SANS DFIR

CHEAT SHEETS & NOTEBOOKS

SANS DFIR ANALYZING MALICIOUS DOCUMENTS

This cheat sheet outlines tips and tools for analyzing malicious documents, such as Microsoft Office, RTF, and PDF files.

General Approach to Document Analysis

1. Examine the document for anomalies, such as risky tags, scripts, and embedded artifacts.
2. Locate embedded code, such as shellcode, macros, JavaScript, or other suspicious objects.
3. Extract suspicious code or objects from the file.
4. If relevant, deobfuscate and examine macros, JavaScript, or other embedded code.
5. If relevant, emulate, disassemble and/or debug shellcode that you extracted from the document.
6. Understand the next steps in the infection chain.

Microsoft Office Format Notes

Binary Microsoft Office document files (.doc, .xls, etc.) use the OLE2 (a.k.a. Structured Storage) format.

SRP streams in OLE2 documents sometimes store a cached version of earlier VBA macro code.

OOXML document files (.docx, .xlsm, etc.) supported by Microsoft Office are compressed zip archives.

VBA macros in OOXML documents are stored inside an OLE2 binary file, which is within the zip archive.

Excel supports XLM macros that are embedded as formulas in sheets without the OLE2 binary file.

RTF documents don't support macros but can contain malicious embedded files and objects.

Useful MS Office File Analysis Commands

zipdump.py ***file.pptx***	Examine contents of OOXML file *file.pptx*.
zipdump.py ***file.pptx -s 3 -d***	Extract file with index *3* from *file.pptx* to STDOUT.
olevba.py ***file.xlsm***	Locate and extract macros from *file.xlsm*.
oledump.py ***file.xls -s 3 -v***	Extract VBA source code from stream *3* in *file.xls*.
xmldump.py pretty	Format XML file supplied via STDIN for easier analysis.
oledump.py ***file.xls*** **-p plugin_http_heuristics**	Find obfuscated URLs in *file.xls* macros.
vmonkey ***file.doc***	Emulate the execution of macros in file.doc to analyze them.
evilclippy -uu ***file.ppt***	Remove the password prompt from macros in *file.ppt*.
msoffcrypto-tool ***infile.docm outfile.docm -p***	Decrypt outfile.docm using specified password to create *outfile.docm*.
pcodedmp ***file.doc***	Disassemble VBA-stomped p-code macro from *file.doc*.
pcode2code ***file.doc***	Decompile VBA-stomped p-code macro from *file.doc*.
rtfobj.py ***file.rtf***	Extract objects embedded into RTF *file.rtf*.
rtfdump.py ***file.rtf***	List groups and structure of RTF file *file.rtf*.
rtfdump.py ***file.rtf -O***	Examine objects in RTF file *file.rtf*.
rtfdump.py ***file.rtf*** **file.doc**	Extract hex contents from group in RTF file *file.rtf*.
xlmdeobfuscator --file ***file.xlsm***	Deobfuscate XLM (Excel 4) macros in *file.xlsm*.

Risky PDF Keywords

/OpenAction and /AA specify the script or action to run automatically.

/JavaScript, /JS, /AcroForm, and /XFA can specify JavaScript to run.

/URI accesses a URL, perhaps for phishing.

/SubmitForm and /GoToR can send data to URL.

/ObjStm can hide objects inside an object stream.

/XObject can embed an image for phishing.

Be mindful of obfuscation with hex codes, such as /JavaScript vs. /J#61vaScript. (**See examples**)

Useful PDF File Analysis Commands

pdfid.py ***file.pdf*** **-n**	Display risky keywords present in file *file.pdf*.
pdf-parser.py ***file.pdf*** **-a**	Show stats about keywords. Add "**-O**" to include object streams.
pdf-parser.py ***file.pdf -o id***	Display contents of object id. Add "**-d**" to dump object's stream.
pdf-parser.py ***file.pdf -r id***	Display objects that reference object **id**.
qpdf --password=pass --decrypt infile.pdf outfile.pdf	Decrypt *infile.pdf* using password pass to create *outfile.pdf*.

Shellcode and Other Analysis Commands

xorsearch -W -d 3 file.bin	Locate shellcode patterns inside the binary file *file.bin*.
scdbg /f ***file.bin***	Emulate execution of shellcode in *file.bin*. Use "**/off**" to specify offset.
runsc32 -f ***file.bin*** **-n**	Execute shellcode in file *file.bin* to observe behavior in an isolated lab.
base64dump.py ***file.txt***	List Base64-encoded strings present in file *file.txt*.
numbers-to-string.py file	Convert numbers that represent characters in *file* to a string.

Additional Document Analysis Tools

SpiderMonkey, cscript, and box-js help deobfuscate JavaScript that you extract from document files.

Use the debugger built into Microsoft Office to deobfuscate macros in an isolated lab.

Use AMSIScriptContentRetrieval.ps1 to observe Microsoft Office execute macros in an isolated lab.

Some automated analysis sandboxes can analyze aspects of malicious document files.

REMnux distro includes many of the free document analysis tools mentioned above.

Authored by Lenny Zeltser (**zeltser.com**) with feedback from **Pedro Bueno** and **Didier Stevens**. Malicious document analysis and related topics are covered in the SANS Institute course **FOR610: Reverse-Engineering Malware**, which they've co-authored.

SANS DFIR REMnux Usage Tips for Malware Analysis on Linux

This cheat sheet outlines the tools and commands for analyzing malicious software on REMnux Linux distro.

Getting Started with REMnux

1. Get REMnux as a virtual appliance, install the distro on a dedicated system, or add it to an existing one.
2. Review REMnux documentation at docs.remnux.org.
3. Keep your system up to date by periodically running "remnux upgrade" and "remnux update".
4. Become familiar with REMnux malware analysis tools available as Docker images.
5. Know default logon credentials: remnux/malware

General Commands on REMnux

Shut down the system ..... `shutdown`

Reboot the system ..... `reboot`

Switch to a root shell ..... `sudo -s`

Renew DHCP lease ..... `renew-dhcp`

See current IP address..... `myip`

Edit a text file..... `code file`

View an image file..... `feh file`

Start web server ..... `httpd start`

Start SSH server..... `sshd start`

Analyze Windows Executables

Static Properties: manalyze, peframe, pefile, exiftool, clamscan, pescan, portex, bearcommander, pecheck

Strings and Deobfuscation: pestr, bbcrack, brxor.py, base64dump, xorsearch, flarestrings, floss, cyberchef

Code Emulation: binee, capa, vivbin

Disassemble/Decompile: ghidra, cutter, objdump, r2

Unpacking: bytehist, de4dot, upx

Reverse-Engineer Linux Binaries

Static Properties: trid, exiftool, pyew, readelf.py

Disassemble/Decompile: ghidra, cutter, objdump, r2

Debugging: edb, gdb

Behavior Analysis: ltrace, strace, frida, sysdig, unhide

Investigate Other Forms of Malicious Code

Android: apktool, droidlysis, androgui.py, baksmali, dex2jar

Java: cfr, procyon, jad, jd-gui, idx_parser.py

Python: pyinstxtractor.py, pycdc

JavaScript: js, js-file, objects.js, box-js

Shellcode: shellcode2exe.bat, scdbg, xorsearch

PowerShell: pwsh, base64dump

Flash: swfdump, flare, flasm, swf_mastah.py, xxxswf

Examine Suspicious Documents

Microsoft Office Files: vmonkey, pcodedmp, olevba, xlmdeobfuscator, oledump.py, msoffice-crypt, ssview

RTF Files: rtfobj, rtfdump

Email Messages: emldump, msgconvert

PDF Files: pdfid, pdfparser, pdfextract, pdfdecrypt, peepdf, pdftk, pdfresurrect, qpdf, pdfobjflow

General: base64dump, tesseract, exiftool

Explore Network Interactions

Monitoring: burpsuite, networkminer, polarproxy, mitmproxy, wireshark, tshark, ngrep, tcpxtract

Connecting: thug, nc, tor, wget, curl, irc, ssh, unfurl

Services: fakedns, fakemail, accept-all-ips, nc, httpd, inetsim, fakenet, sshd, myip

Gather and Analyze Data

Network: Automater.py, shodan, ipwhois_cli.py, pdnstool

Hashes: malwoverview.py, nsrllookup, Automater.py, vt, virustotal-search.py

Files: yara, scalpel, bulk_extractor, ioc_writer

Other: dexray, viper, time-decode.py

Other Analysis Tasks

Memory Forensics: vol.py, vol3, linux_mem_diff.py, aeskeyfind, rsakeyfind, bulk_extractor

File Editing: wxHexEditor, scite, code, xpdf, convert

File Extraction: 7z, unzip, unrar, cabextract

Use Docker Containers for Analysis

Thug Honeyclient: remnux/thug

JSDetox JavaScript Analysis: remnux/jsdetox

Rekall Memory Forensics: remnux/recall

RetDec Decompiler: remnux/retdec

Radare2 Reversing Framework: remnux/radare2

Ciphey Automatic Decrypter: remnux/ciphey

Viper Binary Analysis Framework: remnux/viper

REMnux in a Container: remnux/remnux-distro

Interact with Docker Images

List local images ..... `docker images`

Update local image..... `docker pull image`

Delete local image..... `docker rmi imageid`

Delete unused resources..... `docker system prune`

Open a shell inside a transient container ..... `docker run --rm -it image bash`

Map a local TCP port 80 to container's port 80 ... `docker run --rm -it -p 80:80 image bash`

Map your current directory into container ... `docker run --rm -it -v .:dir image bash`

Authored by Lenny Zeltser for REMnux v7. Lenny writes a security blog at **zeltser.com** and is active on Twitter at @lennyzeltser. Many REMnux tools and techniques are discussed in the Reverse-Engineering Malware course at SANS Institute, which Lenny co-authored.

SANS DFIR Tips for Reverse-Engineering Malicious Code

Cheat sheet for reversing malicious Windows executables via static and dynamic code analysis.

Overview of the Code Analysis Process

1. Examine static properties of the Windows executable for initial assessment and triage.
2. Identify strings and API calls that highlight the program's suspicious or malicious capabilities.
3. Perform automated and manual behavioral analysis to gather additional details.
4. Emulate code execution to identify characteristics and areas for further analysis.
5. Use a disassembler and decompiler to statically examine code related to risky strings and APIs.
6. Use a debugger for dynamic analysis to examine how risky strings and API calls are used.
7. If appropriate, unpack the code and its artifacts.
8. As your understanding of the code increases, add comments, labels; rename functions, variables.
9. Progress to examine the code that references or depends upon the code you've already analyzed.
10. Repeat steps 5–9 above as necessary (the order may vary) until analysis objectives are met.

Common 32-Bit Registers and Uses

EAX	Addition, multiplication, function results
ECX	Counter; used by LOOP and others
EBP	Baseline/frame pointer for referencing function arguments (EBP+value) and local variables (EBP-value)
ESP	Points to the current "top" of the stack; changes via PUSH, POP, and others
EIP	Instruction pointer; points to the next instruction; shellcode gets it via call/pop
EFLAGS	Contains flags that store outcomes of computations (e.g., Zero and Carry flags)
FS	F segment register; FS[0] points to SEH chain, FS[0x30] points to the PEB.

