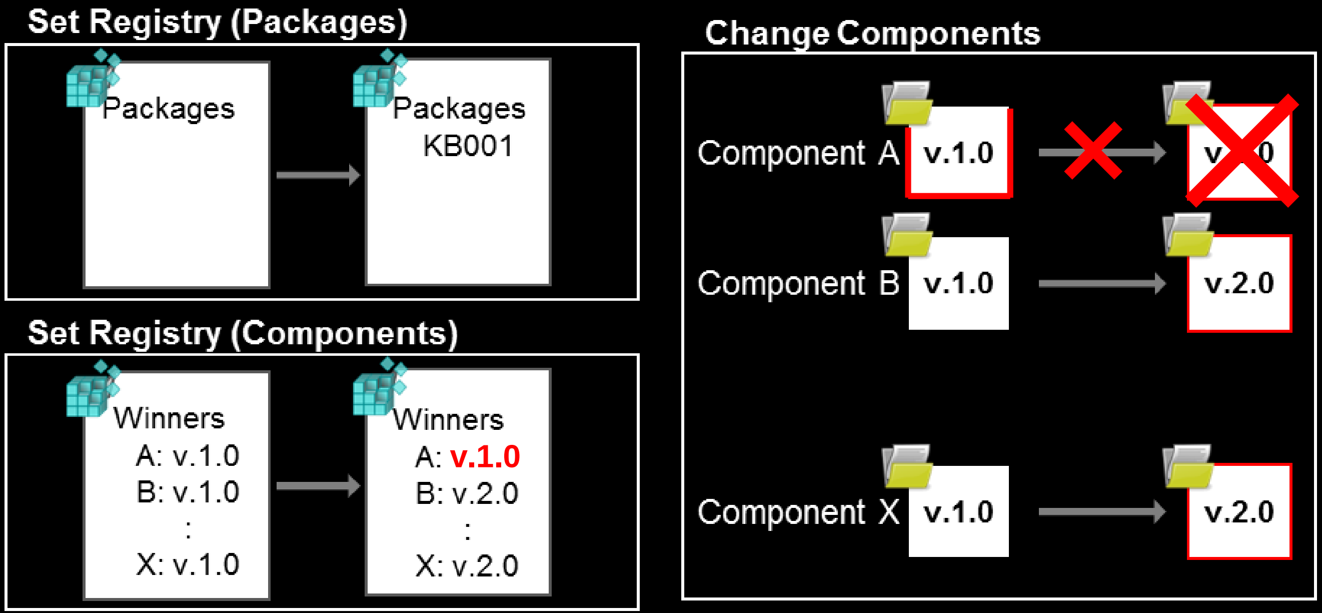
We define it as

- **A state in which a part of component is different from the system that is usually updated**

The result from the package-component mismatch

- A part of the components *can be replaced with the previous versions* that contain *known vulnerabilities*
- At the same time, the update history remains unchanged

Package -Component Mismatch



Blind Spots?

After things going...

- There is nothing to diagnose the package-component mismatch

Two types of blind spots

- *(Type I)* The system loads the components that do not match the current update state
- *(Type II)* The system does not provide a means to detect or fix update status abnormalities

Blind Spot 1 (Type Ⅰ)

What does Code Integrity Policy do?

- Windows can only load components with valid digital signatures

However,

- The package-component consistency is not a concern for the code integrity policy
- Regardless of the fact that a component has vulnerabilities, every core component that Microsoft provides has a valid digital signature

Eventually, the system loads the components that do not match the current update state.

Blind Spot 2 (Type Ⅱ)

What does Update Check do?

1. Check the version of the update agent
2. Collect package information installed on the system
3. Download and install packages

However,

- Check package information only
- Does not care about component information

Eventually, the Update Check cannot detect update status abnormalities

Blind Spot 3 (Type Ⅱ)

What does SFC (System File Check) do?

