Skeleton Key Malware Analysis

- **Author:** Dell SecureWorks Counter Threat Unit™ Threat Intelligence
- **Date:** 12 January 2015
- **URL:** http://www.secureworks.com/cyber-threat-intelligence/threats/skeleton-key-malware-analysis/

Summary

Dell SecureWorks Counter Threat Unit(TM) (CTU) researchers discovered malware that bypasses authentication on Active Directory (AD) systems that implement single-factor (password only) authentication. Threat actors can use a password of their choosing to authenticate as any user. This malware was given the name "Skeleton Key."

CTU researchers discovered Skeleton Key on a client network that used single-factor authentication for access to webmail and VPN, giving the threat actor unfettered access to remote access services. Skeleton Key is deployed as an in-memory patch on a victim's AD domain controllers to allow the threat actor to authenticate as any user, while legitimate users can continue to authenticate as normal. Skeleton Key's authentication bypass also allows threat actors with physical access to login and unlock systems that authenticate users against the compromised AD domain controllers.

The only known Skeleton Key samples as of this publication lack persistence and must be redeployed when a domain controller is restarted. CTU researchers suspect that threat actors can only identify a restart based on their inability to successfully authenticate using the bypass, as no other malware was detected on the domain controllers. Between eight hours and eight days of a restart, threat actors used other remote access malware already deployed on the victim's network to redeploy Skeleton Key on the domain controllers.

Skeleton Key requires domain administrator credentials for deployment. CTU researchers have observed threat actors deploying Skeleton Key using credentials stolen from critical servers, administrators' workstations, and the targeted domain controllers.

Analysis

CTU researchers initially observed a Skeleton Key sample named ole64.dll on a compromised network (see Table 1).

Attribute	Value or description
Filename	ole64.dll
MD5	bf45086e6334f647fda33576e2a05826

SHA1	5083b17ccc50dd0557dfc544f84e2ab55d6acd92
Compile time	2014-02-19 09:31:29
Deployed	As required (typically downloaded using malware and then deleted after use)
File size	49664 bytes
Sections	.text, .rdata, .data, .pdata, .rsrc, .reloc
Exports	ii (installs the patch) uu (uninstalls the patch) DllEntryPoint (default DLL entry point)

Table 1. Skeleton Key sample ole64.dll.

When investigating ole64.dll, CTU researchers discovered an older variant named msuta64.dll on a "jump host" in the victim's network (see Table 2). The jump host is any system previously compromised by the threat actors' remote access malware. This variant includes additional debug statements, which allow the Skeleton Key developer to observe the memory addresses involved in the patching process.

Attribute	Value or description
Filename	msuta64.dll
MD5	66da7ed621149975f6e643b4f9886cfd
SHA1	ad61e8daeeba43e442514b177a1b41ad4b7c6727
Compile time	2012-09-20 08:07:12
Deployed	2013-09-29 07:58:16
File size	50688 bytes
Sections	.text, .rdata, .data, .pdata, .rsrc, .reloc
Exports	i (installs the patch) u (uninstalls the patch) DllEntryPoint (default DLL entry point)

Table 2. Skeleton Key sample msuta64.dll.

The threat actors used the following process to deploy Skeleton Key as a 64-bit DLL file:

1. Upload the Skeleton Key DLL file to a staging directory on a jump host in the victim's network. CTU researchers have observed three filenames associated with the Skeleton Key DLL file: ole64.dll, ole.dll, and msuta64.dll. Windows systems include a legitimate ole32.dll file, but it is not related to this malware.
2. Attempt to access the administrative shares on the domain controllers using a list of stolen domain administrator credentials.
3. If the stolen credentials are no longer valid, use password theft tools to extract clear text domain administrator passwords from one of the following locations, which suggest a familiarity with the victim's environment:

    - memory of another accessible server on the victim's network
    - domain administrators' workstations
    - targeted domain controllers
4. Use valid domain administrator credentials to copy the Skeleton Key DLL to C:\WINDOWS\system32\ on the target domain controllers.
5. Use the PsExec utility to run the Skeleton Key DLL remotely on the target domain controllers using the rundll32 command. The threat actor's chosen password is formatted as an NTLM password hash rather than provided in clear text. After Skeleton Key is deployed, the threat actor can authenticate as any user using the threat actor's configured NTLM password hash:
   psexec -accepteula \\*%TARGET-DC%* rundll32 *<DLL filename>* ii *<NTLM password hash>*
6. Delete the Skeleton Key DLL file from C:\WINDOWS\system32\ on the targeted domain controllers.
7. Delete the Skeleton Key DLL file from the staging directory on the jump host.
8. Test for successful Skeleton Key deployment using "net use" commands with an AD account and the password that corresponds to the configured NTLM hash.

CTU researchers have observed a pattern for the injected password that suggests that the threat group has deployed Skeleton Key in multiple organizations.

