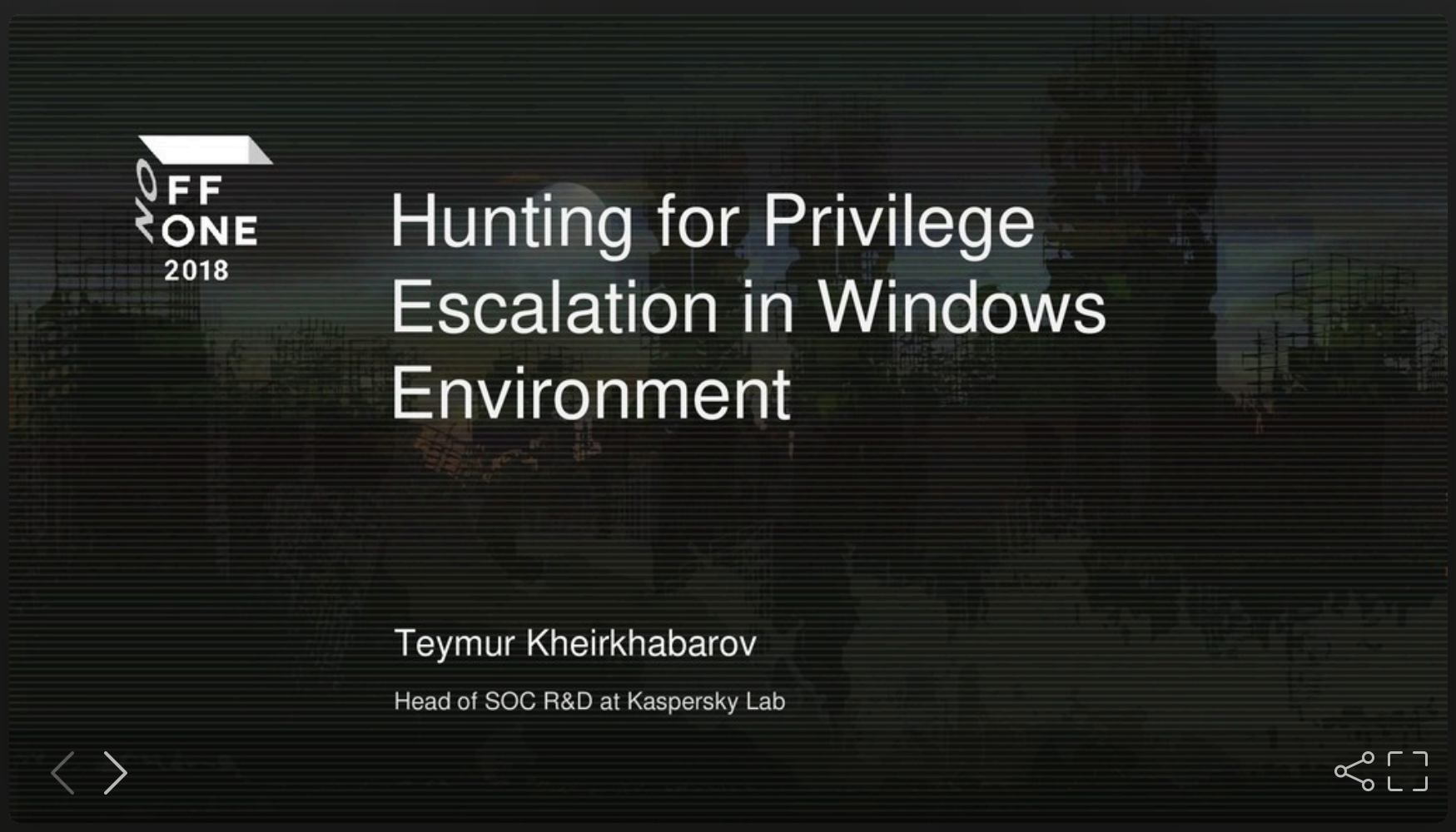


Heirhabarov November 16, 2018 Technology 13 28k

Hunting for Privilege Escalation in Windows Environment

Slides from my talk at the OFFZONE 2018 conference (https://www.offzone.moscow/report/hunting-for-privilege-escalation-in-windows-environment/)