1. Check the integrity of components
2. Detect component damages
3. Repair corrupted components

However,

- Check component information only
- Does not take into account the current update status

Eventually, the SFC cannot detect update status abnormalities

Impact on Windows Platforms

Impact of the blind spots on Windows

Version	Blind Spot1	Blind Spot2	Blind Spot3
Windows 7	O	O	O
Windows 8	O	O	O
Windows 8.1	O	O	O
windows 10	O	O	O

Version	Blind Spot1	Blind Spot2	Blind Spot3
Windows Server 2008	O	O	O
Windows Server 2012	O	O	O
Windows Server 2012 R2	O	O	O
Windows Server 2016	O	O	O

This issue affect not only the desktop users, but also the enterprise environment.

Attack: Update State Tampering Attack

Goal

The Goal of Update State Tampering

- Replacing current components with previous versions that have vulnerabilities while maintaining the record of updates

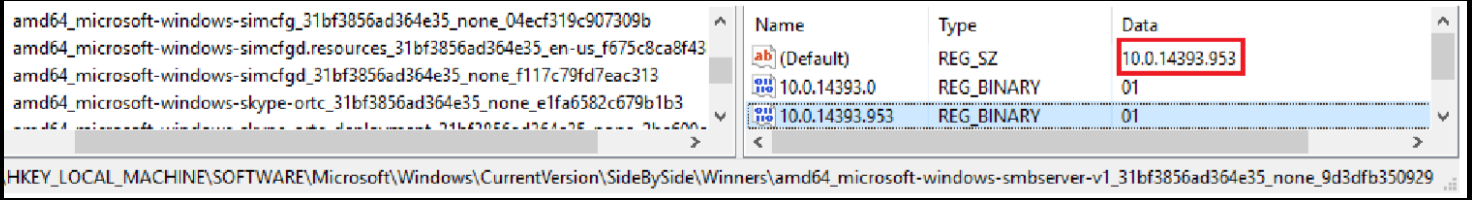
Assumptions

- Administrative privileges are required for the Update State Tampering
- Use the proposed methods with known exploits or 0-days

A Toy Example

- 1. Identify the Target Component

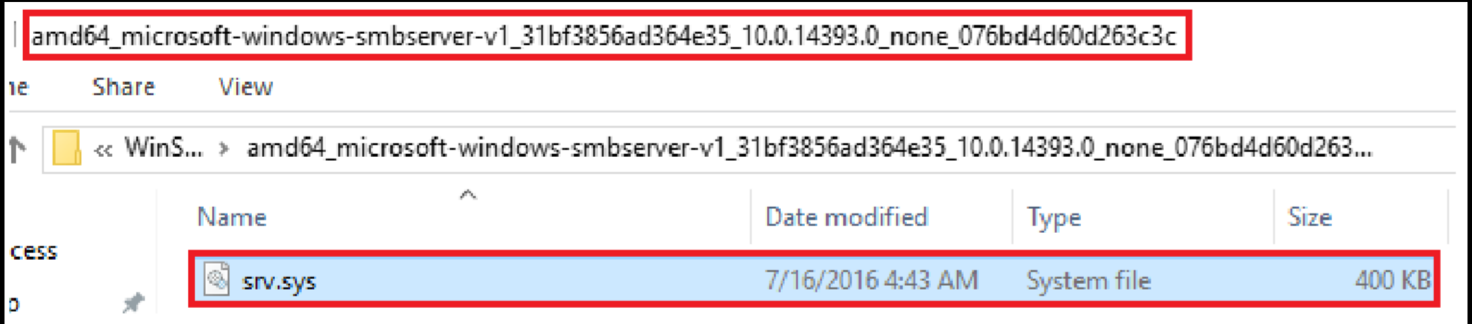
Replace the value of C_v



amd64_microsoft-windows-simcfg_31bf3856ad364e35_none_04ecf319c907309b
amd64_microsoft-windows-simcfgd.resources_31bf3856ad364e35_en-us_f675c8ca8f43
amd64_microsoft-windows-simcfgd_31bf3856ad364e35_none_f117c79fd7eac313
amd64_microsoft-windows-skype-ortc_31bf3856ad364e35_none_e1fa6582c679b1b3

Name	Type	Data
ab (Default)	REG_SZ	10.0.14393.953
10.0.14393.0	REG_BINARY	01
10.0.14393.953	REG_BINARY	01

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\SideBySide\Winners\amd64_microsoft-windows-smbserver-v1_31bf3856ad364e35_none_9d3dfb350929



