The Dangers of Per-User COM Objects

Virus Bulletin 2011 - Barcelona
Jon Larimer
jlarimer@gmail.com
About Me

• Security researcher / software developer / reverse engineer

• 1998-2004: ISS X-Force

• 2004-2009: nCircle Network Security

• 2009-2011: IBM X-Force

• Currently at Google on the Android Security Team

• Live in Atlanta, GA, USA
Agenda

- Component Object Model (COM)
- Why per-user COM objects are dangerous
- Windows integrity levels, access tokens, User Account Control (UAC)
- Privilege elevation attacks with per-user COM objects
- Detecting and preventing attacks
- Q&A
COM: Component Object Model

- From the MSDN Library: COM is a platform-independent, distributed, object-oriented system for creating binary software components that can interact.

- Technology behind OLE (object linking and embedding), OLE Automation, DCOM (Distributed COM), COM+ (pre-cursor to .NET), and ActiveX

- COM components can be accessed by a variety of languages: C/C++, C# and other .NET languages, JScript/VBScript

- Much of the Windows Shell is built on COM and uses COM components
How COM objects are used

- COM objects are registered in system registry
- Objects are referenced through a Class ID (CLSID) and an Interface ID (IID)
- `CoCreateInstance()` uses the CLSID and IID to locate the executable code and load it into memory, then provides the client with a pointer to access class members
- In-process COM objects are DLLs that get loaded into the same process
Per-user COM objects

- Machine-wide COM objects are registered in 
  HKEY_LOCAL_MACHINE\Software\Classes

- Per-user COM objects are registered in 
  HKEY_CURRENT_USER\Software\Classes

- Per-user COM objects:
  - Can be registered by any process at Medium integrity level
  - Are only visible to the user that installs them
  - Take precedence over machine-wide objects in the COM subsystem
Abusing precedence

- Most Windows software uses many COM objects, intentional or not
- Windows looks for per-user COM objects before loading a machine-wide one
- Malicious software could install a per-user COM object with the same CLSID as a machine-wide object
- **COM object hijacking**: Malware can replace a benign system-wide COM object with a malicious per-user object that gets loaded in its place
More on precedence attacks

- **Malware persistence** - Explorer loads COM objects when a user logs on

- **Process injection** - Some programs can be convinced to load a COM object after they're already running (Explorer, web browsers)

- **User mode rootkits** - Inject into a process, hook API calls

- Can be hard to detect - no extra running process

- What about privilege elevation attacks?
Access Tokens

- Contain information on privileges, group membership, and integrity level
- Used to control access to files, registry keys, named pipes, and other objects
- Every process has an access token
- Administrative users get two access tokens - one for normal use, one for elevated privileges
Windows Integrity Levels

- Introduced in Vista
- Designed to restrict the access of less-than-trustworthy applications
- Blocks processes with a lower integrity level from accessing objects with a higher integrity level
- Example: Notepad can't save a file in C:\Windows\System32.
- Five integrity levels: Untrusted, Low, Medium, High, System
  - Sandboxed apps: Low
  - Regular user processes: Medium
  - Elevated (UAC) processes: High
User Account Control (UAC)

- Introduced in Vista
- Used to elevate the integrity level of a process
- Can provide the Administrator token to members of the Administrators group
- Four levels available in Windows 7:
  - Always notify
  - Notify only when programs try to change computer (default)
  - Notify only when programs try to change computer, don't dim desktop
  - Never notify
- Privilege elevation also available through “Run as Administrator...” option
Back to privilege elevation...

- **MSDN says:** *Beginning with Windows Vista® and Windows Server® 2008, if the integrity level of a process is higher than Medium, the COM runtime ignores per-user COM configuration and accesses only per-machine COM configuration.* This action reduces the surface area for elevation of privilege attacks, preventing a process with standard user privileges from configuring a COM object with arbitrary code and having this code called from an elevated process.

- This is mostly true

- What about custom COM object loaders?
Per-user COM object privilege elevation in shell32.dll

- There is a vulnerability in shell32.dll's \texttt{SHCoCreateInstance()} call

- If a high integrity level process uses this API call, it can be tricked into loading a per-user COM object
  - \text{Medium} \rightarrow \text{High integrity level privilege elevation}

- Reported to MSRC in March 2011. They acknowledged the vulnerability but declined to fix it
  - Why? It's an elevation of privilege attack that requires administrator rights

- The bug is still useful in a couple of different attack scenarios..
UAC hijack attack

• Find an app that requires UAC and makes use of `SHCoCreateInstance()`
  • Many software installers use this...
• Register a per-user COM object for the requested CLSID
• Wait for the user to launch the app and approve the UAC dialog
• Your per-user COM object, registered with a Medium integrity process, now runs at High integrity
UAC hijack demo
UAC bypass attack

- There's a “UAC Whitelist” with 80+ applications
  - This was new in Windows 7
- These programs, digitally signed by Microsoft, are allowed to elevate to High integrity without UAC by default
- Some of these applications might call `SHCoCreateInstance()` while running at High integrity...
- Also see the research by Leo Davidson - he found a UAC bypass attack in the Windows 7 beta that's still not fixed
UAC bypass demo
Protecting yourself

• Don't use your PC as Administrator
  • Easier said than done
• Crank up UAC settings to “Always Notify”
• Treat per-user COM objects the same as anything in the registry's Run keys - be suspicious of anything there
• Be wary of processes with mapped DLLs that are in your home directory
Protecting your customers

- Scan registry for suspicious entries - a per-user COM object registered for a CLSID that also has a machine-wide entry
- Intercept registry access in high integrity processes, don't allow them access to per-user COM objects
- Run as much code as possible with Low integrity level, which doesn't have access to register per-user COM objects
- Audit your code to make sure it can't be targeted by malware for privilege elevations
  - Run Sysinternals Process Monitor to watch for High integrity level processes accessing per-user COM registrations
Protecting your employees

- Don't give users local Administrator access
- Implement application whitelisting
  - Ensure the solution allows whitelisting of DLLs
  - This can be painful to implement
- Periodically scan systems for suspicious per-user COM objects
Questions?

• Contact info:
  • Email: jlarimer@gmail.com
  • Twitter: @shydemeanor
  • LinkedIn, Google+, etc