README BSD-2-Clause license

build passing

UACMe

- Defeating Windows User Account Control by abusing built-in Windows AutoElevate backdoor.

System Requirements

- x86-32/x64 Windows 7/8/8.1/10/11 (client, some methods however works on server version too).
- Admin account with UAC set on default settings required.

Usage

Run executable from command line: akagi32 [Key] [Param] or akagi64 [Key] [Param]. See "Run examples" below for more info.

First parameter is number of method to use, second is optional command (executable file name including full path) to run. Second parameter can be empty - in this case program will execute elevated cmd.exe from system32 folder.

Note: Since 3.5.0 version all "fixed" methods are considered obsolete and removed altogether with all supporting code/units. If you still need them - use v3.2.x branch

▶ Keys (click to expand/collapse)

Note:

- Method (30) (63) and later implemented only in x64 version;
- Method (30) require x64 because it abuses WOW64 subsystem feature;
- Method (55) is not really reliable (as any GUI hacks) and included just for fun;
- Method (78) requires current user account password not to be blank.

Run examples:

- akagi32.exe 23
- akagi64.exe 61
- akagi32 23 c:\windows\system32\calc.exe
- akagi64 61 c:\windows\system32\charmap.exe

hfiref0x

IvanovCosmin IvanovCosmin

Languages

C 94.0% C++ 6.0%

Warning

- This tool shows ONLY popular UAC bypass method used by malware, and re-implement some of them in a different way improving original concepts. There are different, not yet known to the general public, methods. Be aware of this;
- This tool is not intended for AV tests and not tested to work in aggressive AV environment, if you still plan to use it with installed bloatware AV soft - use it at your own risk;
- Some AV may flag this tool as HackTool, MSE/WinDefender constantly marks it as malware, nope;
- If you run this program on real computer remember to remove all program leftovers after usage, for more info about files it drops to system folders see source code;
- Most of methods created for x64, with no x86-32 support in mind. I don't see any sense in supporting 32 bit versions of Windows or wow64, however with small tweaks most of them will run under wow64 as well.

If you wondering why this still exists and working - here is the explanation - an official Microsoft WHITEFLAG (including totally incompetent statements as bonus) https://devblogs.microsoft.com/oldnewthing/20160816-00/?p=94105

Windows 10 support and testing policy

- UACMe tested only with LSTB/LTSC variants (1607/1809) and Last RTM-1 versions, e.g. if current version is 2004 it will be tested on 2004 (19041) and previous version 1909 (18363);
- Insider builds are not supported as methods may be fixed there.

Protection

- Account without administrative privileges.

Malware usage

- We do not take any responsibility for this tool usage in the malicious purposes. It is free, open-source and provided AS-IS for everyone.

Other usage

- Currently used as "signature" by "THOR APT" scanner (handmade pattern matching fraudware from Germany). We do not take any responsibility for this tool usage in the fraudware;
- The repository https://github.com/hfiref0x/UACME and it contents are the only genuine source for UACMe code. We have nothing to do with external links to this project, mentions anywhere as well as modifications (forks);
- In July 2016 so-called "security company" Cymmetria released report about script-kiddie malware bundle called "Patchwork" and false flagged it as APT. They stated it was using "UACME method", which in fact is just slightly and unprofessionally modified injector dll from UACMe v1.9 and was using Carberp/Pitou hybrid method in malware self-implemented way. We do not take any responsibility for UACMe usage in the dubious advertising campaigns from third party "security companies".

Build

- UACMe comes with full source code, written in C;

- In order to build from source you need Microsoft Visual Studio 2019 and later versions.

Compiled Binaries

- They are not provided since 2.8.9 and will never be provided in future. The reasons (and why you too should not provide them to the general public):
  - If you look at this project in a nutshell it is a HackTool, despite initial goal to be a demonstrator. Of course several AV's detects it as HackTool (MS WD for example), however most of VirusTotal patients detects it as generic "malware". Which is of course incorrect, however unfortunately some lazy malware writers blindly copy-paste code to their crapware (or even simple use this tool directly) thus some AV created signatures based on project code parts;
  - By giving compiled binaries to everyone you make life of script-kiddies much easier because having need to compile from source works as perfect barrier for exceptionally dumb script-kiddies and "button-clickers";
  - Having compiled binaries in the repository will ultimately lead to flagging this repository pages as malicious (due to above reasons) by various content filters (SmartScreen, Google Safe Browsing etc).
- This decision is a final and won't be changed.

Instructions

- Select Platform ToolSet first for project in solution you want to build (Project->Properties->General):
  - v142 for Visual Studio 2019;
  - v143 for Visual Studio 2022.