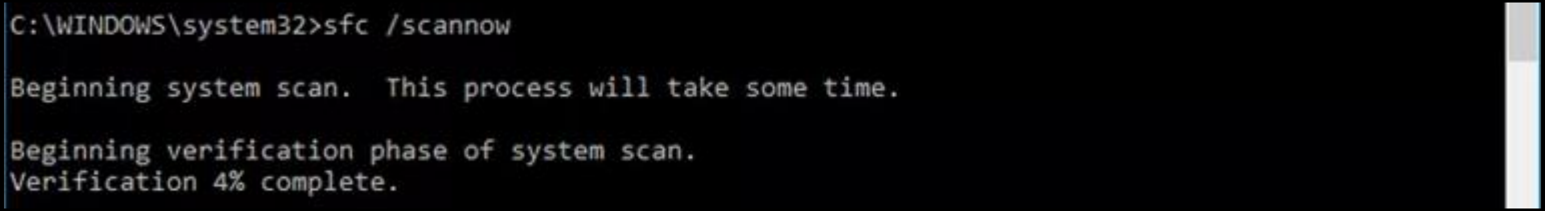
amd64_microsoft-windows-smbserver-v1_31bf3856ad364e35_10.0.14393.0_none_076bd4d60d263c3c

Share
View
« WinS... › amd64_microsoft-windows-smbserver-v1_31bf3856ad364e35_10.0.14393.0_none_076bd4d60d263...

Name	Date modified	Type	Size
srv.sys	7/16/2016 4:43 AM	System file	400 KB

A Toy Example
- 2. Use SFC Tool

Let poqexec.exe perform component replacement



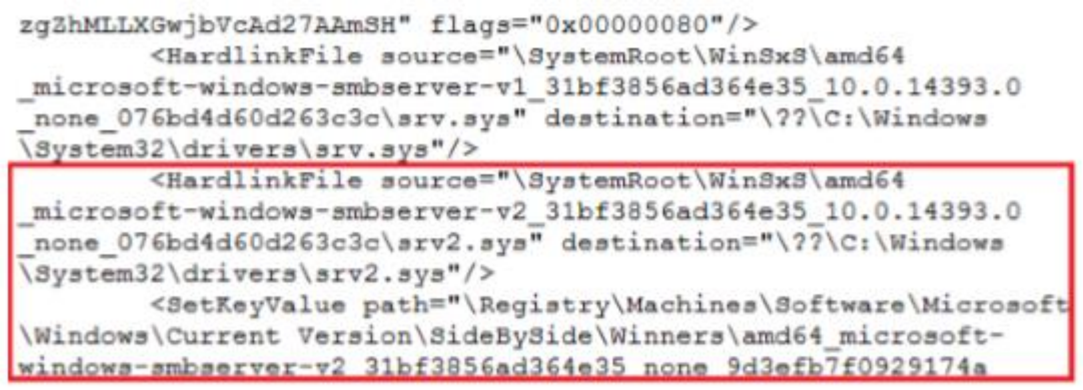
```
C:\WINDOWS\system32>sfc /scannow

Beginning system scan.  This process will take some time.

Beginning verification phase of system scan.
Verification 4% complete.
```

A Toy Example
- 3. Tamper at a Single Point

Insert additional pending jobs in the pending.xml file



```
zgZhMLLXGwjbVcAd27AAmSH" flags="0x00000080"/>
        <HardlinkFile source="\SystemRoot\WinSxS\amd64
_microsoft-windows-smbserver-v1_31bf3856ad364e35_10.0.14393.0
_none_076bd4d60d263c3c\srv.sys" destination="\??\C:\Windows
\System32\drivers\srv.sys"/>
        <HardlinkFile source="\SystemRoot\WinSxS\amd64
_microsoft-windows-smbserver-v2_31bf3856ad364e35_10.0.14393.0
_none_076bd4d60d263c3c\srv2.sys" destination="\??\C:\Windows
\System32\drivers\srv2.sys"/>
        <SetKeyValue path="\Registry\Machines\Software\Microsoft
\Windows\Current Version\SideBySide\Winners\amd64_microsoft-
windows-smbserver-v2_31bf3856ad364e35_none_9d3efb7f0929174a
```

Inserted component rollback tasks (srv2.sys) in the pending.xml file

A Toy Example
- 4. Reboot System

Just reboot the system,

The vulnerability will be hidden.