Heirhabarov
November 16, 2018

Tweet Share

More Decks by Heirhabarov

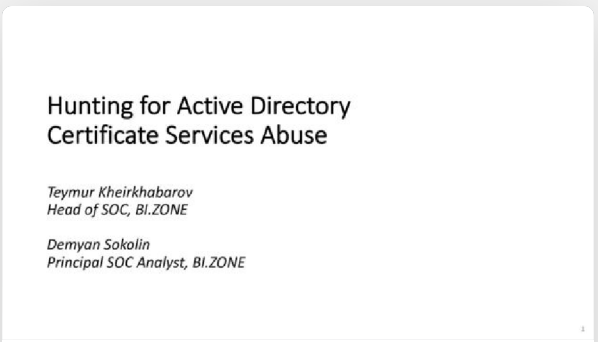
See All by Heirhabarov

Hunting For The Most Unusual Atta…

heirhabarov 1 170

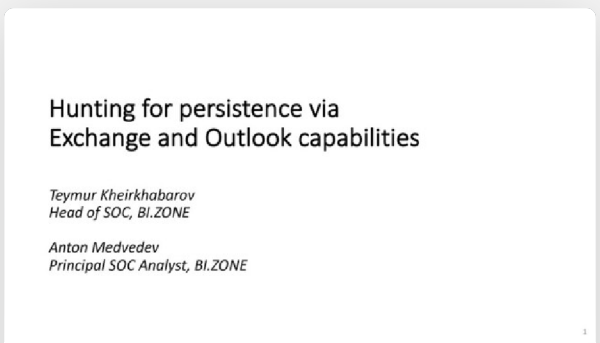
Hunting for macOS attack techniqu…

heirhabarov 2 2.7k



Hunting for Active Directory Certi…

heirhabarov 2 6.5k



Hunting for persistence via Micros…

heirhabarov 1 7.5k

Hunting for PowerShell Abuse

heirhabarov 9 18k

PHDays 2018 Threat Hunting Hand…

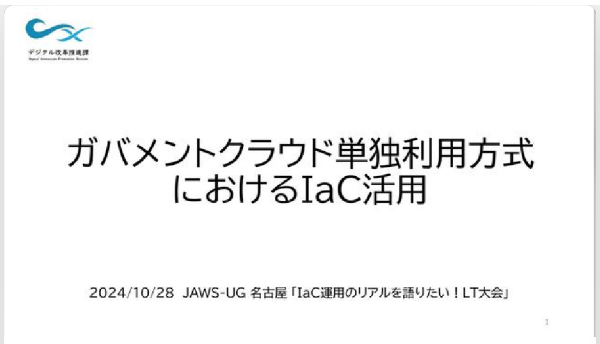
heirhabarov 7 5.2k

Hunting for Credentials Dumping in…

heirhabarov 5 6k

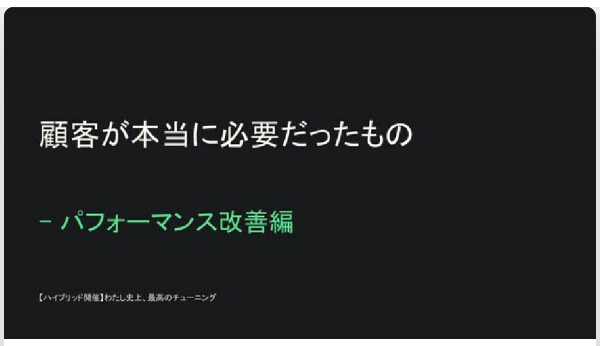
Other Decks in Technology

See All in Technology



ガバメントクラウド単独利用方式...

techniczna ☆ 3 👁 240



顧客が本当に必要だったもの - パ...

soudai ☆ 23 👁 6.4k

なんで、私がAWS Heroに!? ～社外...

minorun365 ... ☆ 6 👁 980

ネット広告に未来はあるか？「3rd...

cyberagentde... ☆ 1 👁 120

AWS re:Inventを徹底的に楽しむた...

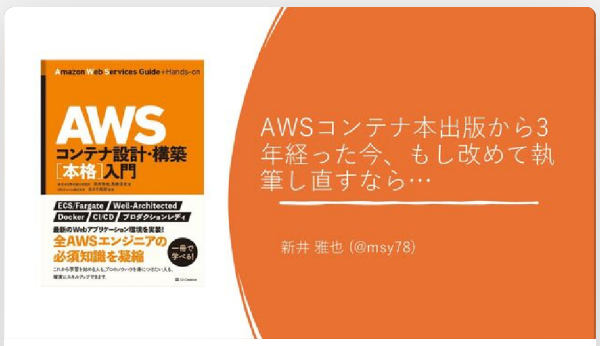
yuj1osm ☆ 1 👁 460

生成AIの強みと弱みを理解して、...

cyberagentde... ☆ 1 👁 160

Product Engineer Night #6プロダク...

hacomono ... ☆ 1 👁 210



AWSコンテナ本出版から3年経っ...

iselegant ☆ 13 👁 3.6k

いまならこう作りたい AWSコンテ...

horsewin ☆ 7 👁 1.8k

生成AIとAWS CDKで実現！ 自社プ...

ymae ☆ 2 👁 280

APIテスト自動化の勘所

yokawasa ☆ 6 👁 3.6k

ガチ勢によるPipeCD運用大全～滑...

cyberagentde... ☆ 3 👁 180

Featured

See All Featured >

Transcript

1. **Hunting for Privilege Escalation in Windows Environment Teymur Kheirkhabarov Head**

   of SOC R&D at Kaspersky Lab

2. **• Head of SOC R&D at Kaspersky Lab • Threat**

   Hunter • Big fan of ELK stack • Zero Nights / PHDays speaker • Ex- System Admin • Ex- Infosec Admin • Ex- Infosec dept. Head • Twitter @HeirhabarovT

3. **What are we going to talks about? Privilege escalation is**

   the result of actions that allows an adversary to obtain a higher level of permissions on a system or network. We will look at different methods of local privilege escalation in Windows environment and how to detect them via logs.