Common x86 Assembly Instructions

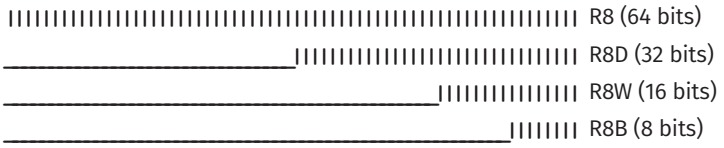
`mov EAX,0xB8`	Put the value 0xB8 in EAX.
`push EAX`	Put EAX contents on the stack.
`pop EAX`	Remove contents from top of the stack and put them in EAX .
`lea EAX, [EBP-4]`	Put the address of variable EBP-4 in EAX.
`call EAX`	Call the function whose address resides in the EAX register.
`add esp,8`	Increase ESP by 8 to shrink the stack by two 4-byte arguments.
`sub esp,0x54`	Shift ESP by 0x54 to make room on the stack for local variable(s).
`xor EAX,EAX`	Set EAX contents to zero.
`test EAX,EAX`	Check whether EAX contains zero, set the appropriate EFLAGS bits.
`cmp EAX,0xB8`	Compare EAX to 0xB8, set the appropriate EFLAGS bits.

Understanding 64-Bit Registers

EAX→RAX, ECX→RCX, EBX→RBX, ESP→RSP, EIP→RIP

Additional 64-bit registers are R8-R15.

RSP is often used to access stack arguments and local variables, instead of EBP.



Passing Parameters to Functions

arg0	[EBP+8] on 32-bit, RCX on 64-bit
arg1	[EBP+0xC] on 32-bit, RDX on 64-bit
arg2	[EBP+0x10] on 32-bit, R8 on 64-bit
arg3	[EBP+0x14] on 32-bit, R9 on 64-bit

Decoding Conditional Jumps

`JA / JG`	Jump if above/jump if greater.
`JB / JL`	Jump if below/jump if less.
`JE / JZ`	Jump if equal; same as jump if zero.
`JNE / JNZ`	Jump if not equal; same as jump if not zero.
`JGE / JNL`	Jump if greater or equal; same as jump if not less.

Some Risky Windows API Calls

Code injection: CreateRemoteThread, OpenProcess, VirtualAllocEx, WriteProcessMemory, EnumProcesses

Dynamic DLL loading: LoadLibrary, GetProcAddress

Memory scraping: CreateToolhelp32Snapshot, OpenProcess, ReadProcessMemory, EnumProcesses

Data stealing: GetClipboardData, GetWindowText

Keylogging: GetAsyncKeyState, SetWindowsHookEx

Embedded resources: FindResource, LockResource

Unpacking/self-injection: VirtualAlloc, VirtualProtect

Query artifacts: CreateMutex, CreateFile, FindWindow, GetModuleHandle, RegOpenKeyEx

Execute a program: WinExec, ShellExecute, CreateProcess

Web interactions: InternetOpen, HttpOpenRequest, HttpSendRequest, InternetReadFile

Additional Code Analysis Tips

Be patient but persistent; focus on small, manageable code areas and expand from there.

Use dynamic code analysis (debugging) for code that's too difficult to understand statically.

Look at jumps and calls to assess how the specimen flows from "interesting" code block to the other.

If code analysis is taking too long, consider whether behavioral or memory analysis will achieve the goals.

When looking for API calls, know the official API names and the associated native APIs (Nt, Zw, Rtl).

Authored by Lenny Zeltser (**zeltser.com**) with feedback from **Anuj Soni**. Malicious code analysis and related topics are covered in the SANS Institute course **FOR610: Reverse-Engineering Malware**, which they've co-authored. For additional reversing, security and IT tips, visit **zeltser.com/cheat-sheets**.

SANS DFIR Malware Analysis & Reverse Engineering Cheat Sheet

The analysis and reversing tips behind this reference are covered in the SANS Institute course FOR610: Reverse-Engineering Malware.

Overview of the Malware Analysis Process

1. Use automated analysis sandbox tools for an initial assessment of the suspicious file.
2. Set up a controlled, isolated laboratory in which to examine the malware specimen.
3. Examine static properties and meta-data of the specimen for triage and early theories.
4. Emulate code execution to identify malicious capabilities and contemplate next steps.
5. Perform behavioral analysis to examine the specimen's interactions with its environment.
6. Analyze relevant aspects of the code statically with a disassembler and decompiler.
7. Perform dynamic code analysis to understand the more difficult aspects of the code.
8. If necessary, unpack the specimen.
9. Repeat steps 4-8 above as necessary (the order may vary) until analysis objectives are met.
10. Augment your analysis using other methods, such as memory forensics and threat intel.
11. Document findings, save analysis artifacts and clean up the laboratory for future analysis.

Behavioral Analysis

Be ready to revert to good state via virtualization snapshots, Clonezilla, dd, FOG, PXE booting, etc.

Monitor local interactions (Process Explorer, Process Monitor, ProcDOT, Noriben).

Detect major local changes (RegShot, Autoruns).

Monitor network interactions (Wireshark, Fiddler).

Redirect network traffic (fakedns, accept-all-ips).

Activate services (INetSim or actual services) requested by malware and reinfect the system.

Adjust the runtime environment for the specimen as it requests additional local or network resources.

Ghidra for Static Code Analysis

Go to specific destination	**g**
Show references to instruction	**Ctrl+Shift+f**
Insert a comment	**;**
Follow jump or call	**Enter**
Return to previous location	**Alt+Left**
Go to next location	**Alt+Right**
Undo	**Ctrl+z**
Define data type	**t**
Add a bookmark	**Ctrl+d**
Text search	**Ctrl+Shift+e**
Add or edit a label	**l**
Disassemble values	**d**

x64dbg/x32dbg for Dynamic Code Analysis

Run the code	**F9**
Step into/over instruction	**F7/F8**
Execute until selected instruction	**F4**
Execute until the next return	**Ctrl+F9**
Show previous/next executed instruction	**-/+**
Return to previous view	*****
Go to specific expression	**Ctrl+g**
Insert comment/label	**;/:**
Show current function as a graph	**g**
Find specific pattern	**Ctrl+b**
Set software breakpoint on specific instruction	**Select instruction » F2**
Set software breakpoint on API	**Go to Command prompt » SetBPX *API Name***
Highlight all occurrences of the keyword in disassembler	**h » Click on keyword**
Assemble instruction in place of selected one	**Select instruction » Spacebar**
Edit data in memory or instruction opcode	**Select data or instruction » Ctrl+e**
Extract API call references	**Right-click in disassembler » Search for » Current module » Intermodular calls**

Unpacking Malicious Code

Determine whether the specimen is packed by using Detect It Easy, Exeinfo PE, Bytehist, peframe, etc.

To try unpacking the specimen quickly, infect the lab system and dump from memory using Scylla.

For more precision, find the Original Entry Point (OEP) in a debugger and dump with OllyDumpEx.

To find the OEP, anticipate the condition close to the end of the unpacker and set the breakpoint.

Try setting a memory breakpoint on the stack in the unpacker's beginning to catch it during cleanup.

To get closer to the OEP, set breakpoints on APIs such as LoadLibrary, VirtualAlloc, etc.

To intercept process injection set breakpoints on VirtualAllocEx, WriteProcessMemory, etc.

If cannot dump cleanly, examine the packed specimen via dynamic code analysis while it runs.

Rebuild imports and other aspects of the dumped file using Scylla, Imports Fixer, and pe_unmapper.

Bypassing Other Analysis Defenses

Decode obfuscated strings statically using FLOSS, xorsearch, Balbuzard, etc.

Decode data in a debugger by setting a breakpoint after the decoding function and examining results.

Conceal x64dbg/x32dbg via the ScyllaHide plugin.

To disable anti-analysis functionality, locate and patch the defensive code using a debugger.

Look out for tricky jumps via TLS, SEH, RET, CALL, etc. when stepping through the code in a debugger.

If analyzing shellcode, use scdbg and runsc.

Disable ASLR via setdllcharacteristics, CFF Explorer.

Authored by Lenny Zeltser, who is the CISO at **Axonius** and Faculty Fellow at **SANS Institute**. You can find him at **twitter.com/lennyzeltser** and **zeltser.com**. Download this and other Lenny's security cheat sheets from **zeltser.com/cheat-sheets**.

SANS DFIR Apple File System (APFS) Reference Sheet

This guide was created by by Sarah Edwards | T: @iamevltwin | E: oompa@csh.rit.edu

Object Header (`obj_phys_t`)

Offset	Size (in bytes)	Field	Notes
0	8	`o_cksum`	Fletcher 64 Checksum
8	8	`o_oid`	Object ID
16	8	`o_xid`	Transaction ID
24	2	`o_type.type`	Object Type
26	2	`o_type.flags`	Object Flags
28	4	`o_subtype`	Object Subtype

Object Type (Hex)	Object Type (Dec)	Object Type/Subtype
0x0000	0	None
0x0100	1	Container Super Block
0x0200	2	B-Tree
0x0300	3	B-Tree Node
0x0500	5	Spaceman
0x0B00	11	Object Map (OMAP)
0x0D00	13	File System (Volume Super Block)
0x0E00	14	File System Tree

Container Super Block (`nx_superblock_t`)

Offset	Size (in bytes)	Field	Notes
32	4	`magic "NXSB"`	Container Magic Number: 0x4E585342 = “NXSB”
36	4	`nx_block_size`	Block Size (ie: 4096)
40	8	`nx_block_count`	Block Count (Block Count*Block Size = Container Size in Bytes
48	8	`nx_features`	Features
56	8	`nx_read_only_compatible_features`	Read-only Compatible Features
64	8	`nx_incompatable_features`	Incompatible Features
72	16	`nx_uuid`	Container UUID (diskutil info /dev/disk#)
88	8	`nx_next_oid`	Next Object ID (OID)
96	8	`nx_next_xid`	Next Transaction ID (XID)
104	4	`nx_xp_desc_blocks`	Blocks used by Checkpoint Descriptor Area
108	4	`nx_xp_data_blocks`	Blocks used by Checkpoint Data Area
112	8	`nx_xp_desc_base`	Base address of Checkpoint Descriptor Area or Physical Object ID
120	8	`nx_xp_data_base`	Base address of Checkpoint Data Area or Physical Object ID
128	4	`nx_xp_desc_next`	Next Index for Checkpoint Descriptor Area
132	4	`nx_xp_data_next`	Next Index for Checkpoint Data Area
136	4	`nx_xp_desc_index`	Index for first item in Checkpoint Descriptor Area
140	4	`nx_xp_desc_len`	Number of blocks in Checkpoint Descriptor Area Used
144	4	`nx_xp_data_index`	Index for first item in Checkpoint Data Area
148	4	`nx_xp_data_len`	Number of blocks in Checkpoint Data Area Used
152	8	`nx_spaceman_oid`	Space Manager Object ID (OID)
160	8	`nx_omap_oid`	Container Object Map Object ID (OID)
168	8	`nx_reaper_oid`	Reaper Object ID (OID)
176	4	`nx_test_type`	Reserved for Testing
180	4	`nx_max_file_systems`	Maximum Number of Volumes in this Container
184	8	`nx_fs_oid[0]`	Array of OIDs for Volumes in this Container

