KILLSUIT RESEARCH

F-Secure Countercept Whitepaper

F-Secure

CONTENTS

WHAT IS KILLSUIT AND HOW DOES IT WORK

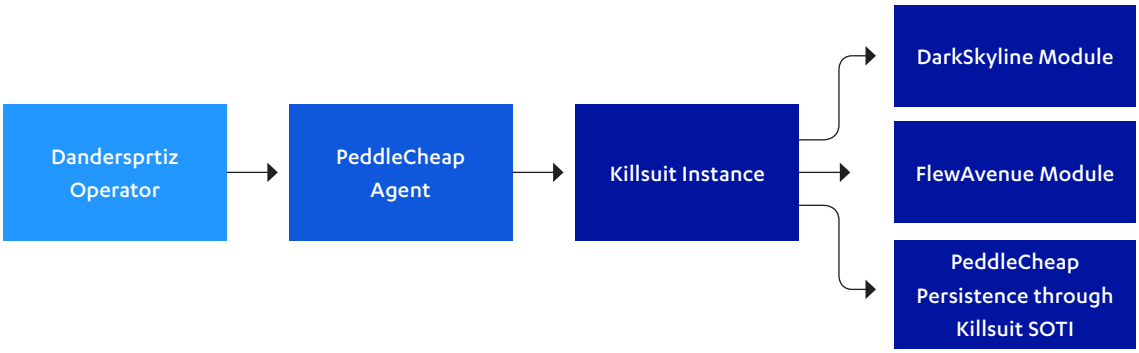
What is Killsuit

Killsuit (KiSu) is a modular persistence and capability mechanism employed in post-exploitation frameworks including Danderspritz (DdSz), which was developed by The Equation Group and leaked in April 2017 by The Shadow Brokers as part of the "Lost in Translation" leak. KiSu is used for two reasons - it enables persistence on a host and it works as a sort of catalyst allowing specific exploitative functions to be conducted.

How does Killsuit infect a machine

As KiSu is a post-exploitation tool it is used as part of a hands-on-keyboard attack where a malicious actor is actively compromising a network. The DdSz exploitation framework includes various tools including PeddleCheap (PC), a payload that can allow for a highly tuned interaction with a compromised host. PC is a post-exploitation tool that can install KiSu instances on a host in order to run its various capabilities as part of the attacker's process. Although PC is loaded onto a host typically though a tool such as DoublePulsar and as such injected into a running process, KiSu is installed deliberately as an action of the PC payload as a post-exploitation operation.

FIGURE 01 – KILLSUIT INSTALLATION AND FUNCTION DIAGRAM



How does it work

KiSu is a facilitator for specific functions within the DdSz framework. As such KiSu is not a malicious actor itself, but rather a component for other operations; it essentially works as a local repository for installation and operation for other tools. Each instance is installed into an encrypted database within the registry. In order to utilize the appropriate instance the operator must "connect" to the instance and then perform relevant actions. It is worth noting that a PC agent can only "connect" to one KiSu instance for the duration of its operation. These instances each have their own specialized functionality associated with specific tools such as "StrangeLand" (StLa), which is used for covert keylogging using the Strangeland keylogger; "MagicBean" (MaBe), which is used for WiFi man-in-the-middle (MITM) attacks by installing necessary drivers for packet injections, and many others (full list in appendix).

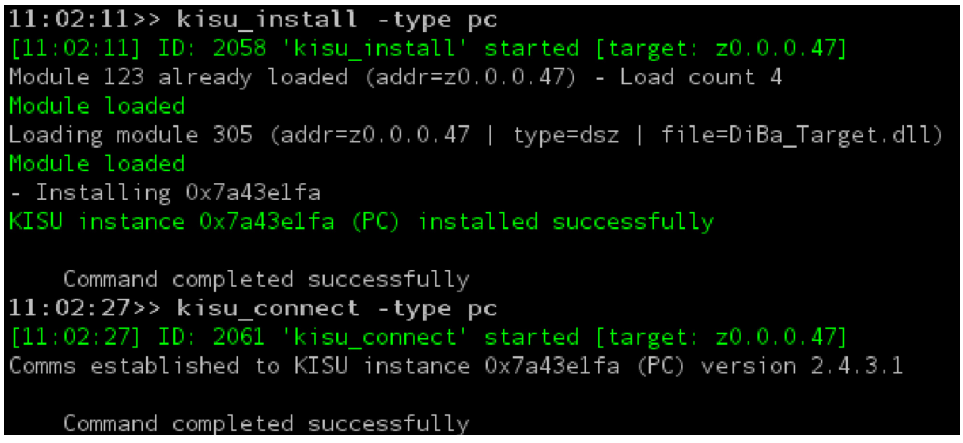
DecibelMinute (DeMi) is believed to be the controller for KiSu installation and module management. This framework component is seen in operation during instance/module installations. As its name suggests, this is a stealthy mechanism that can bypass some driver signing issues that may be encountered with installations by using the connected KiSu instance to facilitate the process. DeMi also can install modules into an instance from the internal module store to increase capabilities - such as FlewAvenue (FlAv), DoormanGauze (DmGz) or DarkSkyline - which are used for the frameworks proprietary IPv4 & IPv6 network stacks and network monitoring. However, if the related instances are removed from the host the specialised tools and loaded modules no longer function.

How to deploy and configure a KillSuit instance

As mentioned this tool allows for customised operational configurations through its instance & modular structure. One of the instance types allows for a persistence of the PC agent primary to the DdSz framework operations (the PC instance type). Although there are multiple persistence methods, the KiSu related persistence method for PC is one of the more advanced and stealthy methods in the frameworks arsenal as it can use the SOTI bootloader.

In order to use the KiSu persistence you must first install the PC instance on the host using the command "KiSu_install –type pc". This installs the necessary data/packages to the host within the encrypted registry DB to be utilised during persistence installation. Next you must connect to the newly created instance using command "kisu_connect –type pc". This tells the current PC agent that we are connecting to the PC KiSu instance on the host, as previously mentioned an agent can only connect to one instance at a time (therefore can only use one KiSu instance at a time although you can have many installed). Now we run "pc_install", this is what creates the persistence.

Figure 02 – Installation and connection to PC KiSu instance



This command will generate a menu with a number of options and status information. You will see that one says "KiSu connection", this checks that the session has an active connection to a PC instance type on the host. If the connection is not there it will prompt whether the user wants to install or connect to an instance, it is pretty user friendly for such a sophisticated tool.