4. **Theory**

5. **Theory. Access token An access token is an object that**

   describes the security context of a process or thread. It is created during logon and never changes* after creation. Token contains: • User SID • Group SIDs / Restricted group SIDs • Integrity level (Mandatory label) • Logon Session SID • Token type (primary or impersonation) • Impersonation level • User privileges list • Other

6. **Theory. Mandatory integrity control IL Usage IL SID Untrusted Anonymous**

   S-1-16-0 Low Everyone. Used by Protected Mode of Internet Explorer S-1-16-4096 Medium Used by normal applications being launched while UAC is enabled S-1-16-8192 High Privileged users (if UAC enabled) or all authenticated users (if UAC disabled) S-1-16-12288 System LocalSystem. NetworkService, LocalService S-1-16-16384 Default mandatory policy for all objects: No-Write-Up Default mandatory policy for processes: No-Write-Up + No-Read-Up Default implicit integrity level for files – Medium

7. **Theory. Authorization and privilege escalation Mandatory Integrity Control Discretionary Access**

   Control Is access granted? Subject Object Security Descriptor Owner SID Group SID DACL SACL (obj IL is here) Access Token User SID Groups SIDs Privileges Integrity Level Access Denied Access Denied YES YES NO NO Is access granted? Influence on Bypass, using special privileges (Debug, Restore, Backup, Take Ownership)

8. **Stored Credentials**

9. **Stored Credentials. Files In case of OS unattended installation, answer**

   files may be left in the system. These answer files can contain credentials of the privileged local accounts (e.g. Administrator): • C:\sysprep\sysprep.xml • C:\sysprep\sysprep.inf • C:\sysprep.inf • C:\unattend.xml • C:\Windows\Panther\Unattend.xml • C:\Windows\Panther\Unattend\Unat tend.xml sysprep.xml Base64 “encryption” Unattended.xml Base64 “encryption” sysprep.inf

10. **Stored Credentials. Files. Group Policy Preferences Group policy preferences allows**

    domain admins to create and deploy across the domain local users and local administrators accounts. In case of usage this function policy preference files are created. These files are located in the SYSVOL shared directory and any authenticated user in the domain has read access to these files since it is needed in order to obtain group policy updates. Policy preference files contain encrypted passwords… But encryption key is hardcoded and published by Microsoft Policy preference files are located: • C:\ProgramData\Microsoft\Group Policy\History\????\Machine\Preferences\Groups\Groups.xml • \\????\SYSVOL\\Policies\????\Machine\Preferences\Groups\Groups.xml Also several other policy preference files can be useful: • Services\Services.xml • ScheduledTasks\ScheduledTasks.xml • Printers\Printers.xml • Drives\Drives.xml • DataSources\DataSources.xml

11. **Stored Credentials. Files. Group Policy Preferences AES decryption routine “test”**

12. **Stored Credentials. Files. Let’s hunt it! Deception-like approach – usage**

    of fake files with fake credentials. Monitor accesses to these files. Fake Unattend.xml Fake policy preference file

13. **event_id:4663 AND event_data.AccessList:"*%%4416*" AND event_data.ObjectName:("\\{641ECF7F-6AC4-4A63- BF85-DFDE140E9F89}\\Machine\\Preferences\\Groups\\Groups.xml" "\\Panther\\Unattend.xml") Stored Credentials. Files.**

    Let’s hunt it! Search for accessing of fake files with stored credentials: Fake files with credentials

14. **Stored Credentials. Registry Adversaries may query the Registry looking for**

    credentials and passwords that have been stored for use by other programs or services. For example, these credentials can be used for automatic logon. The idea behind the Windows Auto Login is that a user, specified in DefaultUserName can logon at a computer without having to type their password.

15. **Stored Credentials. Files/registry. Let’s hunt it! Deception-like approach – usage**

    of fake files with fake credentials. Monitor unsuccessful authentication attempts with fake credentials. Fake groups.xml with fake admin account Unsuccessful authentication attempt with fake account Unsuccessful authentication attempt with fake account

16. **(event_id:(4625 OR 4648) OR (event_id:4776 AND -event_data.Status:0x0)) AND event_data.TargetUserName:FakeAccountUserName Stored**

    Credentials. Files/registry. Let’s hunt it! Source computer. Outbound login attempt Fake account Destination host Fake account Destination computer. Inbound unsuccessful login attempt Source host Search for unsuccessful authentication attempts with fake account (that is specified in fake file with stored credentials):

17. **Tricking some privileged processes into executing arbitrary code**

18. **Windows stores local service configuration information in the Registry under**

    HKLM\SYSTEM\CurrentControlSet\Services. Adversaries can change the service ImagePath, FailureCommand or ServiceDll to point to a different executable under their control, if the permissions for users and groups are not properly set and allow access to the Registry keys for a service. Service registry permissions weakness 1. Find writeable registry keys for services, using Accesschk 2. Change ImagePath 3. Restart Service

19. **Events, related to changing Services registry keys by non-privileged users**