APFS Format References:

- Apple File System Reference (Apple Developer Documentation)
- 2019-02-07

APFS is Little Endian & 64-bit

Updated: 01/2020

Volume Super Block (`apfs_superblock_t`)

Offset	Size (in bytes)	Field	Notes
32	4	`apfs_magic "APSB"`	Volume Magic Number 0x41505342 = “APSB”
36	4	`apfs_fs_index`	Index in Volume Array
40	8	`apfs_features`	Features
48	8	`apfs_readonly_compatible_features`	Read-only Incompatible Features
56	8	`apfs_incompatible_features`	Incompatible Features
64	8	`apfs_unmount_time`	Timestamp when volume was last unmounted
72	8	`apfs_fs_reserve_block_count`	Block Pre-allocated for Volume (Default is none)
80	8	`apfs_fs_quota_block_count`	Maximum Block Allocated (Default is none)
88	8	`apfs_fs_alloc_count`	Number of blocks currently allocated
96	2	`wrapped_crypto_state_t. wrapped_crypto_state.major_version`	Key Encryption Metadata – Major Version
98	2	`wrapped_crypto_state_t. wrapped_crypto_state.minor_version`	Key Encryption Metadata – Minor Version
100	4	`wrapped_crypto_state_t. wrapped_crypto_state.cpflags`	Key Encryption Metadata – Encryption State Flags
104	4	`wrapped_crypto_state_t. wrapped_crypto_state.persistent_class`	Key Encryption Metadata – Protection Class
108	4	`wrapped_crypto_state_t. wrapped_crypto_state.key_os_version`	Key Encryption Metadata – Creator OS Version 0x39004313 = 19 C 57 – 19C57 – Catalina 10.15.2
112	2	`wrapped_crypto_state_t. wrapped_crypto_state.key_revision`	Key Encryption Metadata – Key Version
114	2	`wrapped_crypto_state_t. wrapped_crypto_state.key_len`	Key Encryption Metadata – Key Size (0 for no Encryption)
N/A	0	`wrapped_crypto_state_t. wrapped_crypto_state.persistent_key`	Key Encryption Metadata – Wrapped Key No Key field is null, see key_len above
116	4	`apfs_root_tree_oid_type`	Type of Root File System Tree = B-Tree
120	4	`apfs_extentref_tree_oid_type`	Type of Extent Reference Tree = B-Tree, Physical
124	4	`apfs_snap_meta_tree_oid_type`	Type of Snapshot Metadata Tree = B-Tree, Physical
128	8	`apfs_omap_oid`	Physical Object ID (OID) of Object Map
136	8	`apfs_root_tree_oid`	Virtual Object ID (OID) of Root File System Tree
144	8	`apfs_extentref_tree_oid`	Physical Object ID (OID) of Extent Reference Tree
152	8	`apfs_snap_meta_tree_oid`	Virtual Object ID (OID) of Snapshot Metadata Tree
160	8	`apfs_revert_to_xid`	Transaction ID (XID) that volume will revert to
168	8	`apfs_revert_to_sblock_oid`	Virtual Object ID (OID) of Volume Superblock to revert to
176	8	`apfs_next_obj_id`	Next Object ID (OID)
184	8	`apfs_num_files`	Number of Regular Files
192	8	`apfs_num_directories`	Number of Directories
200	8	`apfs_num_symlinks`	Number of Symbolic Links
208	8	`apfs_num_other_fsobjects`	Number of Other Files
216	8	`apfs_num_snapshots`	Number of Snapshots
224	8	`apfs_total_blocks_alloced`	Blocks Allocated by Volume
232	8	`apfs_total_blocks_freed`	Blocked Freed by Volume
240	16	`apfs_vol_uuid`	Volume UUID (diskutil info /dev/disk#s# [Volume])
256	8	`apfs_last_mod_time`	Last Modified Timestamp
264	8	`apfs_fs_flags`	Flags
272	32	`apfs_modified_by_t.formatted_by.id[]`	Format Program and Version
304	8	`apfs_modified_by_t.formatted_by. timestamp`	Format Timestamp
312	8	`apfs_modified_by_t.formatted_by. last_xid`	Format Transaction ID (XID)
320	32	`apfs_modified_by_t.modified_by.id[]`	Last Modified Program and Version
352	8	`apfs_modified_by_t.modified_by. timestamp`	Last Modified Timestamp
360	8	`apfs_modified_by_t.modified_by. last_xid`	Last Modified Transaction ID (XID)
368	336	`apfs_modified_by_t.modified_by[1-7]`	Array of apfs_modified_by_t[8]
704	256	`apfs_volname`	APFS Volume Name
960	4	`apfs_next_doc_id`	Next Document ID
964	2	`apfs_role`	APFS Role (None, System, Data, Preboot, VM, Recovery)
966	2	`apfs_reserved`	Reserved
976	8	`apfs_root_to_xid`	Transaction ID (XID) of Snapshot to Root
984	8	`apfs_er_state_oid`	Current State of Encryption/Decryption

B-Tree Node (`btree_node_phys_t`)

Offset	Size (in bytes)	Field	Notes
32	2	`btn_flags`	Flags (Leaf Node)
34	2	`btn_level`	Number of Child Levels below this Node
36	4	`btn_nkeys`	Number of Keys
40	2	`btn_table_space.off`	Offset to Table of Contents (after btree_node_phys_t)
42	2	`btn_table_space.len`	Length of Table of Contents
44	2	`btn_freespace.off`	Offset Key/Value Free Space
46	2	`btn_freespace.len`	Length of Key/Value Free Space
48	2	`btn_key_free_list.off`	Offset to Free Key Space
50	2	`btn_key_free_list.len`	Length of Free Key Space
52	2	`btn_val_free_list.off`	Offset to Free Value Space
54	2	`btn_val_free_list.len`	Length of Free Value Space

B-Tree Node – Table of Contents

Offset	Size (in bytes)	Field	Notes
TOC Entry + 2	2	`key_offset`	Key Offset
TOC Entry + 4	2	`key_length`	Key Length
TOC Entry + 6	2	`value _offset`	Value Offset
TOC Entry + 8	2	`value_length`	Value Length

B-Tree Node – File System Key

Offset	Size (in bytes)	Object ID – Inode Number
7	1	Entry Kind
		0x30 – Inode
		0x60 – Data Stream
		0x40 – Xattr (2 byte Name Length + Variable Xattr Name)
		0x60 – File Extent (8 byte Logical Address)

Value – Inode File Metadata

Offset	Size (in bytes)	Field	Notes
0	8	`parent_id`	Parent Inode Number
8	8	`private_id`	Inode Number
16	8	`create_time`	Create Timestamp
24	8	`mod_time`	Modification Timestamp
32	8	`change_time`	Change Timestamp
40	8	`access_time`	Access Timestamp
48	8	`internal_flags`	Internal Flags
56	4	`nchildren or nlink`	Children or Links
60	4	`default_protection_class`	Default Protection Class
64	4	`write_generation_counter`	Write Generation Counter
68	4	`bsd_flags`	BSD Flags
72	4	`owner`	Owner
76	4	`group`	Group
80	2	`mode`	File Mode
82	2	`pad1`	Pad1
84	8	`pad2`	Pad2
92	2	`xf_num_exts`	Number of Extended Fields
94	2	`xf_used_data`	Extended Fields Data Used
96	x_field_t[] = 4 bytes Each	Extended Field: `x_type` (1 byte), `x_flags` (1 byte), `x_size` (2 bytes)	
96	4	EXAMPLE EXTENDED FIELD: `0x04 = 4`, `0x02` (Do Not Copy), `0x1100 = 17` (File Name)	
100	4	EXAMPLE EXTENDED FIELD: `0x08 = 8, 0x20` (System Field), `0x2800 = 40` (Data Stream)	
104	{17}	File Name	smudge_yoda.jpeg (w/1 padding bytes 0x00), 17 total bytes
120	{40}	Data Stream (Size: First 8 bytes, Allocated: Next 8 bytes)	0x00000000000000 – 7 unused bytes Size: 0x261C020000000000 = 138278 bytes Allocated: 0x0020020000000000 = 139264

Value – Inode File Extent

Offset	Size (in bytes)	Field
0	8	File Size
8	8	Physical Block Location
16	8	Crypto ID

Directory Commands

`cd ..`	Change Directory...up one directory (`../..` – two directories up)
`cd /var/log`	Change Directory...to `/var/log`
`cd ~`	Change Directory...to your home directory
`cd /`	Change Directory...to the root directory
`ls`	List Directory (Short Listing)
`ls -l`	List Directory (Long Listing)
`ls -a`	List Directory items...including hidden items (files beginning with ".")
`ls -lh`	List Directory items...with human readable sizes
`ls -R`	List Directory items...recursively
`open .`	Open Current Directory
`pwd`	Print Working Directory
`mkdir`	Create a Directory
`rmdir`	Remove a Directory
`rm -r`	Remove a Directory (and its contents)
`.`	Current Directory
`..`	Parent Directory

File Commands

`pico <filename>`	Open a file in a simple text editor (`q`– to quit editor)
`xxd <filename>`	Open a file in a hex editor
`open <filename>`	Opens a file in the default program
`open –a <programname> <filename>`	Opens a file in a specified program
`cat <filename>`	Concatenate a file to the terminal screen
`<command> | more`	Pipe command output to more to show contents screen by screen
`<command> | less`	Pipe command output to less to show contents screen by screen (and be able to go back and forth)
`rm <filename>`	Remove File
`cp <filename> <newfilename>`	Copy File
`mv <filename> <newfilename>`	Move File
`<command> > <filename>`	Redirect command output to a file
`<command> >> <filename>`	Append command output to a file
`touch <filename>`	Create an empty file
`head <filename>`	Show first 10 lines of a file
`tail <filename>`	Show last 10 lines of a file (`-f` to watch appended input)
`strings <filename>`	Show the strings of a file
`exiftool <filename>`	Show the exif/metadata of the file
`plutil –p <propertylist>`	Print the contents of a property list
`file <filename>`	Show a file signature type
`grep –i <searchterm> <filename>`	Search for term within a file (case-insensitive)
`python <file>.py`	Execute a Python program

Miscellaneous Commands

`sudo <command>`	Execute program as another user (default is root user)
`sudo -s`	Open a privileged shell
`su -`	Substitute User to root
`whoami / id`	Display Effective User ID / Show UID/GID Info
`history`	Command History
`man <command>`	Command Manual (`q`– to exit manual)

Terminal Shortcuts

Control + A	Jump to beginning of line
Control + E	Jump to end of of line
Tab	Tab Completion
Control + C	Kill Current Command
Command + K or Control + L	Clear Screen (or clear command)
Command + T	New Terminal Tab
Command + W	Close Terminal Tab
Command +/-	Increase or Decrease Terminal Font Size
Option + Left/Right Arrow	Move back/forth by word
Option + Click in Command Line	Put command line cursor where mouse cursor is

Generic Tool Compilation and Installation

```
tar –xvf <archive>.tar.gz
./configure
make
sudo make install
```