Can We Detect Them with Existing Tools?

Microsoft Baseline Security Analyzer

- A free security compliance tool that detects insecure configurations and missing security updates

MyPCInspector

- A third-party security compliance tool used by Korean government offices to check the security status of Windows-based PCs

They show similar behavior to the Windows Update Check.
Thus, They cannot detect the package-component mismatches

Method 1 - Using SFC to tamper with the update state

The procedure of method 1

1. Change the target component version C_v
   - “HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\SideBy Side\Winners\[Component Name]\[Windows Version]\(default).”
2. Run the “SFC /SCANNOW” command
   - The SFC generates the pending.xml to replace target components
3. Reboot the system
   - The target components will be replaced without affecting to the update history

Method 2 - Leave Components Unchanged

The procedure of method 2

1. Delete the tags T_f and T_v from the pending.xml
    - <HardlinkFile source="\SystemRoot\WinSxS\[Component Name]\[System File Name]" destination="\??\C:\[Destination Path]\[System File Name]">
    - <SetKeyValue path="\Registry\Machine\Software\Microsoft\Windows\CurrentVersion\SideBySide\Winners\[Component Name]\[Windows Version]" name=" " type="0x00000001" encoding="base64" value="[base64 encoded Version Name]">
2. Reboot the system
    - The target components will not be replaced, but package information is updated normally

Method 3 - Revert Component to the past

The procedure of method 3

1. Insert the target component replacement tags to the pending.xml
   - <HardlinkFile source="\SystemRoot\WinSxS\[Component Name]\[System File Name]" destination="\??\C:\[Destination Path]\[System File Name]">
   - <SetKeyValue path="\Registry\Machine\Software\Microsoft\Windows\CurrentVersion\SideBySide\Winners\[Component Name]\[Windows Version]" name=" " type="0x00000001" encoding="base64" value="[base64 encoded Version Name]">
2. Reboot the system
   - Target components will be replaced without affecting to the update history

Impact on Windows Platforms

Impact of the identified methods on Windows

Version	Method 1	Method 2	Method 3
Windows 7	O	O	O
Windows 8	O	O	O
Windows 8.1	O	X	X
windows 10	O	X	X

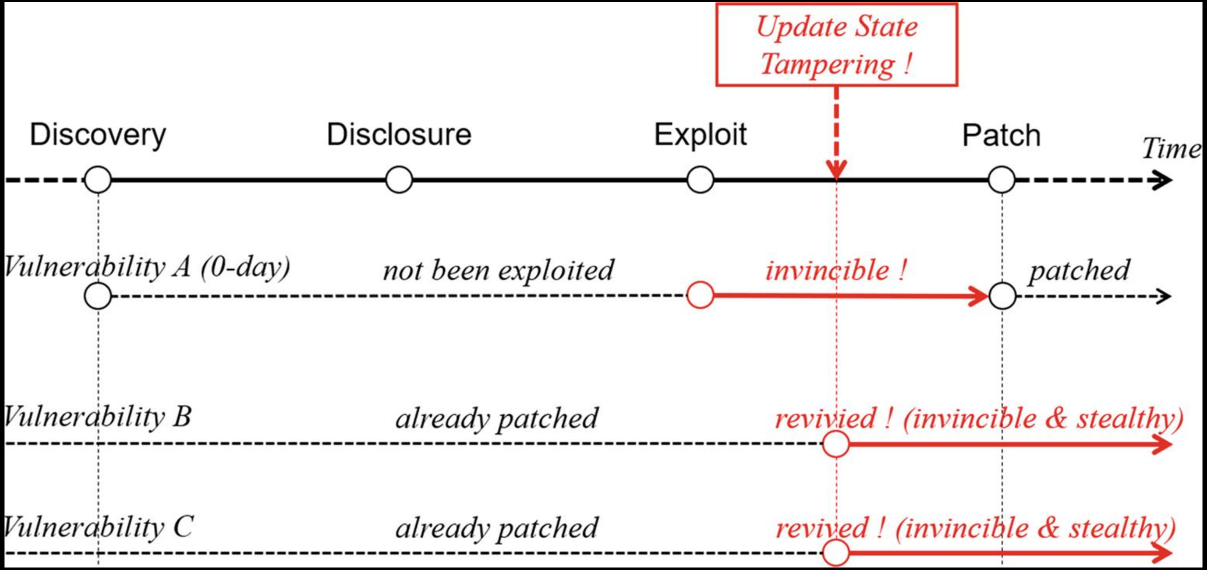
Version	Method 1	Method 2	Method 3
Windows Server 2008	O	O	O
Windows Server 2012	O	O	O
Windows Server 2012 R2	O	X	X
Windows Server 2016	O	X	X