The use of PsExec can be detected within a Windows environment by alerting on the Windows events generated by the utility. The following Event IDs observed on the targeted domain controllers record the PsExec tool installing its service, starting the service, and stopping the service. These events are created every time PsExec is used, so additional analysis of the events is required to determine if they are malicious or legitimate:

- Unexpected PSEXESVC service install events (event ID 7045) on AD domain controllers:
  Log Name: System
  Source: Service Control Manager
  Summary: A service was installed in the system.
  Service File Name: %SystemRoot%\PSEXESVC.exe
- Unexpected PSEXESVC service start / stop events (event ID 7036) on AD domain controllers:
  Log Name: System
  Source: Service Control Manager
  Summary:
    - "The PSEXESVC service entered the running state."
    - "The PSEXESVC service entered the stopped state."

When run, Skeleton Key performs the following tasks:

1. Check for one of the following compatible 64-bit Windows versions. The malware is not compatible with 32-bit Windows versions or with Windows Server versions beginning with Windows Server 2012 (6.2).
    - 6.1 (Windows 2008 R2)
    - 6.0 (Windows Server 2008)
    - 5.2 (Windows 2003 R2)
2. Use the SeDebugPrivilege function to acquire the necessary administrator privileges to write to the Local Security Authority Subsystem Service (LSASS) process. This process controls security functions for the AD domain, including user account authentication.
3. Enumerate available processes to acquire a handle to the LSASS process.
4. Obtain addresses for the authentication-related functions that will be patched:
    - CDLocateCSystem — located in cryptdll.dll
    - SamIRetrieveMultiplePrimaryCredentials — located in samsrv.dll
    - SamIRetrievePrimaryCredentials — located in samsrv.dll
5. Perform OS-specific adjustments using the global variable set during the compatibility check in Step 1.
6. Use the OpenProcess function to acquire a handle to the LSASS process.
7. Reserve and allocate the required memory space to edit and patch the LSASS process's memory.
8. Patch relevant functions based on the operating system:
    - CDLocateCSystem (all compatible Windows versions)
    - SamIRetrieveMultiplePrimaryCredentials (only Windows 2008 R2 (6.1))
    - SamIRetrievePrimaryCredentials (all compatible Windows versions other than Windows 2008 R2 (6.1))

Skeleton Key performs the following steps to patch each function:

1. Call the VirtualProtectEx function to change the memory protection to allow writing to the required memory allocations (PAGE_EXECUTE_READWRITE, 0x40). This step allows the function's code to be updated in memory.
2. Call the WriteProcessMemory function to change the address of the target function to point to the patched code. This change causes calls to the target function to use the patch instead.
3. Restore the original memory protection by calling VirtualProtectEx with the original memory protection flags. This step is likely to avoid suspicious writable and executable memory allocations.

After patching, the threat actor can use the Skeleton Key password configured at the time of deployment to log in as any domain user. Legitimate users can still log in using their own passwords. This authentication bypass applies to all services that use single-factor AD authentication, such as web mail and VPNs, and it also allows a threat actor with physical access to a compromised system to unlock the computer by typing the injected password on the keyboard.

Possible link to domain replication issues

The Skeleton Key malware does not transmit network traffic, making network-based detection ineffective. However, the malware has been implicated in domain replication issues that may indicate an infection. Shortly after each deployment of the Skeleton Key malware observed by CTU researchers, domain controllers experienced replication issues that could not be explained or addressed by Microsoft support and eventually required a reboot to resolve. These reboots removed Skeleton Key's authentication bypass because the malware does not have a persistence mechanism. Figure 1 shows the timeline of these reboots and the threat actors' subsequent password theft, lateral expansion, and Skeleton Key deployment. Redeployments typically occurred within several hours to several days of the reboot.

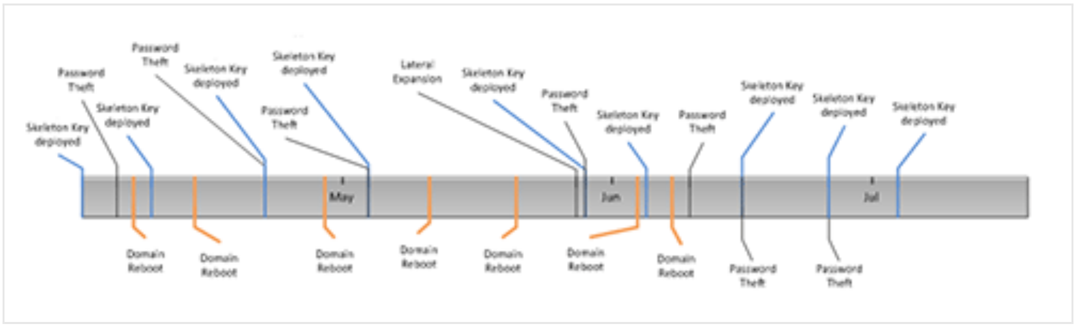

Figure 1. Relationships of deployments and reboots observed by CTU researchers, April - July 2014. (Source: Dell SecureWorks)

Countermeasures

The Skeleton Key malware bypasses authentication and does not generate network traffic. As a result, network-based intrusion detection and intrusion prevention systems (IDS/IPS) will not detect this threat. However, CTU researchers wrote the YARA signatures in Appendix A to detect a Skeleton Key DLL and the code it injects into the LSASS process's memory.