Live Response

`date`	Local System Time (`-u` for UTC)
`hostname`	System Hostname
`uname -a`	OS & Architecture Information
`sw_vers`	macOS Version & Build
`netstat -anf inet or netstat -an`	Active Network Connections
`lsof -i`	Active Network Connections (by process)
`netstat -rn`	Routing Table
`arp -an | ndp -an`	ARP Table (IPv4 | IPv6)
`ifconfig`	Network Interface Configuration
`lsof`	List Open Files
`who -a, w`	List Logged On Users
`last`	List user logins
`ps aux`	List Processes
`system_profiler -xml -detaillevel full > file.spx`	System Profiler (XML, Full Detail Level), open with System Information.app

Disk & Partitions

`/dev/`	Device Directory
`diskutil list`	List Connected Disks
`diskutil info <disk>`	Disk Information (use Disks `/dev/disk#`, `disk#`, or partitions `/dev/disk#s#`)
`diskutil cs|ap list`	List partitions using CoreStorage (cs) or APFS Containers (ap)
`gpt -r show [-l]`	List partitions using GUID Partition Table Format (`-l` to show label rather than GUID) – 10.13+ SIP must be disabled.
`csrutil disable|enable`	Disable/Enable SIP, must reboot into Recovery Mode (Reboot, Cmd+Option+R)
`mmls <diskimage>`	Display partitions using The Sleuth Kit
`hdiutil imageinfo *.dmg`	Disk Image Information including Partition Data

Keychains

`security list-keychains`	List Keychains on a system for a logged in user
`security dump-keychains -d <keychain>`	Dump contents of a Keychain

Extended Attributes

`xattr -xl <file>`	Show Extended Attributes of a file
`xattr -p <attribute name> <file> | xxd -r -p >output_file.plist`	Extract embedded binary property list from extended attribute
`istat /dev/disk# <CNID>`	Use The Sleuth Kit to view file information including extended attributes
`icat /dev/disk# <CNID>-<TSK Attribute Number>`	View a specific extended attribute using The Sleuth Kit

Log Analysis

`bzcat system.log.1.bz2 system.log.0.bz2 >> system_all.log` `cat system.log >> system_all.log`	Create a “all-in-one” system.log file. Can also be used with gzcatfor Gzip compressed log files
`syslog -f <file> | -d <directory>`	View ASL File or Directory of ASL files
`syslog -T utc -F raw -d /var/log/asl`	Output ASL files the /var/log/asl directory and output in raw format with UTC timestamps.
`praudit -xn /var/audit/*`	View audit logs in XML format without user/group resolution
`sudo log collect`	Create a logarchive bundle on live system, root required
`log show`	View logs in logarchive bundle (use with --predicate to filter)
`log stream`	View live logs (use with --predicate to filter)

Time Machine

`tmutil uniquesize <machinedirectory_path>/*`	Show the unique sizes of each snapshot
`tmutil calculatedrift <machinedirectory_path>`	Show the size changes (added/removed/changed) between each snapshot
`tmutil compare <snapshotdirectory1> <snapshotdirectory2>`	Compare the file changes (added/removed/changed) between two snapshots

Spotlight

`mdls <file>`	List the Spotlight metadata for a file
`mdfind "<attribute_name> == *"`	Find files based on a specific metadata query
`mdfind -onlyin /Volumes/mounted_disk`	Find files only in a certain directory or mounted image.
`mdimport -X | -A`	Print a list of attributes that can be queried.

Memory Analysis & Encrypted Containers

`vol.py --profile=<profile> -f <memory image> <plugin>`	Volatility Usage
`hdiutil attach –readonly –nomount –stdinpass filevault2image.dmg`	Mount a FileVault volume using a password
`security unlock-keychain FileVaultMaster.keychain diskutil corestorage unlockvolume <UUID> -recoverykeychain FileVaultMaster.keychain`	Access and mount a FileVault volume using a master password
`diskutil corestorage unlockvolume <UUID> -passphrase <recovery key>`	Mount a FileVault volume using the Recovery Key
`hdiutil attach -readonly -nomount -stdinpass sekretstuff_USB.dmg`	Mount an Encrypted DMG File
`strings <MemoryImage> | sort -u > dictionary.txt`	Create a dictionary file

Disk Arbitration

```
sudo launchctl load /System/Library/LaunchDaemons/com.apple.diskarbitrationd.plist
sudo launchctl unload /System/Library/LaunchDaemons/com.apple.diskarbitrationd.plist
ps auxw | grep diskarbitrationd
```

Image Mount & Eject

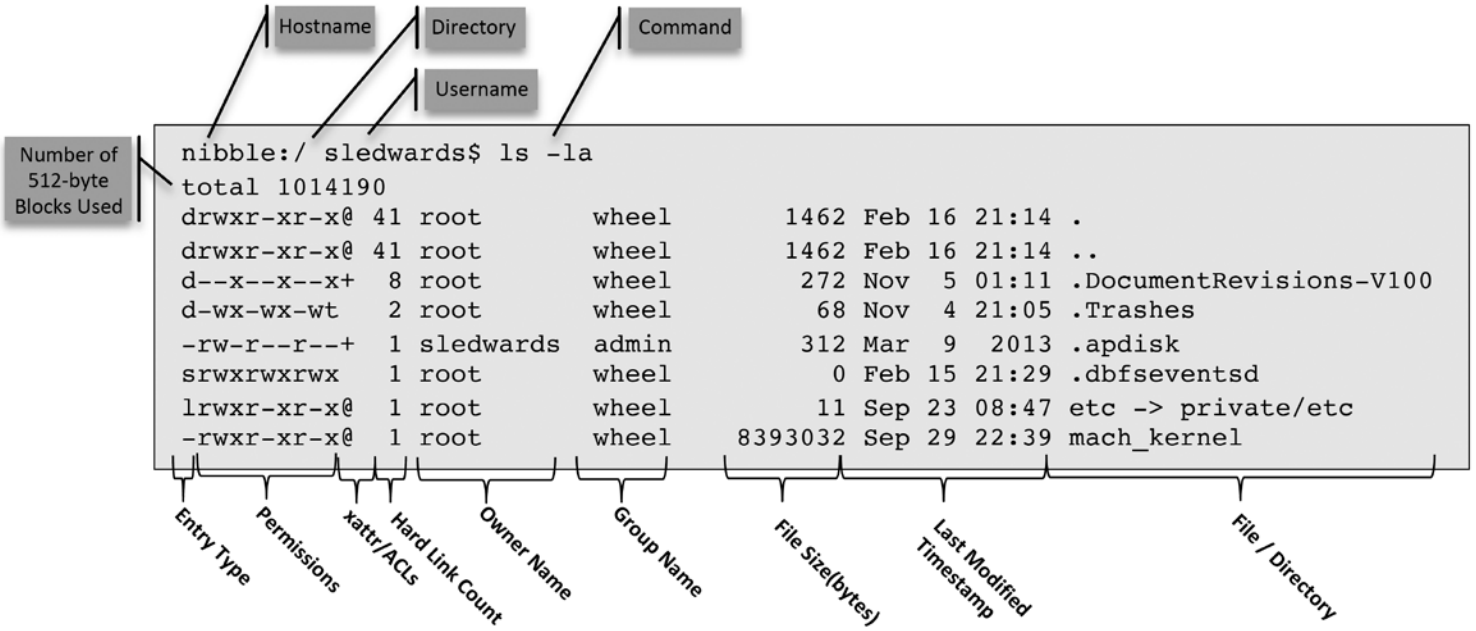
APFS with xmount (xmount v.0.7.*)	`$ sudo mkdir /Volumes/galaga_image/`
	`$ sudo mkdir /Volumes/galaga_mounted/`
	`$ sudo xmount --in ewf ~/FOR518/galaga.E01 --out dmg /Volumes/galaga_image/`
	`$ hdiutil attach –nomount /Volumes/galaga_image/galaga.dmg`
	`$ sudo mount_apfs -o rdonly,noexec,noowners /dev/disk# /Volumes/galaga_mounted/`

HFS+ Method 1 – xmount (xmount v.0.7.*)	`$ mkdir /Volumes/dademurphy_image/`
	`$ mkdir /Volumes/dademurphy_mounted/`
	`$ sudo xmount --in ewf ~/FOR518/dademurphy.E01 --out dmg /Volumes/dademurphy_image/`
	`$ hdiutil attach –nomount /Volumes/dademurphy_image/dademurphy.dmg`
	`$ mount_hfs -j -o rdonly,noexec,noowners /dev/disk# /Volumes/dademurphy_mounted/`

Eject Disk	`$ diskutil list`
	`$ diskutil eject /dev/disk#`
	`$ mount`
	`$ sudo umount /Volumes/galaga_image/`

Timestamp Formats

APFS	64-bit – Number of Seconds from 1/1/1970 00:00:00 UTC
HFS+/MacOS	32-bit – Number of seconds from 1/1/1904 00:00:00 UTC
UNIX Epoch	32-bit – Number of seconds from 1/1/1970 00:00:00 UTC
Mac Epoch/Mac Absolute/Cocoa/WebKit	32-bit – Number of seconds from 1/1/2001 00:00:00 UTC
Property List Dates in Xcode	Local Host System Time



GPT Header

Offset	Size (in bytes)	Field
0	8	Signature (EFI PART)
8	4	Revision (1.0)
12	4	Size of Header (bytes)
16	4	Header CRC32
20	4	Reserved
24	8	LBA of GPT Header
32	8	LBA of Backup GPT Header
40	8	First Usable LBA
48	8	Last Usable LBA
56	16	Disk GUID
72	8	Starting LBA of GUID Partition Table (Little Endian)
80	4	Number of Partition Entries Available (Little Endian)
84	4	Size of Partition Entry
88	4	Partition Entry Array CRC32
92	Rest	Reserved

GPT Table Entry

Offset	Size (in bytes)	Field
0	16	Partition Type GUID
16	16	Unique Partition GUID
32	8	Starting LBA (Little Endian)
40	8	Ending LBA (Little Endian)
48	8	Attributes
56	72	Partition Name
128	Rest	Reserved

GPT Partitions GUIDs

Type	Common GPT Partition GUIDs
EFI System Partition	C12A7328-F81F-11D2-BA4B-00A0C93EC93B
HFS+ Partition	48465300-0000-11AA-AA11-00306543ECAC
Apple Boot Partition	426F6F74-0000-11AA-AA11-00306543ECAC
Apple CoreStorage (possible FileVault or Fusion Drive)	53746F72-6167-11AA-AA11-00306543ECAC
APFS Partition	7C3457EF-0000-11AA-AA11-00306543ECAC
Basic Data Partition (Boot Camp)	EBD0A0A2-B9E5-4433-87C0-68B6B72699C7

The aim of this guide is to provide a list of the most interesting files and folders in the "Data" and "Shared" folders for the most commonly used third-party apps.

iOS third-party apps can be installed from the Apple App Store, where they are organized based on categories (e.g., Social Networking, Business/Productivity, Navigation & Travel, and so on).