- Method 2 & 3 are only applicable if an update installation creates the pending.xml file
- Windows 8.1 (Windows Server 2012 R2) and above does not generate the pending.xml during the online servicing

Attack Scenario (Reinfection !)

1. Compromis the target system once with a 0-day (get root!)
2. Replace target components with old versions that contain remote code execution vulnerability
   > A hidden remote code execution vulnerability is generated !
3. The attacker’s initial exploit is removed (by user)
4. The attacker exploits the hidden remote code execution vulnerability
   > The attacker reinfects the target system !

Extending the Life Cycle of a Vulnerability



Stopping Criteria

1. The next cumulative update replaces the tampered components
2. The operating system of a target host is re-installed
3. The user detects and recovers the tampered components by himself/herself

Countermeasures

We need a blind spot monitor.. !

Package-Component Mappings

The information about *which packages contain which components*

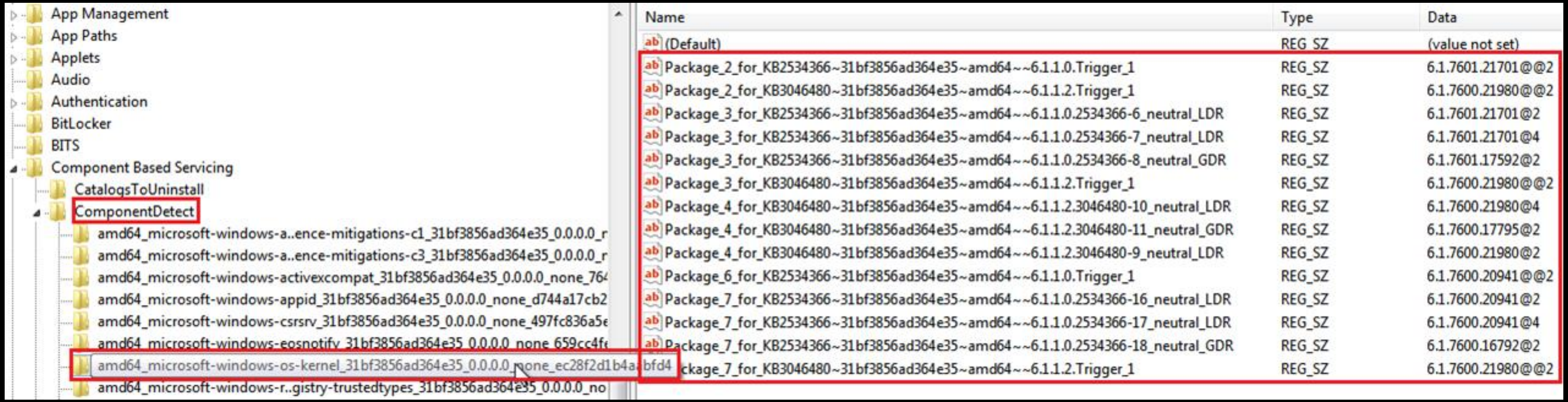
Required Information

- The list of package-component mappings
  (A record set of *[Package Name | Component Name | Component Version]*)

- Component Information on the system (names, versions)
- Package Information on the system (names, installation status)

Where can we Find the Package-Component Mappings?