Figure 03 – Default values for persistence installation

```
- Pc Install
-
- Current Configuration:
-      Load Method : AppCompat
-     Process Name : lsass.exe
-       COMMS Type : Winsock
-     Trigger Name : ntfltmgr
-          Payload : None
-  KiSu Connection : Connected
-
```

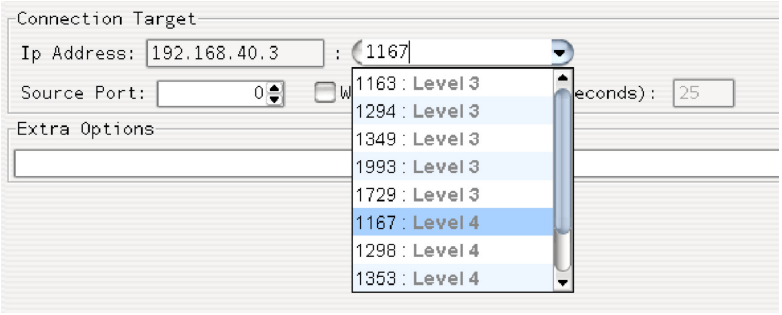
Now you can change the load method, this has a default of "AppCompat" but we want to change it to "KillSuit". You will see that on selecting the option the status changes. However, one stat is still yellow which is "Payload". You must create a new PC payload on the host to use for the persistence. Now one of the key things to note here is the payload level, this is a level 4 PC payload which is relevant when trying to connect to it later. Select the payload type you want and run through the options the same as you would have done for creating the original PC instance. You can install a knocking trigger for this payload if you want which is a powerful triggering mechanism, but we will talk about that another time. Once the payload is created you will be taken back to the menu, finally with no yellow options.

Figure 04 – KiSu persistence configuration

```
- Pc Install
-
- Current Configuration:
-      Load Method : KillSuit
-     Process Name : lsass.exe
-       COMMS Type : Winsock
-     Trigger Name : ntfltmgr
-          Payload : Standard TCP
-                    PeddleCheap_2019_02_20_11h02m47s.598/PC_Level4_dll.configured
-  KiSu Connection : Connected
-
```

Select installation and the script will communicate with the connected instance and install the relevant persistence. Now you can safely reboot the machine. For reconnection, while in the PC tab change the "connection target" port from the drop down to a level 4 connection port. If you do not make this change you will not be able to connect to the PC instance type that is loaded at boot on your target machine. You now have a KiSu persistence instance of PC on the target machine.

Figure 05 – Connection level for persistence KiSu PC instance



Extra capabilities – Some modular functions

DarkSkyline

DarkSkyline (DSky) is a packet capture utility that can be installed as part of any KiSu instance by installing the associated module from the module store. By connecting to a specific instance, typically the PC persistence module, the operator can install the necessary module for the DSky tool to operate. To do this the operator needs to use the command "darkskyline –method demi" in order to specify using the DSky control mechanism in association with the KiSu control element DeMi. As when installing persistence, the menu will display certain criteria in different colours, if you have followed the steps correctly the "Use DecibelMinute" & "Connected" options should both be green with the value "true". If this is the case you can then select "install tools" followed by "load driver".

Figure 06 – DarkSkyline configuration options for execution

```
-     SUCCESS (SYSTEM\CurrentControlSet\Services\secdrv\Parameters)
- DSky Control (DSky 3.0.1)
-
- Current Configuration:
-         Driver Name : tdi6
-        Capture File : \SystemRoot\Fonts\simtrbx.tff
-  Capture File Win32 : C:\Windows\Fonts\simtrbx.tff
-      Encryption Key : a4 cf a8 b1 a8 42 4c 36 7a c6 64 82 0f e7 c1 05
-   Use DecibelMinute : True
-           Connected : True
-        Connected To : 0x7a43e1fa - PC
-
```

Verify both installation and driver running, if both are success then you can start packet capture activity on the target. This can be controlled through either the control mechanism or specific scripts such as "DSky_start". It is worth noting that the file used to store the packet captures is in the tff (true font file) format that is also used for the SOTI persistence mechanism. Once sufficient capture has occurred, the operator can then request to retrieve the captures which are automatically attempted to be compiled into a pcap file for analysis.

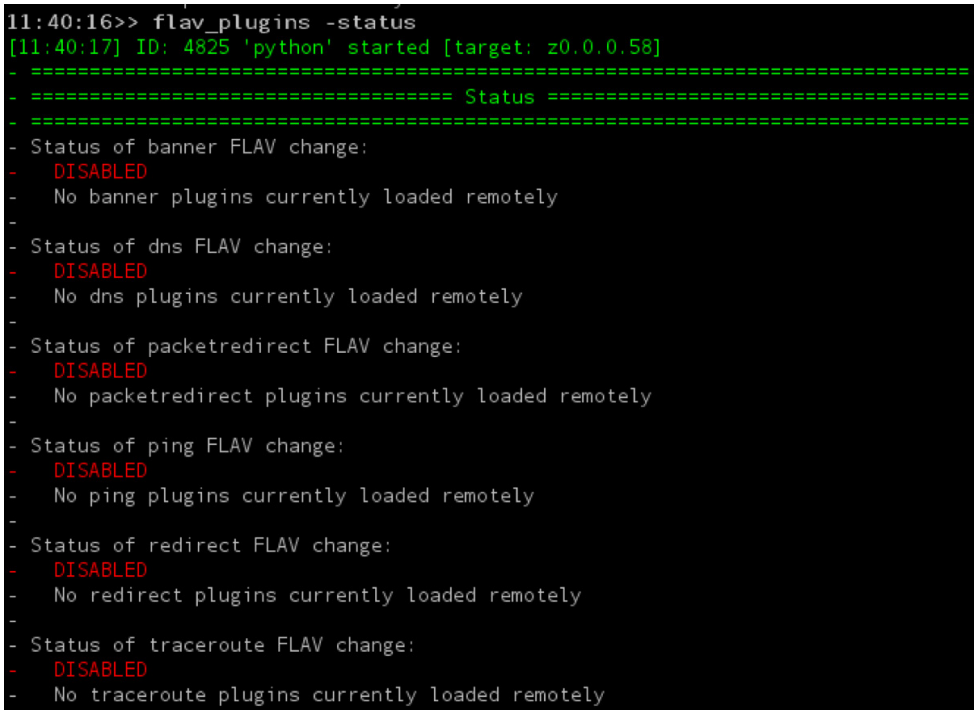
Figure 07 – Executing DarkSkyline packet collection from capture session

```
- Control Commands
-  15) Start capturing
-  16) Stop capturing
-  17) Get capture file
-  18) Delete capture file
-
- KiSu Commands
-  19) Disconnect From Kisu
Enter the desired option
17
- Getting capture file (C:\Windows\Fonts\simtrbx.tff)
-     SUCCESS (1056416 bytes)
- Parsing capture file into .pcap files
```

FlewAvenue

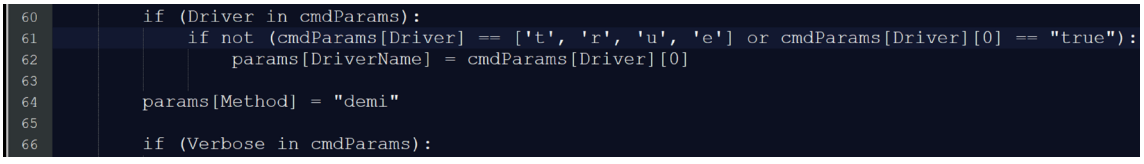
FlewAvanue (flav) is the codename for the custom TCP/IP IPV4 stack that was created for this tool in order to avoid detection. By installing flav you can control plugins of the custom TCP/IP stack including packet redirection, flav dns, flav traceroute and others that are otherwise unavailable.