    Service registry permissions weakness. Let’s hunt it! Medium IL shows us that user is non-privileged

20. **Search for usage of reg or Powershell by non-privileged users**

    to modify service configuration in registry: event_id:1 AND event_data.IntegrityLevel:Medium AND ((event_data.CommandLine:*reg* AND event_data.CommandLine:*add*) OR (event_data.CommandLine:*powershell* AND event_data.CommandLine:("*set- itemproperty*" "* sp *" "*new-itemproperty*"))) AND event_data.CommandLine:(*ControlSet* AND *Services*) AND event_data.CommandLine:(*ImagePath* *FailureCommand* *ServiceDll*) Service registry permissions weakness. Let's hunt it! Medium IL shows us that user is non-privileged

21. **source_name:"Microsoft-Windows-Sysmon" AND event_id:13 AND event_data.IntegrityLevel:Medium AND event_data.TargetObject:(*ControlSet* AND *services*) AND**

    event_data.TargetObject:("\\ImagePath" "\\FailureCommand" "\\ServiceDll") Service registry permissions weakness. Let's hunt it! Search for changing Services registry keys by non-privileged users: Save to memcached Get from memcached

22. **Using Logstash memcached filter we can cache some information about**

    started processes for further enrichment of other events: • Integrity Level • User • Command line • Parent Image Building information block for caching: Saving previously built information block in cache (key is concatenation of ProcessGuid and computer_name): Cache information about started processes

23. **Add additional information from cache, that is available only in**

    Process Creation event (User, IL…) Enrich Sysmon events with additional information about process Get information from cache Enrich event

24. **Enrich Sysmon events with additional information about process Result of**

    enrichment with information about process

25. **Enrich Sysmon process creation events with information about parent process**

    Get previously cached information about parent process from cache to enrich process creation events. Get information from cache Enrich event

26. **Enrich Sysmon process creation events with information about parent process**

    Result of enrichment with information about parent process

27. **Service permissions weakness Service is an operating system object. As**

    any object it has DACL. Sometimes it is possible to discover services that run with SYSTEM privileges and don't have appropriate permissions. Adversaries can use it to elevate privileges by changing the service ImagePath, FailureCommand or ServiceDll to point to a different executable under their control. It can be done via SCM API or using sc.exe utility. 1. Discover service with weak permissions, using Accesschk 2. Change service binPath 3. Restart service

28. **Service permissions weakness. Let's hunt it! Events, related to usage**

    of sc utility by non-privileged users to change service configuration. Medium IL shows us that user is non-privileged Usage of sc to change service binPath Usage of sc to restart service

29. **Search for usage of sc by non-privileged user to change**

    service binPath or Failure command: source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.IntegrityLevel:Medium AND event_data.Image:"\\sc.exe" AND ((event_data.CommandLine:*config* AND event_data.CommandLine:*binPath*) OR (event_data.CommandLine:*failure* AND event_data.CommandLine:*command*)) Service permissions weakness. Let's hunt it! Medium IL shows us that user is non-privileged

30. **When a service in Windows is started, OS try to**

    find the location of its executable. In case when executable path is enclosed in quotes Windows has no question about where to find it. But if the executable path contains spaces and isn't surrounded by quotes OS will try to find file and execute it inside every folder of the path until reach the executable. In this case, the part between the backslash and the space will be treated as the file name, and the remaining part - as command line arguments. Unquoted service path Path with spaces, isn't surrounded by quotes Finding of service executable

31. **Unquoted service path. Exploitation 1. Find vulnerable service 2. Check**

    rights for the folders in path 3. Drop executable with the name as part of the folder name prior to space and restart service

32. **Execution after attack Unquoted service path. Let's hunt it!**

33. **Execution after attack Unquoted service path. Let's hunt it! Path**

    to the dropped executable Command line arguments

34. **Search for process creation events, where parent is “services.exe”, the**

    beginning of command line in the quotes doesn’t end with extension and the same as image path without extension. Also there should be cutted part of a file path in the right side of the command line (after the part in quotes): {"bool":{"must":[{"query_string": {"query":" event_id:1 AND event_data.ParentImage:\"*\\\\services.exe\" AND event_data.Image.keyword:*exe AND event_data.CommandLine.keyword:/*\\\\[\\\\a-zA-Z]+\\\"(|).+/ AND -event_data.CommandLine:(*svchost* *msiexec* *schtasks* *rundll32*) "}},{"script":{"script":" if (!doc[\"event_data.CommandLine.keyword\"].empty && !doc[\"event_data.Image.keyword\"].empty) { String file_path_stripped = doc[\"event_data.Image.keyword\"].value.toLowerCase().replace(\".exe\",\"\"); String[] filecmdline_parts = /\"\\s/.split(doc[\"event_data.CommandLine.keyword\"].value.toLowerCase()); if (filecmdline_parts.length >= 2 && filecmdline_parts[1].contains(\"\\\\\")) { if (filecmdline_parts[0].substring(1) == file_path_stripped) { return true; } } return false; } "}}]}} Unquoted service path. Let’s hunt it!