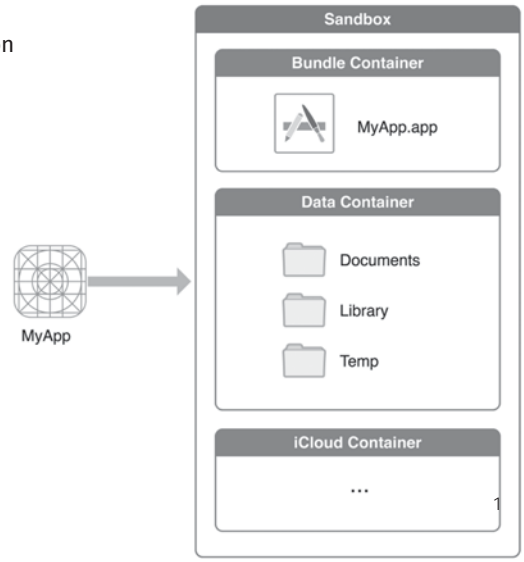
Once an app is installed on an iOS device:

- App Bundle is installed in a subfolder in the **/private/var/containers/Bundle** folder
- App Data is stored in a subfolder in the **/private/var/mobile/Containers/Data/Application/** folder (App Sandbox)

The easiest way to track down an iOS application's Data folder is to analyze the **/private/var/mobile/Library/FrontBoard/applicationstate.db** database, as described in a blog post by Alexis Brignoni[2].

Some Apps can also store data in other subfolders like the **/private/var/mobile/Containers/Share/AppGroup/** folder. Two good ways to locate the Sandbox folder for the AppGroup are mentioned in blog posts by Scott Vance[3] and Yogesh Khatri[4].

The internal structure of an App folder can be determined by the developer, but Apple provides some guidelines in its *File System Programming Guide*[5].



[1] Image is extracted from:
https://developer.apple.com/library/archive/documentation/FileManagement/Conceptual/FileSystemProgrammingGuide/FileSystemOverview/FileSystemOverview.html

[2] **https://abrignoni.blogspot.com/2018/12/identifying-installed-and-uninstalled.html**

[3] **https://blog.d204n6.com/2020/09/ios-tracking-bundle-ids-for-containers.html**

[4] **https://www.swiftforensics.com/2021/01/ios-application-groups-shared-data.html**

[5] **https://developer.apple.com/library/archive/documentation/FileManagement/Conceptual/FileSystemProgrammingGuide/FileSystemOverview/FileSystemOverview.html**

Entertainment/Photo & Video

Amazon Prime Video

APPSTORE URL: **https://apps.apple.com/us/app/amazon-prime-video/id545519333**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Application Support/<ID>/	store	JSON
/Library/Application Support/com.amazon.AIVWebImageCache/	*	Various
/Library/Caches/com.amazon.AIVAdvertCache/	*	Various
/Library/Preferences/	com.amazon.aiv.AIVApp.plist	Plist

Google Photos

APPSTORE URL: **https://apps.apple.com/us/app/google-photos/id962194608**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Caches/com.google.commmon.SSO/<User_ID>/	Profile.plist	Plist
/Library/Application Support/store/	collections-<User_ID>	SQLite
/Library/Application Support/store/	photos-<User_ID>	SQLite
/Library/Preferences/	com.google.photos.plist	Plist

Instagram

APPSTORE URL: **https://apps.apple.com/us/app/instagram/id389801252**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	time_in_app_<User_ID>.db	SQLite
/Library/Caches/<User_ID>/	*	Various
/Library/Caches/com.burbn.instagram.IGImageCache/	*	Images
/Library/Caches/com.burbn.instagram.IGSparseVideoCache/	*	Videos
/Library/Caches/Items/	lastentries.<User_ID>.1.coded	Plist
/Library/Application Support/<User_ID>/	pending-requests.plist	Plist
/Library/Application Support/DirectSQLiteDatabase/	<User_ID>.db	SQLite
/Library/Preferences/	com.burbn.instagram.plist	Plist

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/<User_ID>/user_bootstrap/	shared_bootstraps.plist	SQLite
/Library/Preferences/	group.com.burbn.instagram.plist	Plist

REFERENCES:

https://salt4n6.com/2018/05/15/a-few-interesting-ios-forensic-artefacts

http://xml.jips-k.org/full-text/view?doi=10.3745/JIPS.03.0097

https://www.forensicfocus.com/articles/forensic-analysis-of-third-party-application-instagram

Imgur

APPSTORE URL: **https://apps.apple.com/us/app/imgur-funny-meme-gif-maker/id639881495**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	default.realm	Realm
/Library/Caches/com.hackemist.SDImageCache/default/	*	Various
/Library/Caches/Logs/	*.log	TXT
/Library/Preferences/	imgurmobile.plist	Plist

REFERENCES:

https://abrignoni.blogspot.com/2019/12/ios-imgur-app-realm-database-example.html

Netflix

APPSTORE URL: **https://apps.apple.com/us/app/netflix/id363590051**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	store.sqlite	SQLite
/Library/assetCache/	*	JPG
/Library/Caches/br/ch/	*	Various
/Library/Caches/com.netflix.Netflix/	Cache.db	SQLite
/Library/Caches/com.netflix.Netflix/fsCachedData/	*	Various
/Library/Preferences/	com.netflix.Netflix.plist	Plist

Pinterest

APPSTORE URL: **https://apps.apple.com/us/app/pinterest/id429047995**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	activeUser	Plist
/Documents/	activeUser<User_ID>	Plist
/Library/Caches/com.pinterest.PINDiskCache.PINRemoteImageManagerCache/	*	Various
/Library/Caches/com.pinterest.PINDiskCache.PINRemoteModelCache/	*	Various
/Library/Preferences/	pinterest.plist	Plist

Snapchat

APPSTORE URL: **https://apps.apple.com/us/app/snapchat/id447188370**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	user.plist	Plist
/Documents/	chatConversationStore.plist	Plist
/Documents/	friendsForAsyncDecode.plist	Plist
/Documents/	stories.plist	Plist
/Documents/gallery_data_object/1/<User_ID>/	scdb-27.sqlite3	SQLite
/Documents/global_scoped/Gallery/	*	Various
/Library/Caches/com.snap.file_manager_1_SCContent_<GUID>/	*	Various
/Library/Caches/SCCache/	*	Various
/Library/Preferences/	com.toyopagroup.picaboo.plist	Plist

REFERENCES:

https://www.researchgate.net/profile/Imam_Riadi/publication/320467249_The_digital_forensic_analysis_of_snapchat_application_using_XML_records/links/59e73e87a6fdcc0e882d82e7/The-digital-forensic-analysis-of-snapchat-application-using-XML-records.pdf

https://www.marshall.edu/forensics/files/Cindy-Q.-Wu-Forensic-Analysis-of-Data-Transience-PPT.pdf

https://resources.infosecinstitute.com/ios-application-security-part-10-ios-filesystem-and-forensics/#gref

https://doubleblak.com/blogPosts.php?id=5

http://www.carpeindicium.com/blog/gone_10-seconds

Private Photo Vault

APPSTORE URL: **https://apps.apple.com/us/app/private-photo-vault-pic-safe/id417571834**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Application Support/	PPVCoreData.sqlite	SQLite
/Library/PPV_Pics/	*	Various
/Library/Preferences/	com.enchantedcloud.photovault.plist	Plist

REFERENCES:

https://cdn.ymaws.com/www.oshean.org/resource/resmgr/Email/ruledtheword.pdf

Spotify

APPSTORE URL: **https://apps.apple.com/us/app/spotify-music-and-podcasts/id324684580**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Application Support/PersistentCache/	mercury.db	SQLite
/Library/Application Support/PersistentCache/Storage/	*	JPG
/Library/Application Support/Users/<User_ID>-user/	*	Various
/Library/Caches/com.spotify.client/nsurlcache/fsCachedData/	*	JPG
/Library/Preferences/	com.spotify.client.plist	Plist

REFERENCES:

https://arstechnica.com/information-technology/2016/11/for-five-months-spotify-has-badlyabused-users-storage-drives

YouTube

APPSTORE URL: **https://apps.apple.com/us/app/tiktok/id835599320**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Caches/com.google.commmon.SSO/<User_ID>/	Profile.plist	Plist
/Library/Caches/com.pinterest.PINDiskCache.com.youtube.innertube.imageservice.cache/	*	Various
/Library/Preferences/	com.google.ios.youtube.plist	Plist

TikTok

APPSTORE URL: **https://apps.apple.com/us/app/tiktok/id835599320**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	AwemeIM.db	SQLite
/Documents/drafts/	*	Various
/Library/Application Support/ChatFiles/<User-ID>/	db.sqlite	SQLite
/Library/AWEVideoCache/FileCache/	*	M4V
/Library/Caches/com.ibireme.yykit/images/data/	*	Various
/Library/Caches/TTPlayerCache/	*	M4V
/Library/Heimdallr/	heimdallr.db	SQLite
/Library/Preferences/	com.zhiliaoapp.musically.plist	Plist

REFERENCES:

https://abrignoni.blogspot.com/2018/11/finding-tiktok-messages-in-ios.html

https://www.systoolsgroup.com/updates/retrieve-messages-from-tiktok

https://blog.oxygen-forensic.com/whos-knocking-tiktok

Business/Production

Doodle

APPSTORE URL: **https://apps.apple.com/us/app/doodle-easy-scheduling/id938182547**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	doodle.yapdb	SQLite
/Library/Caches/com.doodle.Doodle-App/fsCachedData/	*	Various
/Library/Preferences/	com.doodle.Doodle-App.plist	Plist

Dropbox

APPSTORE URL: **https://apps.apple.com/us/app/dropbox-backup-sync-share/id327630330**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	spotlight.db	SQLite
/Documents/Users/<User_ID>/	Dropbox.sqlite	SQLite
/Documents/Users/<User_ID>/	metadata.db	SQLite
/Documents/Users/<User_ID>/	offline.db	SQLite
/Documents/Users/<User_ID>/	recent_actions_local.db	SQLite
/Documents/Users/<User_ID>/	recent_actions_server.db	SQLite
/Documents/Users/<User_ID>/	starred_infos_local.db	SQLite
/Library/Cache/Users/<User_ID>/FileCache/Loaded/	*	Various
/Library/Preferences/	com.getdropbox.Dropbox.plist	Plist

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/File Provider Storage/<User-ID>/local-storage/	*	Various
/Library/Preferences/	group.com.getdropbox.Dropbox.plist	Plist
/Users/<User_ID>/	Dropbox.sqlite	SQLite
/Users/<User_ID>/	file_provider_metadata_with_assistant.db	SQLite
/Users/<User_ID>/	upload_queue_v2.db	SQLite
/Users/<User_ID>/FileCache/	*	Various

REFERENCES:

https://abrignoni.blogspot.com/2018/12/profiling-user-activity-in-dropbox-for.html

https://www.marshall.edu/forensics/files/Treleven-Dropbox-Paper-FINAL.pdf

https://arxiv.org/ftp/arxiv/papers/1709/1709.10395.pdf

https://link.springer.com/article/10.1007/s11227-020-03255-5

https://www.tandfonline.com/doi/abs/10.1080/00450618.2015.1110620?scroll=top&needAccess=true&journalCode=tajf20