Figure 08 – FlewAvenue plugin status on initial PC connection



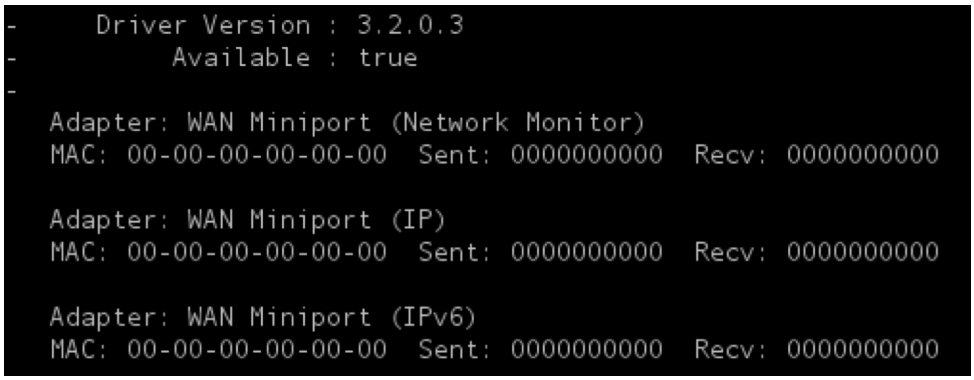
When trying to install flav the operator may encounter issues related to driver signing (especially on Windows 7 pro), even if test signing is enabled and integrity checks are disabled the tools idiot proofing can still prevent the module from being loaded as it incorrectly decides driver signing is still enforced. In order to circumvent this the operator can load flav as a module into a KiSu instance, in testing you will need to edit the _FLAv.py script to achieve this due to an error. Remove line 64 & 65 and replace with the value 'params[Method] = "demi"', this will force the flav controller to comply.

Figure 09 – Work around to override driver warning and install



Once this is done ensure you are connected to an instance then run “flewavenue”, select “install tools” then “load driver” and then verify the installation. If you check the driver status it may not be available, if this is the case you need to restart or wait for the user to restart the host in order for the module to be loaded at the KiSu instance restart.

Figure 10 – Verify installation showing FlewAvenue as “Available”



Once the driver is installed and available, the operator can start creating traffic redirects, targeted redirects, packet redirects and others using commands such as “hittun”, “imr” and “packetredirect” which format redirect commands for them.

HOW DOES KILLSUIT INSTALL ITSELF ON A SYSTEM

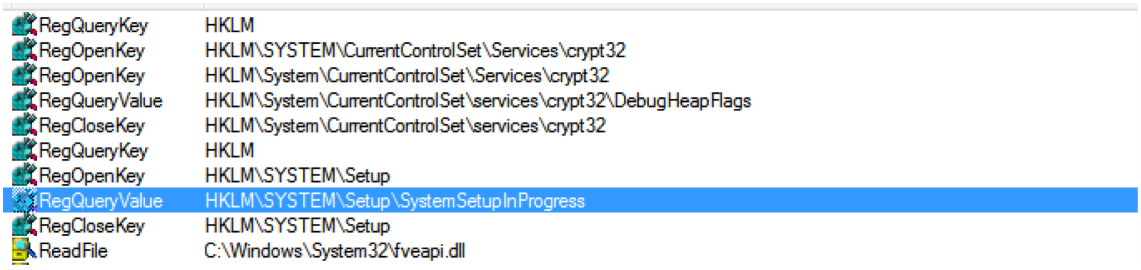
Initial run at installation identifiers analysis

These identifying observations are made against the leaked version of KiSu made available by the ShadowBrokers as part of the "lost in translation" shadowbroker DdSz leak (https://github.com/misterch0c/shadowbroker).

In our research, when the PC agent starts installing KiSu it uses the internal resource library DeMi which as stated is believed to manage the KiSu installation and associated modules on the target. During the installation process the local agent runs a huge amount of redundant API calls, dll loads and system operations in order to temporarily generate massive amounts of debug information. Some of these operations have been shown to be dummies with the supposed specific intention of exacerbating research and reversing of the tool. However, careful examination of logs associated with these operations highlighted points of interest.

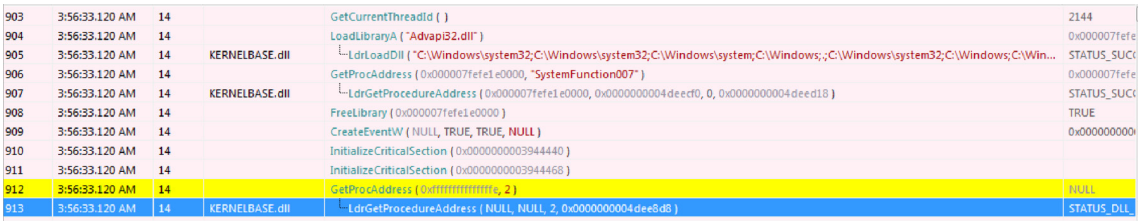
When installing to a host one of the first and last checks the agent makes is the running system mode, primarily whether it is in setup or normal run mode. It does this by querying the registry value "HKLM\System\Setup\SetupInProgress", during normal operation this value is set to 0. Alteration of this value did not affect the installation of instances, indicating this could potentially be a dummy operation or checking for a specific value outside the standard options.

Figure 11 –Killsuit installation check for value SystemSetupInProgress (OS running mode)



One of the noted actions is that the PC instance will make a Kernel API call for "systemfunction007" which is used to generate NTLM hashes for encrypted communication. Since during our testing we injected PC into a calc.exe instance, as you might expect this should not occur and stood out a bit.