35. **If the user has permissions to write a file into**

    the folder of where the binary of the service is located then it is possible to just replace the binary with the a custom payload and then restart the service in order to escalate privileges. Modifiable service binary 1. Find service with writable binary 2. Replace binary and restart service

36. **Non-privileged process (1) drop executable (2), that is then executed**

    as a service with the System rights (3) Modifiable service binary. Let’s hunt it! 1 Replacing service binary using xcopy Medium IL shows us that process isn’t privileged 2 3 Modified by non-privileged user file is launched as service under SYSTEM

37. **Search for dropping of files to Windows/”Program Files” folders by**

    non-privileged processes: source_name:"Microsoft-Windows-Sysmon" AND event_id:11 AND event_data.IntegrityLevel:Medium AND (event_data.TargetFilename:("\\Program Files\\" "\\Program Files (x86)\\") OR event_data.TargetFilename.keyword:/\:\\[W|w][I|i][N|n][D|d][O|o][W|w][S|s]\\.*/) AND - event_data.TargetFilename:(*temp*) Modifiable service binary. Let’s hunt it! Get from memcached

38. **Search for execution as service with System rights of file,**

    that was dropped by non-privileged user: source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.ParentImage:"\\services.exe" AND event_data.User:"NT AUTHORITY\\SYSTEM" AND event_data.ImageModifierIntegrityLevel:Medium Modifiable service binary. Let’s hunt it! Key for caching information about dropped file Key for getting information about dropped file from cache

39. **Caching information about modified/created executables 1. Calculating file path fingerprint**

    2. Caching information about file modifier

40. **Enrich events with information about last modifier 1. Calculating file**

    path fingerprint 2. Obtaining information from cache 3. Enrich event with information about last modifier

41. **Enrich events with information about last modifier Using Logstash memcached**

    filter it is possible to cache information about created files for further enrichments of other events: Example of enrichment with information about last file modifier

42. **Privilege escalation via weak permissions. Accesschk tool usage. Let’s hunt**

    it! Events, related to usage of AccessChk utility to check rights on different objects. Finding writeable registry keys Metadata shows us, that it is renamed AccessChk Finding services, which we can control Metadata shows us, that it is renamed AccessChk

43. **source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.IntegrityLevel:Medium AND (event_data.Product:*accesschk* OR event_data.Description:(*Reports effective**

    permissions*)) Privilege escalation via weak permissions. Accesschk tool usage. Let’s hunt it!

44. **Windows environments provide a group policy setting which allows a**

    regular user to install a Microsoft Windows Installer Package (MSI) with system privileges. This can be discovered in environments where a standard user wants to install an application which requires system privileges and the administrator would like to avoid to give temporary local administrator access to a user. Always Install Elevated

45. **Always Install Elevated. Exploitation 1. Get current status of Always**

    Install Elevated Policy 2. MSI launching 3. Shell with the System privileges

46. **Always Install Elevated policy is disabled – in this case**

    if non privileged user runs MSI (1), Windows Installer service will try to install it with privileges of the current user (2) Always Install Elevated. Let’s hunt it! 1 2

47. **Always Install Elevated. Let’s hunt it! Always Install Elevated policy**

    is enabled – in this case if non privileged user runs MSI (1), Windows Installer service will try to install it with SYSTEM privileges (2) 1 2

48. **Search for chain of events: request to start MSI from**

    non privileged user (1) –> Windows Installer service try to install MSI packages with SYSTEM privileges (2): source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND ((event_data.Image:("\\Windows\\Installer\\" AND *msi* AND *tmp) AND event_data.User:"NT AUTHORITY\\SYSTEM") OR (event_data.Image:"\\msiexec.exe" AND -event_data.User:"NT AUTHORITY\\SYSTEM" AND -event_data.IntegrityLevel:System)) Always Install Elevated. Let's hunt it! 1 2

49. **Events, related to the spawning of cmd/Powershell from MSI package.**

    It is anomaly activity Always Install Elevated. Let's hunt it!

50. **source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.Image:("\\cmd.exe" "\\powershell.exe") AND event_data.ParentImage:("\\Windows\\Installer\\" AND *msi***

    AND *tmp) source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.ParentImage:("\\cmd.exe" "\\powershell.exe") AND event_data.ParentOfParent:("\\Windows\\Installer\\" AND *msi* AND *tmp) Search for spawning of cmd or Powershell by MSI package: Always Install Elevated. Let's hunt it! Search for spawning of processes from cmd/Powershell, spawned from MSI package:

51. **Kernel and driver vulnerabilities**

52. **Windows Kernel and 3rd-party drivers exploits Windows Kernel and 3rd-party**

    drivers vulnerabilities can allow an attacker to execute arbitrary code in the kernel mode. The goal of kernel or driver exploitation is often to somehow gain higher privileges (in the most cases SYSTEM). Possible kernel shellcodes, that can be used for LPE: • Token stealing (replacing token of some process with SYSTEM token); • Nulling out ACLs (null DACL means that everybody can access an object); • Changing objects' ACLs (gives full access to arbitrary object, e.g. to the process with SYSTEM privileges, disable auditing); • Changing tokens (new groups, new "super" privileges, increasing integrity level, changing user SID);