- For v140 and above set Target Platform Version (Project->Properties->General):
  - If v140 then select 8.1 (Note that Windows 8.1 SDK must be installed);
  - If v141 and above then select 10.
- The following SDK are required for building the binaries:
  - Windows 8.1 or Windows 10 SDK (tested with 19041 version)
  - NET Framework SDK (tested with 4.8 version)
- To build working binary:
  - Compile payload units
  - Compile Naka module
  - Encrypt all payload units using Naka module
  - Generate secret blobs for these units using Naka module
  - Move compiled units and secret blobs to the Akagi\Bin directory
  - Rebuild Akagi

References

- Windows 7 UAC whitelist, http://www.pretentiousname.com/misc/win7_uac_whitelist2.html
- Malicious Application Compatibility Shims, https://www.blackhat.com/docs/eu-15/materials/eu-15-Pierce-Defending-Against-Malicious-Application-Compatibility-Shims-wp.pdf
- Junfeng Zhang from WinSxS dev team blog, https://blogs.msdn.microsoft.com/junfeng/
- Beyond good ol' Run key, series of articles, http://www.hexacorn.com/blog

- KernelMode.Info UACMe thread, https://www.kernelmode.info/forum/viewtopicf985.html?f=11&t=3643
- Command Injection/Elevation - Environment Variables Revisited, https://breakingmalware.com/vulnerabilities/command-injection-and-elevation-environment-variables-revisited
- "Fileless" UAC Bypass Using eventvwr.exe and Registry Hijacking, https://enigma0x3.net/2016/08/15/fileless-uac-bypass-using-eventvwr-exe-and-registry-hijacking/
- Bypassing UAC on Windows 10 using Disk Cleanup, https://enigma0x3.net/2016/07/22/bypassing-uac-on-windows-10-using-disk-cleanup/
- Using IARPUninstallStringLauncher COM interface to bypass UAC, http://www.freebuf.com/articles/system/116611.html
- Bypassing UAC using App Paths, https://enigma0x3.net/2017/03/14/bypassing-uac-using-app-paths/
- "Fileless" UAC Bypass using sdclt.exe, https://enigma0x3.net/2017/03/17/fileless-uac-bypass-using-sdclt-exe/
- UAC Bypass or story about three escalations, https://habrahabr.ru/company/pm/blog/328008/
- Exploiting Environment Variables in Scheduled Tasks for UAC Bypass, https://tyranidslair.blogspot.ru/2017/05/exploiting-environment-variables-in.html
- First entry: Welcome and fileless UAC bypass, https://winscripting.blog/2017/05/12/first-entry-welcome-and-uac-bypass/
- Reading Your Way Around UAC in 3 parts:
    i. https://tyranidslair.blogspot.ru/2017/05/reading-your-way-around-uac-part-1.html
    ii. https://tyranidslair.blogspot.ru/2017/05/reading-your-way-around-uac-part-2.html

iii. https://tyranidslair.blogspot.ru/2017/05/reading-your-way-around-uac-part-3.html

- Research on CMSTP.exe, https://msitpros.com/?p=3960
- UAC bypass via elevated .NET applications, https://offsec.provadys.com/UAC-bypass-dotnet.html
- UAC Bypass by Mocking Trusted Directories, https://medium.com/tenable-techblog/uac-bypass-by-mocking-trusted-directories-24a96675f6e
- Yet another sdclt UAC bypass, http://blog.sevagas.com/?Yet-another-sdclt-UAC-bypass
- UAC Bypass via SystemPropertiesAdvanced.exe and DLL Hijacking, https://egre55.github.io/system-properties-uac-bypass/
- Accessing Access Tokens for UIAccess, https://tyranidslair.blogspot.com/2019/02/accessing-access-tokens-for-uiaccess.html
- Fileless UAC Bypass in Windows Store Binary, https://www.activecyber.us/1/post/2019/03/windows-uac-bypass.html
- Calling Local Windows RPC Servers from .NET, https://googleprojectzero.blogspot.com/2019/12/calling-local-windows-rpc-servers-from.html
- Microsoft Windows 10 UAC bypass local privilege escalation exploit, https://packetstormsecurity.com/files/155927/Microsoft-Windows-10-Local-Privilege-Escalation.html
- UACMe 3.5, WD and the ways of mitigation, https://swapcontext.blogspot.com/2020/10/uacme-35-wd-and-ways-of-mitigation.html
- UAC bypasses from COMAutoApprovalList, https://swapcontext.blogspot.com/2020/11/uac-bypasses-from-comautoapprovallist.html
- Utilizing Programmatic Identifiers (ProgIDs) for UAC Bypasses, https://v3ded.github.io/redteam/utilizing-programmatic-identifiers-progids-for-uac-bypasses

- MSDT DLL Hijack UAC bypass, https://blog.sevagas.com/?MSDT-DLL-Hijack-UAC-bypass
- UAC bypass through .Net Deserialization vulnerability in eventvwr.exe, https://twitter.com/orange_8361/status/1518970259868626944
- Advanced Windows Task Scheduler Playbook - Part.2 from COM to UAC bypass and get SYSTEM directly, http://www.zcgonvh.com/post/Advanced_Windows_Task_Scheduler_Playbook-Part.2_from_COM_to_UAC_bypass_and_get_SYSTEM_dirtectly.html
- Bypassing UAC with SSPI Datagram Contexts, https://splintercod3.blogspot.com/p/bypassing-uac-with-sspi-datagram.html
- Mitigate some Exploits for Windows' ® UAC, https://skanthak.hier-im-netz.de/uacamole.html

Authors