Image 12 - API collection showing "systemfunction007" kernel operation in calc.exe thread



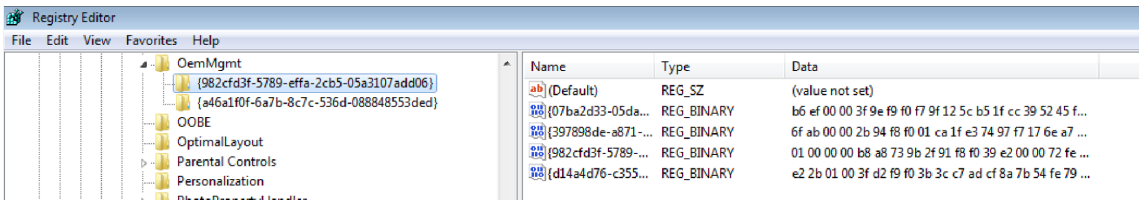
Immediately following this the generated hash values are used in operations with the Kernel crypto modules before being stored in the registry. This likely indicates the hash is used as part of the DB encryption operation or is generated for the encrypted communication channel operation DdSz employs. The second thing is the registry where the generated hashes were stored, namely under "HKLM/SOFTWARE/Microsoft/Windows/CurrentVersion/OemMgmt". Although this looks like a legitimate registry location (as other oem related dirs are here), there is no legitimate registry dir with this path we could discover.

Image 13 – API collection showing Unicode operation for registry keys under dir "OemMgmt"

1409	3:56:33.151 AM	14	CloseHandle (0x0000000000000688)	TRUE
1410	3:56:33.151 AM	14	RtlInitUnicodeString (0x0000000004deece8, "\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\OemMgmt\{982cf...	
1411	3:56:33.151 AM	14	GetCurrentProcessId ( )	832

This registry directory is created at installation of the first instance and is removed when all instances are uninstalled. All instance types have their own corresponding 36 char ID registry entry that looks like "{982cfd3f-5789-effa-2cb5-05a3107add06}" within this directory, these contain keys holding the stored encrypted values including the encrypted communication key and other KiSu configuration data.

Image 14 – Registry Edit displaying the malicious registry entries for two installed KiSu instances



Further investigation of this location found that the path corresponded to the "KiSu Module Root" location specified during usage of the "DoubleFeature" function of the framework. This function is designed so that operators can quickly assess what elements have been previously installed on a target for record keeping but also in case there are multiple operators attacking one network. By generating a number of target machines we determined that the KiSu module root location changes between hosts with the section "OemMgmt" varying between hosts, below is an example of a variant module root being displayed using DoubleFeature.

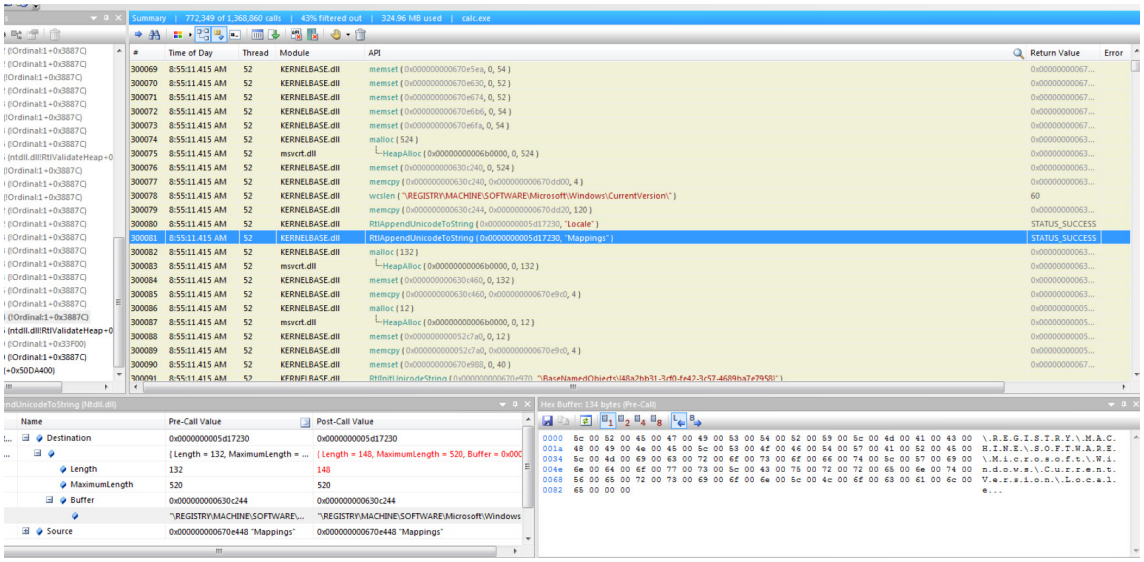
Figure 15 –DoubleFeature display of Killsuit module root location

```
DiveBar Instance Count: 1

KSModule Store Root: \REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\DrivMgmt
```

Through experimenting with OS version, IP address, MAC address, time of OS initialisation, various system identification values and many other factors we were unable to determine the variable that was used to select the masquerading registry value where the modules were stored. Even when two separate OS instances were made with seemingly identical criteria the location would vary, therefore concluding it must not be a standard configurable value. However, after many attempts we began to see repetition in the location selected for installation in parts of the path generated which led to the speculation this location is generated from two separate words spliced together (Oem & Mgmt or Driv & Mgmt e.g). Analysis of the deployed PC operation proved this to be the case as two strings values are observed being concatenated together and appended to the default registry directory during installation. As such we began to work on tracking down when the hive location was selected and where the two names were selected from with the belief that two lists must exist within the framework.

Figure 16 –Observation of concatenation function for Killsuit module root location variable string components



Hunting for the list & into the rabbit hole

We needed to determine if the registry location was decided by the DdSz operator on the attacker machine via received information, configured as part of the PC agent payload generation for that host or generated by the PC agent once installed on the host. By utilising the same PC payload on multiple hosts we were able to quickly rule out the value being coded into the payload as reuse of the payload resulted in varying registry addresses.

Therefore we moved to the operator to examine the Alias and Commands of the framework for all functions related to installation. From this we were able to find scripts that were directly called in order to facilitate KiSu operations and gather a better understanding of how these functions are processed in the framework. Essentially the GUI of the framework translated command line input through the Alias and Command filters to the appropriate scripts, these (for KiSu) then are fed to Dsz script files which interact with a _dsz object type which seemingly allocates, controls and manages PC agents in the field.

Examples of these scripts include "Mcl_Cmd_DiBa_Tasking.py" which handles KiSu module installation/maintenance operations for instances, and "Mcl_Cmd_KisuComms_Tasking.pyo" which is used to dynamically load and unload modules/drivers from an instance and initiate connection between and agent and an instance. Both these scripts are called through the command line and relay & format the operators input to the Dsz resource libraries to perform operations against the agent specified, in the image below this can be seen as the command "mcl.tasking.RpcPerformCall()".