53. **Windows Kernel and 3rd-party drivers exploits. Token stealing How it**

    works: • Enumerate EPROCESS structures in kernel memory; • Find the EPROCESS address of the privileged (SYSTEM) process; • Find the EPROCESS address of the current process; • Read ACCESS TOKEN from the privileged process; • Replace ACCESS TOKEN of the current process with ACCESS TOKEN of the privileged process. winlogon.exe Process System cmd.exe Process User System

54. **Windows Kernel exploits 1. Discovery of missing patches 2. Vulnerability**

    exploitation

55. **Capcom driver vulnerability expliotation example (this driver was distributed with**

    Capcom's Street Fighter V computer game) 3rd-party drivers exploits 1. Find vulnerable driver 2. Vulnerability exploitation

56. **Token before exploitation Token after exploitation Windows Kernel and 3rd-party**

    drivers exploits. Token stealing

57. **Process was started with non-SYSTEM token and Medium IL but**

    spawns the child process with SYSTEM rights! Windows Kernel and 3rd-party drivers exploits. Token stealing Let's hunt it!

58. **Search for spawning of child processes with SYSTEM privileges by**

    parents with non- SYSTEM privileges and Medium integrity level: source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.ParentIntegrityLevel:Medium AND (event_data.IntegrityLevel:System AND event_data.User:"NT AUTHORITY\\SYSTEM") Windows Kernel and 3rd-party drivers exploits. Token stealing Let's hunt it! Save to memcached Get from memcached

59. **Token swapping, using Mimikatz driver 1. Installing mimidrv.sys driver 2.**

    Performing token swapping to SYSTEM via installed driver 3. Spawning cmd under SYSTEM acciunt 4. Checking current rights Token before swapping Token after swapping Token of spawned cmd

60. **Spawning child process under SYSTEM by process with High integrity**

    level Token swapping, using Mimikatz driver. Let's hunt it! Parent process started under account with high IL Child process started under SYSTEM account

61. **Search for spawning child process under SYSTEM by process with**

    High integrity level source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.ParentIntegrityLevel:High AND (event_data.IntegrityLevel:System AND event_data.User:"NT AUTHORITY\\SYSTEM") Save to memcached Get from memcached Token swapping, using Mimikatz driver. Let's hunt it!

62. **Abusing Windows privileges**

63. **Abusing privileges Privilege How it can be used for elevation**

    SeDebugPrivilege A user with this privilege can open any process on the system without regard to the security descriptor present on the process SeImpersonatePrivilege These privileges can be used to act behalf of another user via impersonation mechanism, It can be used to impersonate thread or to spawn process using an elevated token SeAssignPrimaryPrivilege SeTakeOwnershipPrivilege This privilege allows a holder to take ownership any securable object (even process) SeRestorePrivilege A user assigned this privilege can replace any file on the system with her own or change any registry key SeBackupPrivilege A user assigned this privilege can read any file on the system or any registry key SeLoadDriver A malicious user could use this privilege to execute arbitrary code in the kernel SeCreateTokenPrivilege This privilege can be used to generate tokens that represent arbitrary user accounts with arbitrary group membership and privileges assignment SeTcbPrivilege A malicious user can use this privilege to create new logon session that includes the SIDs of more privileged groups or users in the resulting token

64. **Abusing debug privilege Debug privilege (SeDebugPrivilege) allows access to any**

    process or thread, regardless of the process's or thread's security descriptor (except for protected processes). In case of non-administrative account this privilege can be obtained via kernel exploitation or insecure configuration (direct granting SeDebugPrivilege to non-administrative accounts). How it can be used in the context of privilege escalation: • Reading memory of any process ; • Writing to the memory of any process; • Spawning process with arbitrary parent.

65. **Abusing debug privilege. Code injection 1. Discovering user privileges 2.**

    Check groups membership 3. Injecting meterpreter DLL into winlogon.exe process

66. **Abusing debug privilege. Code injection. Let's hunt it! Anomalies, that**

    can be used for hunting: • Injection to the process with higher privileges; • Injected code – is address of LoadLibraryA(W) from kernel32.dll.

67. **source_name:"Microsoft-Windows-Sysmon" AND event_id:8 AND event_data.SourceIntegrityLevel:(Medium High) AND event_data.TargetUser:"NT AUTHORITY\\SYSTEM" AND**

    event_data.TargetIntegrityLevel:System Abusing debug privilege. Code injection. Let's hunt it! Search for injections into the processes with SYSTEM privileges by processes with Medium or High integrity levels: Save to memcached Get from memcached Source process creation event Target process creation event Get from memcached Save to memcached

68. **Abusing debug privilege. Code injection. Let's hunt it! The sane**

    approach can be used for detection of EoP via DLL Hijacking Loading by process with SYSTEM rights of DLL, that was dropped by process with Medium IL

69. **source_name:"Microsoft-Windows-Sysmon" AND event_id:7 AND event_data.IntegrityLevel:System AND event_data.User:"NT AUTHORITY\\SYSTEM" AND event_data.ImageLoadedModifierIntegrityLevel:Medium**