Dust

APPSTORE URL: **https://apps.apple.com/us/developer/radical-app-llc/id1004670836**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	<User_ID>	Realm
/Documents/	contacts.json	JSON
/Documents/	default.realm	Realm
/Library/Preferences/	com.mentionmobile.cyberdust.plist	Plist
/SplashBoard/Snapshots/	*	KTX

REFERENCES:

https://www.nw3c.org/docs/research/dust.pdf

Eventbrite

APPSTORE URL: **https://apps.apple.com/us/app/eventbrite/id487922291**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	default.realm	Realm
/Library/Caches/com.eventbrite.attendee/com.alamofire.imagedownloader/fsCachedData/	*	JPG
/Library/Preferences/	com.eventbrite.attendee.plist	Plist

Gmail

APPSTORE URL: **https://apps.apple.com/us/app/gmail-email-by-google/id422689480**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Preferences/	com.google.Gmail.plist	
/Library/Caches/com.google.commmon.SSO/<User_ID>/	Profile.plist	Plist
/Documents/drivekit/users/<User_ID>/gdx-cello/	cello.db	SQLite
/Library/Application Support/data/<Email_Address>/	sqlitedb	SQLite

Google Drive

APPSTORE URL: **https://apps.apple.com/us/app/google-drive/id507874739**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/<User_ID>/	comments_snapshot_<User_ID>.db	SQLite
/Documents/drivekit/users/<User_ID>/cello/	cello.db	SQLite
/Documents/drivekit/users/<User_ID>/files/	*	Various
/Documents/drivekit/users/<User_ID>/logs/	*	TXT
/Documents/drivekit/users/<User_ID>/thumbnails/	*	Various
/Library/Caches/drivekit/users/<User_ID>/image-fetcher-cache/main-cache/	cacheV0.db	SQLite
/Library/Preferences/	com.google.Drive.plist	Plist

LinkedIn

APPSTORE URL: **https://apps.apple.com/us/app/linkedin-network-job-finder/id288429040**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	Messenger.sqlite	SQLite
/Documents/LIImageCache	*	Various
/Library/Caches/WebKit/NetworkCache/	*	Various
/Library/Preferences/	com.linkedin.LinkedIn.plist	Plist

REFERENCES:

https://www.tandfonline.com/doi/abs/10.1080/00450618.2015.1066854?src=recsys&journalCode=tajf20

https://digitalcommons.newhaven.edu/cgi/viewcontent.cgi?article=1017&context=electricalcomputerengineering-facpubs

https://it.scribd.com/document/57611870/Shmoocon-2011-Inside-the-App-All-Your-Data-are-Belong-to-Me

LogMeIn

APPSTORE URL: **https://apps.apple.com/us/app/logmein/id479229407**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/LogMeInConfig/	com.logmein.ignition.xml	XML
/Library/LogMeInConfig/	sessiondata.xml	XML
/Library/Preferences/	com.logmein.logmein.plist	Plist

Microsoft OneDrive

APPSTORE URL: **https://apps.apple.com/us/app/microsoft-onedrive/id477537958**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Database/	moddatabase.db	SQLite
/Library/Preferences/	com.microsoft.skydrive.plist	Plist
/SplashBoard/Snapshots/	*	KTX

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Preferences/	group.com.microsoft.onedrive.plist	Plist
/OneDrive/DatabaseQT/	QTMetadata.db	SQLite
/OneDrive/StramCacheQT/	*	Various

Microsoft Teams

APPSTORE URL: **https://apps.apple.com/us/app/microsoft-teams/id1113153706**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/.IntuneMAM/	Config.plist	Plist
/Library/Shiftr/	Shiftr.sqlite	SQLite
/Library/Preferences/	com.microsoft.skype.teams.plist	Plist
/SplashBoard/Snapshots/	*	KTX

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Preferences/	group.com.microsoft.skype.teams.plist	Plist
/SkypeSpacesDogfood/<Team_ID>/	SkypeSpacesDogfood-<GUID>.sqlite	SQLite
/SkypeSpacesDogfood/Downloads/<Team_ID>/	*	Various

ProtonMail

APPSTORE URL: **https://apps.apple.com/us/app/protonmail-encrypted-email/id979659905**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Caches/SentryCras/ProtonMail/Data/	CrashState.json	JSON
/Library/Preferences/	ch.protonmail.protonmail.plist	Plist

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/	ProtonMail.sqlite	SQLite
/Library/Preferences/	group.ch.protonmail.protonmail.plist	Plist

Silent Phone

APPSTORE URL: **https://apps.apple.com/us/app/silent-phone/id554269204**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Application Support/ com.silentcircle.SilentPhone/Chat/	ChatMessages_cipher.db	SQLite
/Library/Application Support/com.silentcircle.SilentPhone/tivi/	axo_<User-ID>_secure_sql.db	SQLite
/Library/Application Support/com.silentcircle.SilentPhone/tivi/	zids_sqlite.db	SQLite
/Library/Preferences/	com.silentcircle.SilentPhone.plist	Plist

Slack

APPSTORE URL: **https://apps.apple.com/us/app/slack/id618783545**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Application Support/Slack/<Workspace_id>/Database/	main_db	SQLite
/Library/Caches/com.tinyspeck.chatlyio/fsCachedData/	*	JPG
/Library/Caches/default/com.hackemist.SDWebImageCache.default/	*	JPG
/Library/Preferences/	com.tinyspeck.chatlyio.plist	Plist

REFERENCES:

https://abrignoni.blogspot.com/2018/10/finding-slack-app-messages-in-ios.html

Tutanota

APPSTORE URL: **https://apps.apple.com/us/app/tutanota/id922429609**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Preferences/	de.tutao.tutanota.plist	Plist
/Library/WebKit/WebsiteData/IndexedDB/v1/file__0/	IndexedDB.sqlite3	SQLite
/Library/WebKit/WebsiteData/LocalStorage/	file__0.localstorage	SQLite
/SplashBoard/Snapshots/	*	KTX

Business/Production (CONTINUED)

Wire

APPSTORE URL: **https://apps.apple.com/us/app/wire-secure-messenger/id930944768**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Preferences/	com.wearezeta.zclient.ios.plist	Plist
/SplashBoard/Snapshots/	*	KTX

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/AccountData/<GUID>/store/	store.wiredatabase	SQLite
/Accounts/	*	JSON
/Library/Caches/	*	Various
/Library/Preferences/	group.com.wearezeta.zclient.ios.plist	Plist

REFERENCES:

https://www.x41-dsec.de/reports/X41-Kudelski-Wire-Security-Review-iOS.pdf

https://blog.oxygen-forensic.com/wire-app-extraction

https://wire-docs.wire.com/download/Wire+Security+Whitepaper.pdf

ZOOM

APPSTORE URL: **https://apps.apple.com/us/app/zoom-cloud-meetings/id546505307**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/data/	*	JPG
/Documents/data/	zoommeeting.db	SQLite
/Documents/data/	zoomus.db	SQLite
/Documents/data/	zoomus.tmp.db	SQLite
/Documents/data/<User_ID>/	<User_ID>@xmpp.zoom.us.asyn.db	SQLite
/Documents/data/<User_ID>/	<User_ID>@xmpp.zoom.us.db	SQLite
/Documents/data/<User_ID>/	<User_ID>@xmpp.zoom.us.idx.db	SQLite
/Documents/data/<User_ID>/	<User_ID>@xmpp.zoom.us.sync.db	SQLite
/Library/Preferences/	us.zoom.videomeetings.plist	Plist

REFERENCES:

https://www.hecfblog.com/2020/04/daily-blog-684-solution-saturday-42520.html

Navigation & Travel

Air France

APPSTORE URL: **https://apps.apple.com/us/app/air-france/id391968627**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/	bagtracking.realm	Realm
/	checkinkit.realm	Realm
/	entertainmentkit.realm	Realm
/	flightStatusKit.realm	Realm
/	reservation.realm	Realm
/	trackandtrace.realm	Realm

Airbnb

APPSTORE URL: **https://apps.apple.com/app/airbnb/id401626263**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	guest_inbox_<User_ID>.db	SQLite
/Documents/	host_experience_inbox_<User_ID>.db	SQLite
/Documents/	host_inbox_<User_ID>.db	SQLite
/Library/Application Support/	guest_inbox_<User_ID>.db	SQLite
/Library/Application Support/	messaging_core.sqlite3	SQLite
/Library/Caches/	<User_ID>_itinerary.db	SQLite
/Library/Caches/	AIRAccount_active_account_login	Plist
/Library/Caches/	AIRUser_<User_ID>	Plist
/Library/Caches/	AIRWishlist_<User_ID>	Plist
/Library/Caches/default/ com.hackemist.SDWebImageCache.default/	*	PNG
/Library/Preferences/	com.airbnb.app.plist	Plist

REFERENCES:

https://blog.oxygen-forensic.com/untangling-airbnb/

Foursquare Swarn

APPSTORE URL: **https://apps.apple.com/us/app/foursquare-swarm-check-in-app/id870161082**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Caches/	foursquare.sqlite	SQLite
/Library/Preferences/	com.foursquare.robin.plist	Plist

Booking

APPSTORE URL: **https://apps.apple.com/us/app/booking-com-hotels-travel/id367003839**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Application Support/	BookingClouds	Plist
/Library/Application Support/	KeyValueStorageAccountDomain	Plist
/Library/Application Support/	KeyValueStorageRecentsDomain	Plist
/Library/Application Support/	KeyValueStorageSharedDomain	Plist
/Library/Caches/	location_cache_V2.db	SQLite
/Library/Caches/com.booking.BookingApp/com. alamofire.imagedownloader/fsCachedData/	*	JPG
/Library/Caches/com.booking.BookingApp/fsCachedData/	*	JSON
/Library/Preferences/	com.booking.BookingApp.plist	Plist

Google Maps

APPSTORE URL: **https://apps.apple.com/us/app/google-maps-transit-food/id585027354**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/GMSCacheStorage-AZSpotlightStorageModel/GMSCacheStorage-AZSpotlightStorageModel/	AZSpotlightStorageModel.sqlite	SQLite
/Library/Application Support/CachedRoutes/	*	Plist
/Library/Application Support/GMSCacheStorage-MyMaps/	MyMaps.sqlite	SQLite
/Library/Application Support/GMSCacheStorage-SavedUserEvent3/	SavedUserEvent3.sqlite	SQLite
/Library/Application Support/GMSCacheStorage-Tiles/	Tiles.sqlite	SQLite
/Library/Caches/ImageCache	*	JPG
/Library/Caches/com.google.commmon.SSO/<User_ID>/	Profile.plist	Plist
/Library/Preferences/	com.google.Maps.plist	Plist

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Preferences/	CurrentDirections	Plist
/Library/Preferences/	group.com.google.Maps.plist	Plist

REFERENCES:

https://commons.erau.edu/cgi/viewcontent.cgi?article=1414&context=jdfsl

https://www.ijrte.org/wp-content/uploads/papers/v8i4/D4374118419.pdf

KLM

APPSTORE URL: **https://apps.apple.com/us/app/klm/id391732065**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	CoreAppRedesign_iPhone.sqlite	SQLite
/Library/Caches/com.klm.mobile.iphone.klmmobile/fsCachedData/	*	Various
/Library/Preferences/	com.klm.mobile.iphone.klmmobile.plist	Plist