Figure 17 – Mcl_Cmd_KisuComms_Tasking.pyo extracted function utilising library functions with _dsz component

```
def _handleConnect(namespace):
    import mcl.imports, mcl.target, mcl.tasking
    from mcl.object.Message import MarshalMessage
    mcl.imports.ImportWithNamespace(namespace, 'mcf.cmd.kisucomms', globals())
    mcl.imports.ImportWithNamespace(namespace, 'mcf.cmd.kisucomms.tasking', globals())
    lpParams = mcl.tasking.GetParameters()
    import mcl.tasking.env
    if mcl.tasking.env.CheckValue(mcf.cmd.kisucomms.COMMS_ESTABLISHED_ENV, globalValue=True):
        mcl.tasking.OutputError('Already connected to a KISU instance')
        return False
    tgtParams = mcf.cmd.kisucomms.ParamsConnect()
    tgtParams.id = _getKisuId(namespace, lpParams['instance'], lpParams['type'])
    rpc = mcf.cmd.kisucomms.tasking.RPC_INFO_CONNECT
    msg = MarshalMessage()
    tgtParams.Marshal(msg)
    rpc.SetData(msg.Serialize())
    rpc.SetMessagingType('message')
    res = mcl.tasking.RpcPerformCall(rpc)
    if res != mcl.target.CALL_SUCCEEDED:
        mcl.tasking.RecordModuleError(res, 0, mcf.cmd.kisucomms.errorStrings)
        return False
    return True
```

By following the script commands for installation back through their associated libraries we found the command was issued to the agent through the Dsz resource folder "mcl_platform/tasking" library. In this and associated libraries the use of a "_dsz" import is utilised to create objects and in order to carry out the interactions with the agents, however no corresponding library file was found for the private library.

As this import seemed pivotal to the frameworks operations and KiSu interaction we investigated its use within the "ddsz_core.exe" binary file by searching for any instace of any of the associated scripts or their functions. Through this method we successfully found calls to the function "dsz_task_perform_rpc", by cross referencing this function we were able to uncover data locations for related data objects.

Figure 18 –Radare2 output for search and cross reference of data location relative to _dsz function output

```
[0x0043b808]> izz~dsz_task_perform_rpc
vaddr=0x004b0944 paddr=0x000afd44 ordinal=7916 sz=21 len=20 section=.rdata type=ascii string=dsz_task_perform_rpc
vaddr=0x004b1a20 paddr=0x000b0e20 ordinal=8052 sz=54 len=53 section=.rdata type=ascii string=Mcl_Cmd_Python::dsz_task_perform_rpc had an exception
[0x0043b808]> axt 0x004b1a20
data 0x43edda mov dword [esp], str.Mcl_Cmd_Python::dsz_task_perform_rpc_had_an_exception in unknown function
[0x0043b808]> s 0x43edda
[0x0043edda]>
```

Analysis of the binary showed that the relevant data was not in the binary as standard and was instead loaded as an external C module at runtime, making any attempt to analyse the command functions statically impossible, therefore we moved on to dynamic analysis. Attempting to analyse the binary dynamically produced issues as the binary is aware of debugger hooks and automatically terminates. By hiding the debugger we were able to gain monitoring of the binary but this led to more complications when referencing the function index table as a number of loaded functions are dummy functions with no purpose, a static alternative to the dummy operation seen during KiSu installation.

Further analysis of the discovered memory locations showed several variations most of which remained empty, additionally the values seemed to be loaded and unloaded immediately after use. From these elements it is clear that analysis of this binary was designed to be as difficult as possible. Attempting to analyse the PC payload showed its contents to be encrypted and impossible to analyse as deployed. We attempted to use PcPrep.exe which is included within the framework which provides additional information on PC payloads. However, the information listed does not include the root location and therefore was not conclusive.

Figure 19 –PcPrep.exe output for configured PeddleCheap payload and associated configuration file

```
D:\Resources\Pc\Tools\i386-winnt>PCConfig -display -input D:\
inal
         Version : 2.3.0
              Id : 0x0000000000000000
           Flags : 0x00000088
                       PCHEAP_CONFIG_FLAG_LEVEL3
                       PCHEAP_CONFIG_FLAG_DONT_CREATE_WINDOW
    Listen Times : ALWAYS
 Listen Duration : 300 seconds
    Listen Loops : 6
Listen Bind Addr : 0.0.0.0
    Listen Ports : 5 (1163, 1294, 1349, 1993, 1729)
   Callback Addr : 127.0.0.1
  Callback Ports : 4 (0 -> 443, 0 -> 80, 0 -> 53, 0 -> 1509)
      Info Value : ''
      Internal Filename : 'perfrt.dll'
      Original Filename : 'perfrt.dll'
Public Key:
Bits : 2048
Modulus:
        9c 49 00 24 65 e0 ee 9e a5 df 71 a7 7f fb 33 ac
        86 81 12 5d 3e 74 0b fa 75 c0 05 03 ad 54 82 87
        4e 41 a7 74 34 8d 5f ac 28 84 11 35 53 6d 8a 52
        86 12 1e b0 2d 19 71 21 73 fb b8 1b a8 e0 b2 7c
        58 e3 03 e6 72 3a a7 97 5e 24 c9 99 c2 11 db c7
        2f 50 66 38 2a 79 c3 b2 a8 6a 63 86 fe 98 f3 6d
        54 1d ac d1 a0 8f 23 7a f8 63 5d 20 d0 60 74 df
        7a b3 63 12 5e 72 02 be 63 26 fd 9b a1 ca 4e 86
        1f f2 24 e6 5d 3e ef fa b9 f3 c1 77 56 b1 c4 32
        17 83 5a 51 19 23 c0 2f b2 c0 db f6 f6 69 93 3d
        5a 49 5b 1a 27 b7 cb 86 c3 ee 14 c6 89 0b 97 dc
        36 10 fd 89 dc 09 6b dd 09 e6 f0 7d bd 4e e5 92
        b4 60 20 b0 fa b8 9c f4 df dc 56 7e b6 53 31 7f
        37 ec 66 15 84 33 fa f5 f6 fc 1f b7 0e d2 cb cf
        86 2a 0f 3b 30 31 70 34 36 6d 1f 7b 95 f8 14 8c
        ae 66 35 96 e7 ec 82 30 41 c5 22 53 ee c5 3a e1
Exponent:
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 01

D:\Resources\Pc\Tools\i386-winnt>
```