    Abusing debug privilege. Code injection. Let's hunt it! Search for loading by process with SYSTEM rights of DLL, that was dropped by process with Medium IL: Get from memcached Get from memcached Save to memcached

70. **CreateProcess Win32 API allows to assign the parent of a**

    newly spawned process via the PROC_THREAD_ATTRIBUTE_PARENT_PROCESS attribute. This facility is used by UAC when elevated processes are launched by AppInfo service to look like being launched from non-elevated process that would have been the parent, had there been no elevation. Abusing debug privilege. Create process with arbitrary parent

71. **winlogon.exe Process System process.exe Process User Debug Privilege child.exe Process**

    System 2. Process handle 1. OpenProcess 3. CreateProcess Abusing debug privilege. Create process with arbitrary parent How it works 4. Inherit winlogon.exe process token

72. **Abusing debug privilege. Create process with arbitrary parent Mimikatz process::runp**

73. **Spawning of unusual child processes by different system processes. Unusual**

    parent-child combinations Abusing debug privilege. Create process with arbitrary parent Let's hunt it! Lsass.exe spawn cmd.exe Winlogon.exe spawn powershell.exe

74. **event_id:1 AND source_name:"Microsoft-Windows-Sysmon" AND event_data.ParentImage:("\\winlogon.exe" "\\services.exe" "\\lsass.exe" "\\csrss.exe" "\\smss.exe" "\\wininit.exe"**

    "\\spoolsv.exe" "\\searchindexer.exe") AND event_data.Image:("\\cmd.exe" "\\powershell.exe") AND event_data.User:"NT AUTHORITY\\SYSTEM“ AND -event_data.CommandLine:(*route* *ADD*) Abusing debug privilege. Create process with arbitrary parent Let's hunt it! Search for spawning of unusual child processes by different system processes:

75. **Abusing impersonation User A Primary token Process ServerApp.exe Thread 1**

    User A Primary token Thread 2 User B Impersonation token Thread 3 User C Impersonation token Impersonation is the ability of a thread to execute in a security context that is different from the context of the process that owns the thread. Using impersonation process can act behalf of other user. The server thread uses an access token representing the client's credentials to obtain access to the objects to which the client has access. Related privileges: • SeImpersonatePrivilege • SeAssignPrimaryPrivilege

76. **Abusing impersonation**

77. **Abusing impersonation. Difference between CreateProcessAsUser and CreateProcessWithTokenW SeAssignPrimaryPrivilege is required**

    SeImpersonatePrivilege is required

78. **Thread System Impersonation token System Primary token Process PrivProc.exe 1.**

    Influence on process 2. Connection 3. ImpersonateSecurityContext or ImpersonateNamedPipeClient or DdeImpersonateClient or RpcImpersonateCliet System Primary token Process Child.exe 5. CreateProcessWithToken or CreateProcessAsUser User A Primary token Process Parent.exe Impersonation privilege Abusing impersonation. Tricking privileged process connect to us Endpoint Most actions by the thread are done in the security context of the thread's impersonation. But if an impersonating thread calls the CreateProcess function, the new process inherits the primary token of the process.

79. **https://foxglovesecurity.com/2016/09/26/rotten-potato- privilege-escalation-from-service-accounts-to-system/ Abusing impersonation. Rotten Potato Bad news for defenders**

    (good for offenders) – currently ANY user can obtain impersonation SYSTEM token by tricking the SYSTEM account into performing authentication to some TCP listener user control! Good news for defenders (bad for offenders) – to use obtained token SeImpersonatePrivilege or SeAssignPrimaryPrivilege is required (to call the ImpersonateSecurityContext function)…

80. **By default services accounts have impersonation privileges SeAssignPrimaryPrivilege SeImpersonatePrivilege Abusing**

    impersonation. LOCAL/NETWORK SERVICE privileges

81. **Abusing impersonation. LOCAL/NETWORK SERVICE tokens**

82. **Abusing impersonation. MSSQL/IIS accounts token**

83. **Abusing impersonation. Service account –> SYSTEM EoP using Rotten Potato**

    technique 1. Checking current privileges (NETWOR SERVICE) 2. Downloading JucyPotato tool 3. Downloading binary to run with elevated privileges 4. Launching JucyPotato tool 5. Using obtained SYSTEM token to start downloaded binary via CreateProcessWithTokenW API 6. Pwned!

84. **Network/Local service account starts process with SYSTEM rights Abusing impersonation.**

    Service account –> SYSTEM Let’s hunt it!