The package-component mappings in registry



```
App Management
App Paths
Applets
Audio
Authentication
BitLocker
BITS
Component Based Servicing
    CatalogsToUninstall
    ComponentDetect
        amd64_microsoft-windows-a..ence-mitigations-c1_31bf3856ad364e35_0.0.0.0_r
        amd64_microsoft-windows-a..ence-mitigations-c3_31bf3856ad364e35_0.0.0.0_r
        amd64_microsoft-windows-activexcompat_31bf3856ad364e35_0.0.0.0_none_764
        amd64_microsoft-windows-appid_31bf3856ad364e35_0.0.0.0_none_d744a17cb2
        amd64_microsoft-windows-csrsrv_31bf3856ad364e35_0.0.0.0_none_497fc836a5e
        amd64_microsoft-windows-eosnotify_31bf3856ad364e35_0.0.0.0_none_659cc4fe
        amd64_microsoft-windows-os-kernel_31bf3856ad364e35_0.0.0.0_none_ec28f2d1b4abfd4
        amd64_microsoft-windows-r..gistry-trustedtypes_31bf3856ad364e35_0.0.0.0_no
```

Name	Type	Data
(Default)	REG_SZ	(value not set)
Package_2_for_KB2534366~31bf3856ad364e35~amd64~~6.1.1.0.Trigger_1	REG_SZ	6.1.7601.21701@@2
Package_2_for_KB3046480~31bf3856ad364e35~amd64~~6.1.1.2.Trigger_1	REG_SZ	6.1.7600.21980@@2
Package_3_for_KB2534366~31bf3856ad364e35~amd64~~6.1.1.0.2534366-6_neutral_LDR	REG_SZ	6.1.7601.21701@2
Package_3_for_KB2534366~31bf3856ad364e35~amd64~~6.1.1.0.2534366-7_neutral_LDR	REG_SZ	6.1.7601.21701@4
Package_3_for_KB2534366~31bf3856ad364e35~amd64~~6.1.1.0.2534366-8_neutral_GDR	REG_SZ	6.1.7601.17592@2
Package_3_for_KB3046480~31bf3856ad364e35~amd64~~6.1.1.2.Trigger_1	REG_SZ	6.1.7600.21980@@2
Package_4_for_KB3046480~31bf3856ad364e35~amd64~~6.1.1.2.3046480-10_neutral_LDR	REG_SZ	6.1.7600.21980@4
Package_4_for_KB3046480~31bf3856ad364e35~amd64~~6.1.1.2.3046480-11_neutral_GDR	REG_SZ	6.1.7600.17795@2
Package_4_for_KB3046480~31bf3856ad364e35~amd64~~6.1.1.2.3046480-9_neutral_LDR	REG_SZ	6.1.7600.21980@2
Package_6_for_KB2534366~31bf3856ad364e35~amd64~~6.1.1.0.Trigger_1	REG_SZ	6.1.7600.20941@@2
Package_7_for_KB2534366~31bf3856ad364e35~amd64~~6.1.1.0.2534366-16_neutral_LDR	REG_SZ	6.1.7600.20941@2
Package_7_for_KB2534366~31bf3856ad364e35~amd64~~6.1.1.0.2534366-17_neutral_LDR	REG_SZ	6.1.7600.20941@4
Package_7_for_KB2534366~31bf3856ad364e35~amd64~~6.1.1.0.2534366-18_neutral_GDR	REG_SZ	6.1.7600.16792@2
ckage_7_for_KB3046480~31bf3856ad364e35~amd64~~6.1.1.2.Trigger_1	REG_SZ	6.1.7600.21980@@2

Component Name　Package Name　Component Version

Detecting the Package-Component Mismatches

The detection procedure

1. List the installed packages on the system
2. Check the package-component mappings
3. Verify the hardlink information of components