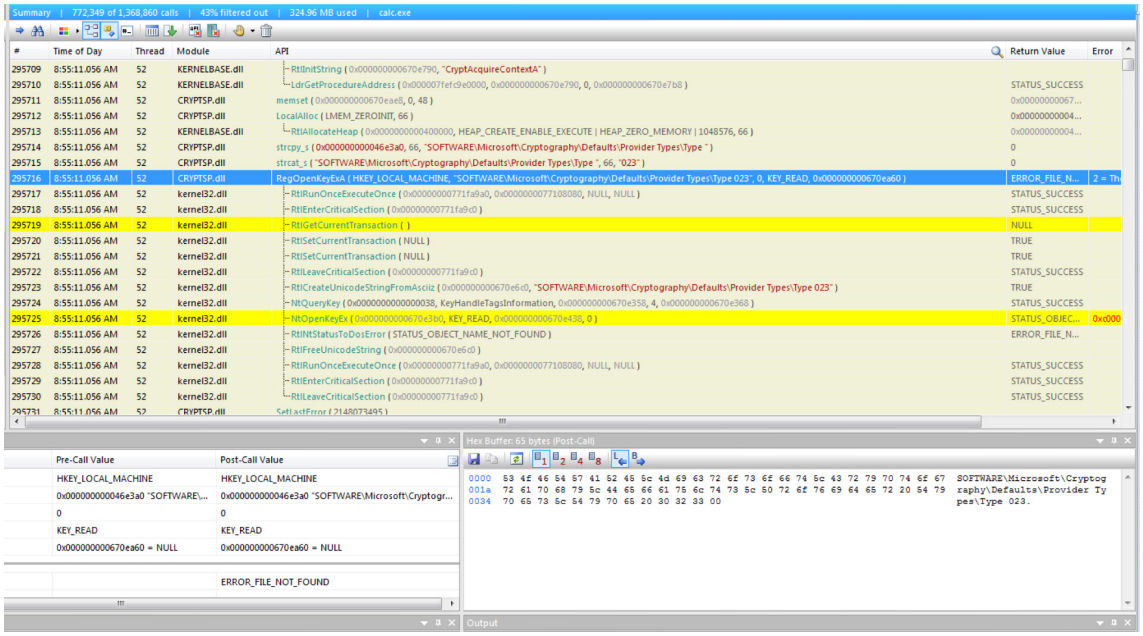
When trying to analyse the PC payload dynamically for operations relating to registry creating/edit/adding, the values appear not to be loaded into the stack of the running process but instead into the kernel in such a way which we were unable to recover. As such, due to these complications during analysis, we were unable to find the lists for the registry locations within the framework.

Refocus and results

As the list location eluded us we re-examined the installation process again for any further abnormal behaviour. Careful investigation of the operations called during installation did reveal an identifiable registry query that can be used to identify the process across hosts and is uniform. The operation queries a default cryptography provider type within the registry, a value which does not exist in typical systems. The value queried by the installation is "HKLM\Software\Microsoft\Cryptography\Defaults\Provider Types\Type 023" which, without additional action within the framework, results in "NAME NOT FOUND". This query operation was found in every experimental installation of a KiSu instance.

Cryptography provider types typically are the definition of cryptographic function with specific criteria so that an encryption algorithm such as RSA can be used in various ways for different effects. Although it is possible for a custom provider type to be defined, it is extremely unlikely that they will be stored as part of the Microsoft defaults within the HKLM hive. Research into a legitimate instance of the entry "Type 023" in that location generated no results.

Figure 20 –Observation of uniform Killsuiit installation activity, registry query for "Type 023" cryptography default provider type

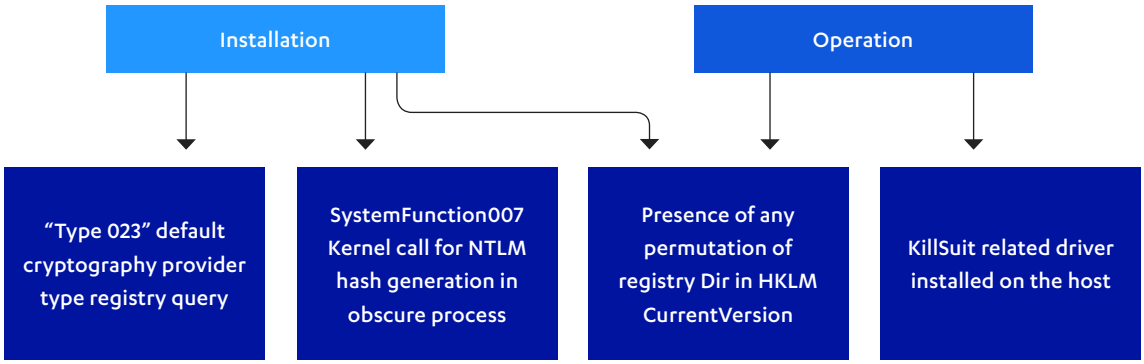


HOW TO DETECT AND REMEDIATE A KILLSUIT COMPROMISE

How to detect KillSuit installation

From our analysis of the installation and persistence mechanisms employed by KiSu there is a defined list of consistent identifiers for installation. In addition to these examples we also found an extensive list of drivers within the framework packages. This list seemed to consist of drivers known to be used for specific legitimate applications, known pieces of malware and frameworks components themselves including KiSu specific drivers. From the information available several of the drivers are labelled for removal, however one driver (mpdkg32) is not labelled for removal and should be present if any instances are installed. As such, presence of this driver or any of the other related drivers directly indicates installation on the host. A full list of the drivers associated with DdSz framework capabilities, including KiSu installation, can be found in the appendix.

Figure 21 –Detection methods for KillSuit stages



As such, the conclusion of our research and the specified driver list above indicates three possible methods of detecting installation of KiSu on a monitored host. First, the unusual use of the "systemfunction007" Kernel call followed almost immediately by registry write operations from an executable that is not meant to perform such actions. As the operator of DdSz can designate which running executable to reflect into this may be very difficult. By default the tool injects into lsass.exe, therefore the Kernel call for encryption generation is not unusual. Although an identifiable criteria, a clever operator will choose a process with such actions as standard to blend in.

Second, installation of any of the specified drivers related to KiSu in the list provided. Detection of the installation and removal of any KiSu drivers in the list is a clear indicator of the framework being used against a host. A scan across an estate for the presence of driver "mpdkg32" will be a very easy way to quickly sweep for legacy installation of the tool that may not be detectable during operation due to the lengths put in place to disguise the frameworks activity (custom TCP channel, full encryption etc.).

Finally, registry operations related to installation can be monitored for to detect live attacks or legacy installations. Sweeping/monitoring for registry keys under the HKLM matching any permutation of the two lists provided in the appendix may lead to the detection of KiSu installation or running operations. As this list is not certain to be conclusive this is only a partial measure but if paired with driver monitoring should create a high certainty of the presence of an instance on the host. However, monitoring for registry query operations against the cryptography default provider type "Type 023" is a very confident way to detect installation attempts on a host. If monitoring for such operations is available, the installation of a rule to monitor for that one registry key could provide clear evidence of malicious actions as part of a live monitoring system.