Lufthansa

APPSTORE URL: **https://apps.apple.com/us/app/lufthansa/id299219152**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	*	HTML
/Library/Application Support/	Database.sqlite	SQLite
/Library/Caches/com.lufthansa.launcher/fsCachedData/	*	XML
/Library/Preferences/	com.lufthansa.launcher.plist	Plist

Skyscanner

APPSTORE URL: **https://apps.apple.com/us/app/skyscanner-travel-deals/id415458524**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	MiniEvents.sqlite	SQLite
/Documents/WatchedFlights/	WatchedFlights.json	JSON
/Library/Caches/com.hackemist.SDImageCache/default/	*	JPG
/Library/Caches/net.skyscanner.iphone/netcache/	Cache.db	SQLite
/Library/Caches/net.skyscanner.iphone/netcache/fsCachedData/	*	JSON
/Library/Preferences/	net.skyscanner.iphone.plist	Plist

Tripadvisor

APPSTORE URL: **https://apps.apple.com/us/app/uber/id368677368**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	database.db	SQLite
/Library/Application Support/com.ubercab.UberClient/	*	Various
/Library/Application Support/PersistentStorage/BootstrapStore/RealtimeRider.StreamModelKey/	client	JSON
/Library/Caches/com.ubercab.UberClient/com.uber.images/fsCachedData/	*	Various
/Library/Preferences/	com.ubercab.UberClient.plist	Plist

REFERENCES:

https://www.researchgate.net/publication/323759986_A_Dynamic_and_Static_Analysis_of_the_Uber_Mobile_Application_from_a_Privacy_Perspective

Uber

APPSTORE URL: **https://apps.apple.com/us/app/uber/id368677368**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	database.db	SQLite
/Library/Application Support/com.ubercab.UberClient/	*	Various
/Library/Application Support/PersistentStorage/BootstrapStore/RealtimeRider.StreamModelKey/	client	JSON
/Library/Caches/com.ubercab.UberClient/com.uber.images/fsCachedData/	*	Various
/Library/Preferences/	com.ubercab.UberClient.plist	Plist

REFERENCES:

https://www.researchgate.net/publication/323759986_A_Dynamic_and_Static_Analysis_of_the_Uber_Mobile_Application_from_a_Privacy_Perspective

Waze

APPSTORE URL: **https://apps.apple.com/us/app/waze-navigation-live-traffic/id323229106**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Document/	user	TXT
/Document/	session	TXT
/Document/	preferences	TXT
/Document/	user.db	SQLite
/Library/Preferences/	com.waze.iphone.plist	Plist

Brave Browser

APPSTORE URL: **https://apps.apple.com/us/app/brave-private-browser-vpn/id1052879175**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/Downloads/	*	Various
/Library/Application Support/	Brave.sqlite	SQLite
/Library/Preferences/	com.brave.ios.browser.plist	Plist

Burner

APPSTORE URL: **https://apps.apple.com/us/app/burner-private-phone-line/id505800761**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Caches/com.adhoclabs.burner/	Cache.db	SQLite
/Library/Caches/com.adhoclabs.burner/fsCachedData/	*	Various
/Library/Preferences/	com.adhoclabs.burner.plist	Plist
/SplashBoard/Snapshots/	*	KTX

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/	Phoenix.sqlite	SQLite

REFERENCES:

https://digitalforensicstips.com/2013/07/forensic-artifact-analysis-of-the-burner-app-for-the-iphone

DuckDuckGo Browser

APPSTORE URL: **https://apps.apple.com/us/app/duckduckgo-privacy-browser/id663592361**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Preferences/	com.duckduckgo.mobile.ios.plist	Plist

Firefox

APPSTORE URL: **https://apps.apple.com/us/app/firefox-private-safe-browser/id989804926**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Caches/WebKit/NetworkCache/	*	Various
/Library/Preferences/	org.mozilla.ios.Firefox.plist	Plist

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/profile.profile/	browser.db	SQLite
/profile.profile/	logins.db	SQLite
/profile.profile/	places.db	SQLite
/profile.profile/	tabState.archive	Plist
/Library/Preferences/	group.org.mozilla.ios.Firefox.plist	Plist

Firefox Focus

APPSTORE URL: **https://apps.apple.com/us/app/firefox-focus-privacy-browser/id1055677337**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Preferences/	org.mozilla.ios.Focus.plist	Plist
/Library/Caches/KSCrash/Firefox Focus/Data/	CrashState.json	JSON

Google Chrome

APPSTORE URL: **https://apps.apple.com/us/app/google-chrome/id535886823**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Application Support/Google/Chrome/Default/	*	Various
/Library/Caches/com.google.commmon.SSO/<User_ID>/	Profile.plist	Plist
/Library/Preferences/	com.google.chrome.ios.plist	Plist

Microsoft Edge

APPSTORE URL: **https://apps.apple.com/us/app/microsoft-edge-web-browser/id1288723196**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	OfflineCache.sqlite	SQLite
/Documents/CitrixLogs/Diagnostics/	*	CSV
/Documents/TabScreenshot/	*	JPG
/Library/Application Support/ChromeSync/	*	Various
/Library/Caches/com.microsoft.msedge/fsCachedData/	*	Various
/Library/Preferences/	com.microsoft.msedge.plist	Plist

Onion Browser

APPSTORE URL: **https://apps.apple.com/us/app/onion-browser/id519296448**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	bookmarks.plist	Plist
/Library/Caches/com.miketigas.OnionBrowser/	Cache.db	SQLite
/Library/Caches/tor/	state	TXT
/Library/Preferences/	com.miketigas.OnionBrowser.plist	Plist

REFERENCES:

https://roselabs.nl/files/audit_reports/Cure53_-_Onion_Browser.pdf

CoverMe

APPSTORE URL: **https://apps.apple.com/us/app/coverme-private-text-call/id593652484**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	ContactCache	Plist
/Documents/	miliao.db	SQLite
/Documents/	notification.plist	Plist
/Documents/	Profile.plist	Plist
/Library/logs/	*.log	TXT
/Library/Preferences/	com.coverme.covermeAdhoc.plist	Plist

Discord

APPSTORE URL: **https://apps.apple.com/us/app/discord-talk-chat-hang-out/id985746746**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/mmkv/	mmkv.default	JSON
/Documents/RCTAsyncLocalStorage_V1/	<GUID>	JSON
/Documents/RCTAsyncLocalStorage_V1/	manifest.json	JSON
/Library/Caches/com.hackemist.SDImageCache/	*	PNG
/Library/Caches/com.hammerandchisel.discord/	Cache.db	SQLite
/Library/Caches/com.hammerandchisel.discord/fsCachedData/	*	JSON
/Library/Preferences/	com.hammerandchisel.discord.plist	Plist

REFERENCES:

https://abrignoni.blogspot.com/2018/08/finding-discord-chats-in-ios.html

https://abrignoni.blogspot.com/2020/08/update-on-discord-forensic-artifacts.html

https://www.nw3c.org/docs/research/discord.pdf

Facebook

APPSTORE URL: **https://apps.apple.com/us/app/facebook/id284882215**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	time_in_app_<User_ID>.db	SQLite
/Library/Caches/com.facebook.Facebook.MosaicIGImageDiskCache/	*	Various
/Library/Caches/graphStoreDB/	GraphStore_<User_ID>.sqlite3	SQLite
/Library/Caches/messenger_contacts.<GUID>/	fbsyncstore.db	SQLite
/Library/Caches/search_bootstrap.<GUID>/search/	graph_search_entity_bootstrap.data	File
/Library/Caches/video_cache.<GUID>/storage/	*	Plist
/Library/Preferences/	com.facebook.Facebook.plist	Plist

REFERENCES:

https://www.academia.edu/10726810/Social_Media_Forensics_on_Mobile_Devices

https://www.tandfonline.com/doi/abs/10.1080/00450618.2015.1066854?src=recsys&journalCode=tajf20

https://www.fbiic.gov/public/2011/jul/Facebook_Forensics-Finalized.pdf

https://www.researchgate.net/publication/224221519_Third_Party_Application_Forensics_on_Apple_Mobile_Devices

https://www.diva-portal.org/smash/get/diva2:651693/fulltext01.pdf

Facebook Messenger

APPSTORE URL: **https://apps.apple.com/us/app/messenger/id454638411**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/com.facebook.Messenger.preferences/	<User_ID>.session	Plist
/Documents/messenger_secure_messages.sessionless.1/	<User_ID>_threadStateStore.db	SQLite
/Documents/messenger_secure_messages.sessionless.1/	<User_ID>_v1464784789_tincan.db	SQLite
/Library/Caches/graphStoreDB/	GraphStore_<User_ID>.sqlite3	SQLite
/Library/Preferences/	com.facebook.Messenger.plist	Plist

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/	lightspeed-<User_ID>.db	SQLite
/_store_<GUID>/messenger_contacts.v1/	fbomnistore.db	SQLite
/shared_messenger_contacts.<GUID>/	fbomnistore.db	SQLite
/shared_messenger_messages.<GUID>/	orca2.db	SQLite
/lightspeed-imageCache/	*	Various

REFERENCES:

https://www.academia.edu/10726810/Social_Media_Forensics_on_Mobile_Devices

https://boncaldoforensics.wordpress.com/2018/07/28/facebook-messenger-windows-app-store-forensics

https://www.champlain.edu/Documents/LCDI/iPhone%20Artifacts.pdf

https://sqliteforensictoolkit.com/forensic-browser-for-sqlite-structured-storage-manager

Google Duo

APPSTORE URL: **https://apps.apple.com/us/app/google-duo/id1096918571**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/logs/	*	TXT
/Library/Application Support/	DataStore	SQLite
/Library/Caches/com.google.commmon.SSO/<User_ID>/	Profile.plist	Plist
/Library/Preferences/	com.google.Tachyon.plist	Plist

Houseparty

APPSTORE URL: **https://apps.apple.com/us/app/houseparty/id1065781769**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	houseparty.rocky.phonenumbers	Realm
/Documents/	houseparty.rocky.realm	Realm
/Library/Caches/com.herzick.houseparty/fsCachedData/	*	JSON
/Library/IBGLog/	INGUserAttribute.txt	TXT
/Library/Preferences/	com.herzick.houseparty.plist	Plist
/Library/Preferences/	Houseparty.plist	Plist
/SplashBoard/Snapshots/	*	KTX

REFERENCES: **https://abrignoni.blogspot.com/2020/04/ios-houseparty-app-more-realm.html**

imo

APPSTORE URL: **https://apps.apple.com/us/app/imo-video-calls-and-chat-hd/id1400579543**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	imo_acc	Plist
/Documents/	imo_last_ts_log_appAlive	Plist
/Documents/	imo_save_media_local_setting_1	Plist
/Documents/	imo_stories	Plist
/Library/Caches/default/com.hackemist.SDWebImageCache.default/	*	Various
/Library/Caches/videos/	*	Various
/Library/Preferences/	co.babypenguin.imo.plist	Plist
/SplashBoard/Snapshots/	*	KTX

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/	IMODb2.sqlite	SQLite
/	IMOShareDb.sqlite	SQLite
/Library/Preferences/	group.co.babypenguin.plist	Plist