Threat indicators

The threat indicators in Table 3 can be used to detect activity related to the Skeleton Key malware.

Indicator	Type	Context
66da7ed621149975f6e643b4f9886cfd	MD5 hash	Skeleton Key patch msuta64.dll
ad61e8daeeba43e442514b177a1b41ad4b7c6727	SHA1 hash	Skeleton Key patch msuta64.dll
bf45086e6334f647fda33576e2a05826	MD5 hash	Skeleton Key patch ole64.dll
5083b17ccc50dd0557dfc544f84e2ab55d6acd92	SHA1 hash	Skeleton Key patch ole64.dll

Table 3. Indicators for the Skeleton Key malware.

Conclusion

The CTU research team recommends that organizations implement the following protections to defend against the Skeleton Key malware:

- Multi-factor authentication for all remote access solutions, including VPNs and remote email, prevents threat actors from bypassing single-factor authentication or authenticating using stolen static credentials.
- A process creation audit trail on workstations and servers, including AD domain controllers, may detect Skeleton Key deployments. Specifically, organizations should look for the following artifacts:
    - Unexpected PsExec.exe processes and the use of the PsExec "-accepteula" command line argument
    - Unexpected rundll32.exe processes
    - Process arguments that resemble NTLM hashes (32 characters long, containing digits 0-9 and characters A-F)
- Monitoring Windows Service Control Manager events on AD domain controllers may reveal unexpected service installation events (event ID 7045) and service start/stop events (event ID 7036) for PsExec's PSEXESVC service.

Appendix A — YARA signatures

The following YARA signatures detect the presence of Skeleton Key on a system, by scanning either a suspicious file or a memory dump of Active Directory domain controllers suspected to contain Skeleton Key.

```
rule skeleton_key_patcher
{
strings:
       $target_process = "lsass.exe" wide
       $dll1 = "cryptdll.dll"
       $dll2 = "samsrv.dll"

       $name = "HookDC.dll"

       $patched1 = "CDLocateCSystem"
       $patched2 = "SamIRetrievePrimaryCredentials"
       $patched3 = "SamIRetrieveMultiplePrimaryCredentials"

condition:
       all of them
}



rule skeleton_key_injected_code
{
strings:
       $injected = { 33 C0 85 C9 0F 95 C0 48 8B 8C 24 40 01 00 00 48 33 CC E8
4D 02 00
           00 48 81 C4 58 01 00 00 C3 }

           $patch_CDLocateCSystem = { 48 89 5C 24 08 48 89 74 24 10 57 48 83
EC 20 48 8B FA
           8B F1 E8 ?? ?? ?? ?? 48 8B D7 8B CE 48 8B D8 FF 50 10 44 8B D8 85
C0 0F 88 A5 00
           00 00 48 85 FF 0F 84 9C 00 00 00 83 FE 17 0F 85 93 00 00 00 48 8B
07 48 85 C0 0F
           84 84 00 00 00 48 83 BB 48 01 00 00 00 75 73 48 89 83 48 01 00 00
33 D2 }

           $patch_SamIRetrievePrimaryCredential = { 48 89 5C 24 08 48 89 6C 24
10 48 89 74
           24 18 57 48 83 EC 20 49 8B F9 49 8B F0 48 8B DA 48 8B E9 48 85 D2
74 2A 48 8B 42
           08 48 85 C0 74 21 66 83 3A 26 75 1B 66 83 38 4B 75 15 66 83 78 0E
73 75 0E 66 83
           78 1E 4B 75 07 B8 A1 02 00 C0 EB 14 E8 ?? ?? ?? ?? 4C 8B CF 4C 8B
C6 48 8B D3 48
           8B CD FF 50 18 48 8B 5C 24 30 48 8B 6C 24 38 48 8B 74 24 40 48 83
C4 20 5F C3 }

           $patch_SamIRetrieveMultiplePrimaryCredential  = { 48 89 5C 24 08 48
89 6C 24 10
```

```
            48 89 74 24 18 57 48 83 EC 20 41 8B F9 49 8B D8 8B F2 8B E9 4D 85
C0 74 2B 49 8B
            40 08 48 85 C0 74 22 66 41 83 38 26 75 1B 66 83 38 4B 75 15 66 83
78 0E 73 75 0E
            66 83 78 1E 4B 75 07 B8 A1 02 00 C0 EB 12 E8 ?? ?? ?? ?? 44 8B CF
4C 8B C3 8B D6
            8B CD FF 50 20 48 8B 5C 24 30 48 8B 6C 24 38 48 8B 74 24 40 48 83
C4 20 5F C3 }

condition:
       any of them
}
```