How to remove Killsuit from your host

Experimentation with remediation for KiSu showed that removal was most effective when the encrypted DB location for the module root had been identified. Removal of this registry location terminated all KiSu capabilities on a host including installed persistence through any of the mechanisms available. Removal of this directory instantly terminates any KiSu operations and removes the attacker's capability to persist.

As the KiSu persistence method relies on a special instance being loaded and then configured with the appropriate mechanism (typically post XP is SOTI), removal of the associated module root disables this instance and neuters the associated PC agent on reboot.

However, this remediation is only relevant for KiSu instance on a target machine and will not remove other persistent methods used by DdSz or other such frameworks for other payloads/ tools. Additionally, there are multiple methods to persist PC agents on a target machine and therefore this method will not guarantee remediation of a DdSz foothold.

CONCLUSION

Our research into KillSuit's indicators of compromise at installation yielded a variety of information, including active installation detection through the abnormal cryptographic provider type and a semi-conclusive legacy installation detection using the registry locations collected. Other identifiers found, although viable, are more difficult to verify and apply to live systems.

In addition to the indicators discovered, we also encountered a number of the methods put in place by the developers to prevent analysis and their development practices. This gave a unique insight into the level of effort and sophistication placed into this tool in order for it be successful for as long as possible. In fact it was this complexity that prevented us from easily retrieving the hard-coded registry installation values, although hopefully the analysis provided will give other researchers a solid starting point for further investigation.

The analysis presented in this report focused on the 2013 version of this tooling as such any indicators can be used to detect legacy Equation Group breaches or more recent breaches by groups reusing legacy tooling. However it is highly likely that the Equation Group themselves have redeveloped their tooling since the Shadowbrokers release and the indicators may no longer apply to current breaches. As always this emphasises the need for defensive teams to focus on continuous research, hunting and response to stay ahead of attackers.

APPENDIX

Killsuit instance ID list

- PC (PeddleCheap)
- UR (UnitedRake)
- STLA (StrangeLand)
- SNUN (SnuffleUnicorn)
- WRWA (WraithWrath)
- SLSH (SleepySheriff)
- WORA (WoozyRamble)
- TTSU (TiltTsunami)
- SOKN (SoberKnave)
- MAGR (MagicGrain)
- DODA (DoubleDare)
- SAAN (SavageAngel)
- MOAN (MorbidAngel)
- DEWH (DementiaWheel)
- CHMU (ChinMusic)
- MAMO (MagicMonkey)
- MABE (MagicBean)

Registry list – First part

- Account
- Acct
- Adapter
- App
- Correction
- Dir
- Directory
- Driv
- Locale
- Network
- Manufacturer
- Oem
- Plugin
- Power
- User
- Shutdown
- Wh

Registry list – Second Part

- Cache
- Cfg
- Config
- Database
- Db
- Exts
- Info
- Libs
- Logs
- Mappings
- Maps
- Mgmt
- Perf
- Settings
- Usage

Full Danderspritz driver list

- "1394ohci","*** SENTRYTRIBE MENTAL ***"
- "ac98intc","*** DARKSKYLINE MENTAL ***"
- "adpkprp","*** KILLSUIT LOADER DRIVER - REMOVE ME ***"
- "adpux86","*** DARKSKYLINE MENTAL ***"
- "agentcpd","*** DEMENTIAWHEEL ***"
- "agilevpn","*** ST MENTAL *** -OR- RAS Agile VPN Driver"
- "Agilevpn","*** ST MENTAL *** -OR- RAS Agile VPN Driver"
- "amdk5","*** DARKSKYLINE MENTAL ***"
- "appinit","*** PEDDLECHEAP ***"
- "ataport32","*** SENTRYTRIBE MENTAL ***"
- "atmdkdrv","*** UNITEDRAKE ***"
- "atpmmon","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "bifsgcom","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "bootvid32","*** SENTRYTRIBE MENTAL ***"
- "cewdaenv","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "clfs32","*** SENTRYTRIBE MENTAL ***"
- "cmib113u","*** STYLISHCHAMP/OLYMPUS ***"
- "cmib129u","*** SALVAGERABBIT/OLYMPUS ***"
- "dasmkit","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "dehihdp","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "devmgr32","*** SENTRYTRIBE MENTAL ***"
- "dlapaw","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "dlcndi","*** DOORWAYNAPKIN/STOWAGEWINK ***"
- "doccfg","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "dpti30","*** DARKSKYLINE MENTAL ***"

- "ds325gts","*** SALVAGERABBIT/OLYMPUS ***"
- "dump_msahci","*** MEM DUMP FOR DARKSKYLINE MENTAL ***"
- "dxg32","*** SENTRYTRIBE MENTAL ***"
- "dxghlp16","*** SENTRYTRIBE MENTAL ***"
- "DXGHLP16","*** SENTRYTRIBE MENTAL ***"
- "DXGHLP32","*** SENTRYTRIBE MENTAL ***"
- "ethip6","*** DOORMANGAUZE ***"
- "exFat","*** DARKSKYLINE MENTAL ***"
- "ext2fs32","*** SENTRYTRIBE MENTAL ***"
- "fast16","*** NOTHING TO SEE HERE - CARRY ON ***"
- "fastfat32","*** SENTRYTRIBE MENTAL ***"
- "FAT32","*** SENTRYTRIBE MENTAL ***"
- "Fdisk","*** OLYMPUS ***"
- "fld21","*** STORMTHUNDER ***"
- "FrzSys","*** Power Shadow / Shadow System ***"
- "FSPRTX","*** YAK 2 ***"
- "gdisdsk","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "Hproc","*** Angelltech Security Policy Management (SPM) ***"
- "hrlib","*** UNITEDRAKE ***"
- "hwinfo","*** *** InfoTeCS ViPNet *** ***"
- "inetcom32","*** LOCUSTTHREAT/UNITEDRAKE ***"
- "ip4fw","*** DARKSKYLINE MENTAL ***"
- "IPLIR","*** InfoTeCS ViPNet ***"
- "IPNPF","*** InfoTeCS ViPNet ***"
- "iqvwx86","*** DARKSKYLINE MENTAL ***"
- "irda32","*** DARKSKYLINE MENTAL ***"
- "irtidvc","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "itcscrpt","*** InfoTeCS ViPNet ***"
- "itcsids","*** InfoTeCS ViPNet ***"
- "itcspe","*** InfoTeCS ViPNet ***"
- "itcsprot","*** InfoTeCS ViPNet ***"
- "itcsrf","*** InfoTeCS ViPNet ***"
- "itcswd","*** InfoTeCS ViPNet ***"
- "jsdw776","*** MANTLESTUMP/UNITEDRAKE ***"
- "kbdclmgr","*** SALVAGERABBIT/UNITEDRAKE ***"
- "kbpnp","*** YAK ***"
- "khlp755w","*** STOWAGEWINK/UNITEDRAKE ***"
- "khlp807w","*** NETSPYDER ***"
- "khlp811u","*** SPINOFFCACTUS/OLYMPUS ***"
- "khlp894u","*** SCOUTRUMMAGE/UNITEDRAKE ***"
- "lhepfi","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "mdnwdiag","*** BEHAVEPEKING ***"
- "mf32","*** CARBONFIBER ***"
- "mipllst","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "mpdkg32","*** KILLSUIT ***"
- "mq32","*** CARBONFIBER ***"
- "msahci","*** DARKSKYLINE MENTAL ***"
- "mscnsp","*** FORMALRITE/UNITEDRAKE ***"
- "mscoreep","*** FOGGYBOTTOM/UNITEDRAKE ***"
- "msdtcs32","*** SPITTINGSPYDER/UNITEDRAKE ***"
- "mskbd","*** ELLIOTSPRINGE/FLEWAVENUE ***"
- "msmps32","*** HASSLEWITTPORT/UNITEDRAKE ***"
- "MSNDSRV","*** UNITEDRAKE 3.4 ***"
- "msndsrv","*** UNITEDRAKE 3.4 ***"
- "msrmdr32","*** STOWAGEWINK/UNITEDRAKE ***"