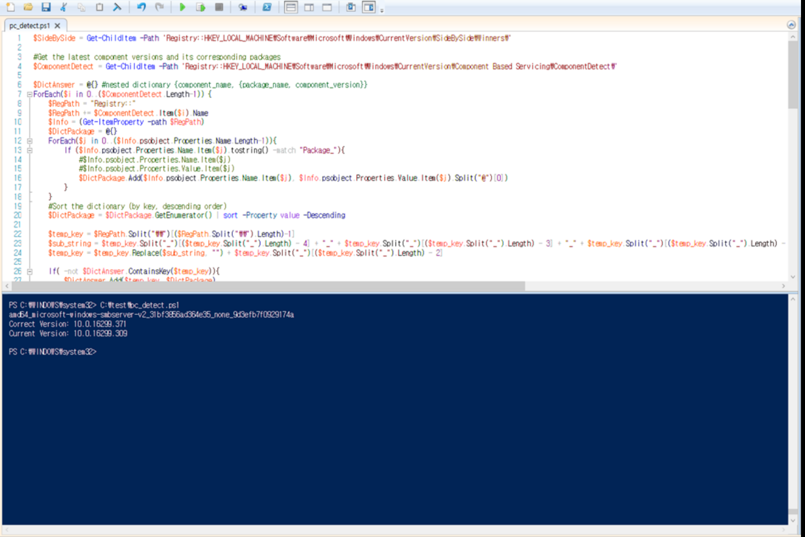
Fixing the Package-Component Mismatches

The correction procedure

1. Identify the replaced components (through the detection scheme)
2. Change the component version in the registry “Winners” (C_v)
3. Run the “SFC /SCANNOW” command

GutHub project
- Update State Checker

https://github.com/ksj1230/Update-State-Checker



```
$SideBySide = Get-ChildItem -Path 'Registry::HKEY_LOCAL_MACHINE₩Software₩Microsoft₩Windows₩CurrentVersion₩SideBySide₩Winners₩'

#Get the latest component versions and its corresponding packages
$ComponentDetect = Get-ChildItem -Path 'Registry::HKEY_LOCAL_MACHINE₩Software₩Microsoft₩Windows₩CurrentVersion₩Component Based Servicing₩ComponentDetect₩'

$DictAnswer = @{} #nested dictionary {component_name, {package_name, component_version}}
ForEach($i in 0..($ComponentDetect.Length-1)) {
    $RegPath = "Registry::"
    $RegPath += $ComponentDetect.Item($i).Name
    $Info = (Get-ItemProperty -path $RegPath)
    $DictPackage = @{}
    ForEach($j in 0..($Info.psobject.Properties.Name.Length-1)){
        If ($Info.psobject.Properties.Name.Item($j).tostring() -match "Package_"){
            #$Info.psobject.Properties.Name.Item($j)
            #$Info.psobject.Properties.Value.Item($j)
            $DictPackage.Add($Info.psobject.Properties.Name.Item($j), $Info.psobject.Properties.Value.Item($j).Split("@")[0])
        }
    }
    #Sort the dictionary (by key, descending order)
    $DictPackage = $DictPackage.GetEnumerator() | sort -Property value -Descending

    $temp_key = $RegPath.Split("₩₩")[($RegPath.Split("₩₩").Length)-1]
    $sub_string = $temp_key.Split("_")[($temp_key.Split("_").Length) - 4] + "_" + $temp_key.Split("_")[($temp_key.Split("_").Length) - 3] + "_" + $temp_key.Split("_")[($temp_key.Split("_").Length) -
    $temp_key = $temp_key.Replace($sub_string, "") + $temp_key.Split("_")[($temp_key.Split("_").Length) - 2]

    If( -not $DictAnswer.ContainsKey($temp_key)){
        $DictAnswer.Add($temp_key, $DictPackage)
```

```
PS C:₩WINDOWS₩system32> C:₩test₩pc_detect.ps1
amd64_microsoft-windows-smbserver-v2_31bf3856ad364e35_none_9d3efb7f0929174a
Correct Version: 10.0.16299.371
Current Version: 10.0.16299.309

PS C:₩WINDOWS₩system32>
```

Conclusion

Limitations (for Attackers)

Component dependency problem

- Security patches are implementation dependent
- Lack of Official Information

In future work, we will cover a VM-based automated testing system that can identify component dependencies.

Limitations (for Defenders)

Incomplete blind spot monitoring

- Does not consider rootkit attacks
- Deal only with type II blind spot.

Eliminating type I blind spots will be pursued in future work.

Summary

The blind spots on the Windows update management

- Two types of blind spots
- No means to detect or fix the package-component mismatches

Update State Tampering Attack

- Take advantage in target reinfection after system recovery.

Countermeasures

- Use the package-component mapping
- We provide *Update State Checker* in GitHub

Q & A

Contact: carp1230@gmail.com