85. **source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.User:"NT AUTHORITY\\SYSTEM" AND event_data.ParentUser:("NT AUTHORITY\\NETWORK SERVICE"**

    "NT AUTHORITY\\LOCAL SERVICE") AND - event_data.CommandLine:(*rundll32* AND *DavSetCookie*) Abusing impersonation. Service account –> SYSTEM Let’s hunt it! Search for spawning SYSTEM processes by processes, started with Network or Local service account: Get from memcached Save to memcached

86. **Webshell/xp_cmdshell. Let’s hunt it! Spawning cmd/powershell (or other unusual child)**

    by server application

87. **source_name:"Microsoft-Windows-Sysmon" AND event_id:1 AND event_data.Image:("\\cmd.exe" "\\powersehll.exe" "\\wscript.exe" "\\cscript.exe") AND event_data.ParentImage:("\\httpd.exe"**

    "\\sqlserver.exe" "\\jbosssvc.exe" "\\w3wp.exe" "\\httpd.exe" "\\nginx.exe" "\\php-cgi.exe" "\\tomcat8.exe" "\\tomcat7.exe" "\\tomcat6.exe" "\\tomcat5.exe" "\\tomcat.exe") AND -event_data.CommandLine:(*sendmail*) Webshell/xp_cmdshell. Let’s hunt it! Search for cmd/powershell (or other unusual child) by server application:

88. **Cobalt Strike getsystem Abusing impersonation. Named pipe impersonation Meterpreter/Cobalt Strike**

    getsystem (technique 1 – fileless) Meterpreter getsystem How it works: 1. Creates a named pipe; 2. Creates and starts service, that spawn a cmd.exe under SYSTEM which then connects to created named pipe; 3. After cmd had connected to the pipe, impersonates SYSTEM security context, using ImpersonateNamedPipeClient function.

89. **Abusing impersonation. Named pipe impersonation Meterpreter getsystem (technique 2 with**

    file dropping) How it works: 1. Creates a named pipe; 2. Drops special DLL with code to connect to named pipe; 3. Creates and stars service, that spawn a rundll32.exe under SYSTEM which then executes code from DLL; 4. Code from DLL connects to the created named pipe; 5. After cmd had connected to the pipe, impersonates SYSTEM security context, using ImpersonateNamedPipeClient function.

90. **(event_id:7045 OR (event_id:1 AND event_data.ParentImage:"\\services.exe")) AND ((event_data.CommandLine:(*cmd* *COMSPEC*) AND event_data.CommandLine:"*echo**

    *" AND event_data.CommandLine:*pipe* AND event_data.CommandLine.keyword:/.*\\\\.\\..*/) OR (event_data.CommandLine:*rundll* AND event_data.CommandLine.keyword:/.*\.dll,a \/p:.*/)) Abusing impersonation. Named pipe impersonation Meterpreter/Cobalt Strike getsystem. Let’s hunt it! Search for services installation events, where Image Path and command line point to the Meterpreter getsystem command execution (redirection cmd output to the named pipe, specific rundll32 command line):

91. **Spawning process under SYSTEM account by parents with High integrity**

    level Abusing impersonation. Named pipe impersonation Meterpreter/Cobalt Strike getsystem. Let’s hunt it!

92. **Thread System Impersonation token System Primary token Process PrivProc.exe 1.DuplicateToken**

    or DuplicateTokenEx 2. TokenHandle 3.1. SetThreadToken or ImpersonateLoggedOnUser User A Primary token Process Parent.exe Impersonation Debug privileges Abusing impersonation + debug privileges. Steal token of other process via DuplicateToken(Ex) In this case also if an impersonating thread calls the CreateProcess function, the new process inherits the primary token of the process rather than the impersonation token of the calling thread. System Primary token Process Child.exe 3.2. CreateProcessWithToken or CreateProcessAsUser

93. **Well-known Incognito tool Abusing impersonation + debug privileges. Incognito https://github.com/fdiskyou/incognito2**

    1. Lists available tokens 2. Executes cmd with SYSTEM token 3. Checks current rights – we are the SYSTEM

94. **Abusing impersonation + debug privileges. Incognito Let's hunt it! Spawning**

    process under SYSTEM account by parents with High integrity level

95. **Search for spawning process under SYSTEM account by parents with**

    High integrity level: event_id:1 AND source_name:"Microsoft-Windows-Sysmon" AND event_data.IntegrityLevel:System AND event_data.User:"NT AUTHORITY\\SYSTEM" AND event_data.ParentIntegrityLevel:(Medium High) Abusing impersonation + debug privileges. Tokenvator Let's hunt it!

96. **Search for spawning child process under account, which is different**

    from parent process (excluding parent processes for which such activity is legitimate – runas tool for example): {"bool":{"must":[{"query_string":{"query":" event_id:1 AND event_data.ParentIntegrityLevel:(High Medium Low) AND -event_data.Image:\"\\runas.exe\" AND -(event_data.Image:\"rundll32.exe\" AND event_data.CommandLine:*RunAsNewUser_RunDLL*) "}},{"script":{"script":" doc[\"event_data.User.keyword\"] != doc[\"event_data.ParentUser.keyword\"] "}}]}} Generic detector of token swapping Enrichment using logstash memcached filter plugin

97. **Search for spawning whoami tool under SYSTEM account source_name:"Microsoft-Windows-Sysmon" AND**

    event_id:1 AND event_data.Image:"\\whoami.exe" AND event_data.IntegrityLevel:System AND event_data.User:"NT AUTHORITY\\SYSTEM" Generic detector of privilege escalation to SYSTEM

98. **Summing Up**

99. **Questions?**