- "msrstd","*** SALVAGERABBIT ***"
- "msrtvd","*** GROK/UNITEDRAKE ***"
- "msrtvid32","*** SPINOFFCACUS/UNITEDRAKE ***"
- "msscd16","*** VALIDATOR ***"
- "mstcp32","*** SENTRYTRIBE ***"
- "mstkpr","*** SALVAGERABBIT/UNITEDRAKE ***"
- "msvcp56","*** PEDDLECHEAP ***"
- "msvcp57","*** PEDDLECHEAP ***"
- "msvcp58","*** PEDDLECHEAP ***"
- "ndis5mgr","*** FULLMOON ***"
- "nethdlr","*** MISTYVEAL ***"
- "netio","*** ST MENTAL *** -OR- NETIO Legacy TDI Support Driver"
- "NETIO","*** ST MENTAL *** -OR- NETIO Legacy TDI Support Driver"
- "netmst","*** SCOUTRUMMAGE/UNITEDRAKE ***"
- "nls_295w","*** SCOUTRUMMAGE/UNITEDRAKE ***"
- "nls_470u","*** SCOUTRUMMAGE/UNITEDRAKE ***"
- "nls_875u","*** SUPERFLEX/OLYMPUS ***"
- "nls_879u","*** SMOGSTRUCK/OLYMPUS ***"
- "nls_895u","*** SHADOWFLEX/OLYMPUS ***"
- "ntevt","*** FLEWAVENUE ***"
- "ntevt32","*** FLEWAVENUE (TEMP) ***"
- "nwlfi","*** DARKSKYLINE MENTAL ***"
- "olok_2k","*** KILLSUIT LOADER DRIVE - REMOVE ME ***"
- "oplemflt","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "otpemod","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "pdresy","*** DRAFTYPLAN ***"
- "perfnw","*** DOORMANGAUZE ***"
- "plugproc","*** SCOUTRUMMAGE/UNITEDRAKE ***"
- "pnpscsi","*** CARBONFIBER ***"
- "prsecmon","*** UTILITYBURST ***"
- "psecmon","*** UTILITYBURST ***"
- "pssdk31","*** microOLAP Packet Sniffer SDK Driver ***"
- "pssdk40","*** microOLAP Packet Sniffer SDK Driver ***"
- "pssdk41","*** microOLAP Packet Sniffer SDK Driver ***"
- "pssdk42","*** microOLAP Packet Sniffer SDK Driver ***"
- "pssdklbf","*** microOLAP Packet Sniffer SDK Driver ***"
- "psxssdll","*** PEDDLECHEAP ***"
- "rasapp","*** FOGGYBOTTOM/UNITEDRAKE ***"
- "rasl2tcp","*** DARKSKYLINE MENTAL ***"
- "rdpvrf","*** DOLDRUMWRAPUP ***"
- "risfclt","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "rls1201","*** FINALDUET/UNITEDRAKE ***"
- "ropdir","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "scsi2mgr","*** SALVAGERABBIT/OLYMPUS ***"
- "segfib","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "serstat","*** DRILLERSKYLINE ***"
- "shlgina","*** DEMENTIAWHEEL TASKING ***"
- "storvsc","*** DARKSKYLINE MENTAL ***"
- "symc81x","*** DARKSKYLINE MENTAL ***"
- "tapindis","*** JEALOUSFRUIT ***"
- "tcphoc","*** Thunder Networking BHO/Download Manager/Adware ***"
- "tdip","*** DARKSKYLINE ***"
- "tnesahs","*** KILLSUIT LAUNCHER DRIVER - REMOVE ME ***"
- "viac7","*** SENTRYTRIBE MENTAL ***"
- "vmm32","*** DARKSKYLINE MENTAL ***"

- "volrec","*** SALVAGERABBIT ***"
- "vregstr","*** VALIDATOR ***"
- "wanarpx86","*** DARKSKYLINE MENTAL ***"
- "wceusbsh32","*** SENTRYTRIBE MENTAL ***"
- "wdmaud32","*** SENTRYTRIBE MENTAL ***"
- "wimmount","*** SENTRYTRIBE MENTAL ***"
- "wmpvmux9","*** STOWAGEWINK/UNITEDRAKE ***"
- "wpl913h","*** GROK/UNITEDRAKE ***"
- "ws2ufsl","*** DARKSKYLINE MENTAL ***"
- "wship","*** PEDDLECHEAP 2.0 ***"
- "xpinet30","*** FOGGYBOTTOM ***"

SOURCES

https://www.cs.bu.edu/~goldbe/teaching/HW55815/presos/eqngroup.pdf

https://www.youtube.com/watch?v=R5mgAsd2VBM

https://github.com/misterch0c/shadowbroker/

http://www.irongeek.com/i.php?page=videos/derbycon8/track-3-17-killsuit-the-equation-groups-swiss-army-knife-for-persistence-evasion-and-data-exfil-francisco-donoso

Nobody has better visibility into real-life cyber attacks than F-Secure. We're closing the gap between detection and response, utilizing the unmatched threat intelligence of hundreds of our industry's best technical consultants, millions of devices running our award-winning software, and ceaseless innovations in artificial intelligence. Top banks, airlines, and enterprises trust our commitment to beating the world's most potent threats.

Together with our network of the top channel partners and over 200 service providers, we're on a mission to make sure everyone has the enterprise-grade cyber security we all need. Founded in 1988, F-Secure is listed on the NASDAQ OMX Helsinki Ltd.

f-secure.com | twitter.com/fsecure | linkedin.com/ f-secure

F-Secure