REFERENCES:

https://www.sciencedirect.com/science/article/abs/pii/S1742287618300094

http://prr.hec.gov.pk/jspui/bitstream/123456789/13429/1/Muhammad%20Asad%20Khan%20Sudozai%20electrical%20engg%202019%20NUST%20isb%20prr.pdf

https://www.champlain.edu/Documents/LCDI/iPhone%20Artifacts.pdf

https://sqliteforensictoolkit.com/forensic-browser-for-sqlite-structured-storage-manager

Kik Messenger

APPSTORE URL: **https://apps.apple.com/us/app/kik/id357218860**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Caches/com.kik.chat/fsCachedData/	*	Various
/Library/Preferences/	com.kik.chat.plist	Plist
/SplashBoard/Snapshots/	*	KTX

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/cores/private/<User-ID>/	kik.sqlite	SQLite
/cores/private/<User-ID>/app-lock/	app-lock-settings	JSON
/cores/private/<User-ID>/attachments/	*	Various
/cores/private/<User-ID>/defaults/	kik.defaults	Plist
/cores/private/<User-ID>/globalDefaults/	kik.defaults	Plist
/cores/private/<User-ID>/suggested-chats/	suggested	JSON
/cores/private/<User-ID>/urlData/	*	PNG
/globalDefaults/	kik.defaults	Plist
/Library/Preferences/	group.com.kik.chat.plist	Plist

REFERENCES:

https://www.sciencedirect.com/science/article/pii/B9781597496599000067

https://researchonline.gcu.ac.uk/files/24282895/K.Ovens_revisedKMOvensManuscript3_2.pdf

https://www.scribd.com/doc/145278610/Artefacts-of-Kik-Messenger-on-iOS

https://blog.oxygen-forensic.com/kickin-kik

LINE

APPSTORE URL: **https://apps.apple.com/us/app/line/id443904275**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Application Support/PrivateStore/	*	Various
/Library/Caches/jp.naver.line/fsCachedData/	*	Various
/Library/Preferences/	jp.naver.line.plist	Plist

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Application Support/KeepFileProvider/	Keep.sqlite	SQLite
/Library/Application Support/PrivateStore/<User-ID>/Messages/	ChetExt.sqlite	SQLite
/Library/Application Support/PrivateStore/<User-ID>/Messages/	E2EEData.sqlite	SQLite
/Library/Application Support/PrivateStore/<User-ID>/Messages/	Line.sqlite	SQLite
/Library/Preferences/	group.com.linecorp.line.plist	Plist

REFERENCES:

https://prezi.com/mloxlacowypf/iphone-forensic-line/

https://reincubate.com/support/how-to/recover-iphone-hike-line-wechat-messages/

https://pdfs.semanticscholar.org/fe66/52f6fe64ce1af44dd7d433ecf5a00b57ca0a.pdf

MeWe

APPSTORE URL: **https://apps.apple.com/us/app/mewe-network/id918464474**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	Sgrouplesdb.sqlite	SQLite
/Library/Caches/com.mewe/fsCachedData/	*	Various
/Library/Caches/com.hackemist.SDImageCache/	*	Various
/Library/Preferences/	com.mewe.plist	Plist

Signal

APPSTORE URL: **https://apps.apple.com/us/app/signal-private-messenger/id874139669**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Caches/	*	JPG
/Library/Logs/	*.log	TXT
/Library/Preferences/	org.whispersystems.signal.plist	Plist

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/Attachments/	*	Various
/grdb/	signal.sqlite	SQLite
/Library/Preferences/	group.org.whispersystems.signal.group.plist	Plist
/ProfileAvatars/	*	JPG

REFERENCES:

https://github.com/Magpol/HowTo-decrypt-Signal.sqlite-for-iOS

https://support.magnetforensics.com/s/article/Decrypt-app-data-using-the-iOS-Keychain-and-GrayKey

https://pdfs.semanticscholar.org/fe66/52f6fe64ce1af44dd7d433ecf5a00b57ca0a.pdf

https://isc.sans.edu/forums/diary/Looking+for+the+insider+Forensic+Artifacts+on+iOS+Messaging+App/21363

http://www.ijits-bg.com/contents/IJITS-No4-2019/2019-N4-07.pdf

Skout

APPSTORE URL: **https://apps.apple.com/us/app/skout-meet-new-people/id302324249**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Application Support/SKOUT/	SKCache.sqlite	SQLite
/Library/Caches/default/com.hackemist.SDWebImageCache.default/	*	Various
/Library/Preferences/	com.skout.SKOUT.plist	Plist
/SplashBoard/Snapshots/	*	KTX

Skype

APPSTORE URL: **https://apps.apple.com/us/app/skype-for-iphone/id304878510**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/RCTAsyncLocalStorage_V1/	manifest.json	JSON
/Library/Application Support/Skype4LifeSlimCore/<Skype_Username>/	main.db	SQLite
/Library/Caches/Logs/	com.skype.*.log	TXT
/Library/LocalDatabase/	s4l-<Skype_Username>.db	SQLite
/Library/Preferences/	com.skype.skype.plist	Plist

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/	s4l-<Skype_Username>.db	SQLite

REFERENCES:

https://bebinary4n6.blogspot.com/2019/07

https://pdfs.semanticscholar.org/fe66/52f6fe64ce1af44dd7d433ecf5a00b57ca0a.pdf

Telegram

APPSTORE URL: **https://apps.apple.com/us/app/telegram-messenger/id686449807**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Preferences/	ph.telegra.Telegraph.plist	Plist
/Library/Caches/ph.telegra.Telegraph/fsCachedData/	*	Various

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/telegram-data/account-<Account_ID>/postbox/db/	db_sqlite	SQLite
/telegram-data/account-<Account_ID>/postbox/media/	*	Various
/telegram-data/logs/	*	TXT
/telegram-data/share-logs/	*	TXT

REFERENCES:

https://www.forensicfocus.com/news/telegram-messenger-data-extraction-in-oxygen-forensic-detective

TextNow

APPSTORE URL: **https://apps.apple.com/us/app/textnow-call-text-unlimited/id314716233**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	<User-ID>	SQLite
/Documents/com.hackemist.SDWebImageCache.default/	*	JPG
/Library/Application Support/	eventHistory.db	SQLite
/Library/Caches/Logs/sip/	*.log	TXT
/Library/Caches/media/	*	JPG
/Library/Preferences/	com.tinginteractive.usms.plist	Plist

Tinder

APPSTORE URL: **https://apps.apple.com/us/app/tinder-dating-new-people/id547702041**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Application Support/	Tinder2.sqlite	SQLite
/Library/Preferences/	com.cardify.tinder.plist	Plist

Viber Messenger

APPSTORE URL: **https://apps.apple.com/us/app/viber-messenger-chats-calls/id382617920**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/Attachments/	*	Various
/Documents/ChatExIcons/	*	Various
/Documents/UserEngagement/	*	Various
/Library/Caches/com.viber/fsCachedData/	*	Various
/Library/Preferences/	com.viber.plist	Plist
/SplashBoard/Snapshots/	*	KTX

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/com.viber/AttachmentsPreview/	*	Various
/com.viber/ContactIcons/	*	JPG
/com.viber/database/	Contacts.data	SQLite
/com.viber/settings/	Settings.data	SQLite

REFERENCES:

https://pdfs.semanticscholar.org/fe66/52f6fe64ce1af44dd7d433ecf5a00b57ca0a.pdf

https://blog.oxygen-forensic.com/viber-messenger-forensics

https://blog.digital-forensics.it/2019/12/checkra1n-era-ep-4-analyzing.html

WeChat

APPSTORE URL: **https://apps.apple.com/us/app/wechat/id414478124**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/<User_ID>/	mmsetting.archive	Plist
/Documents/<User_ID>/	wc005_008.db	SQLite
/Documents/<User_ID>/DB/	MM.sqlite	SQLite
/Documents/<User_ID>/DB/	WCDB_Contact.sqlite	SQLite
/Library/Preferences/	com.tencent.xin.plist	Plist

REFERENCES:

https://www.researchgate.net/publication/261016959_Forensic_Analysis_of_Social_Networking_Application_on_iOS_devices

https://www.ictsecuritymagazine.com/articoli/wechat-forensics-parte-i

Social Networking (CONTINUED)

WhatsApp Messenger

APPSTORE URL: **https://apps.apple.com/us/app/whatsapp-messenger/id310633997**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	Blockedcontacts.dat	Plist
/Documents/	calls.backup.log	Plist
/Documents/	StatusMessages.plist	Plist
/Library/Caches/ChatMedia/	*	Various
/Library/Caches/GalleryMedia/	*	Various
/Library/Caches/net.whatsapp.WhatsApp/fsCachedData/	*	Various
/Library/Caches/spotlight-profile-v2/	*	PNG
/Library/Logs/	whatsapp-*.log	TXT
/Library/Preferences/	net.whatsapp.WhatsApp.plist	Plist
/SplashBoard/Snapshots/	*	KTX

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/	calls.log	Plist
/	CallHistory.sqlite	SQLite
/	ChatStorage.sqlite	SQLite
/	ContactsV2.sqlite	SQLite
/	current_wallpaper.jpg	JPG
/	Location.sqlite	SQLite
/Biz/	Biz.sqlite	SQLite
/fts/	ChatSearch*.sqlite	SQLite
/Library/Preferences/	group.net.whatsapp.WhatsApp.shared.plist	Plist
/Media/Profile/	*	Various
/Message/Media/	*	Various
/stickers/	*	Various

REFERENCES:

https://www.group-ib.com/blog/whatsapp_forensic_artifacts

https://pdfs.semanticscholar.org/fe66/52f6fe64ce1af44dd7d433ecf5a00b57ca0a.pdf

https://sudonull.com/post/30099-WhatsApp-in-the-palm-of-your-hand-where-and-how-can-you-detect-forensic-artifacts-Group-IB-Blog

https://www.ijesm.co.in/uploads/68/5543_pdf.pdf

http://www.securitybydefault.com/2011/06/what-whatsapp-doesnt-tell-you.html

Wickr Me

APPSTORE URL: **https://apps.apple.com/us/app/wickr-me-private-messenger/id528962154**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Caches/KSCrashReports/Wickr Me/	*	JSON
/Library/Caches/Sessions/Wickr Me/	*	JSON
/Library/Preferences/	com.mywickr.wickr.plist	Plist

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/	wickrLocal.sqlite	SQLite

REFERENCES:

https://thebinaryhick.blog/2019/08/23/wickr-alright-well-call-it-a-draw

https://blog.oxygen-forensic.com/wickr-some-forensics-up

https://support.magnetforensics.com/s/article/Decrypt-app-data-using-the-iOS-Keychain-and-GrayKey

News

Reddit

APPSTORE URL: **https://apps.apple.com/us/app/reddit/id1064216828**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/release02/accounts/	*	Plist
/Documents/release02/accountData/	*	Plist
/Library/Caches/com.reddit.Reddit/fsCachedData/	*	Various
/Library/Caches/com.reddit.Reddit/imagedownload/fsCachedData/	*	Various
/Library/Preferences/	com.reddit.Reddit.plist	Plist

Twitter

APPSTORE URL: **https://apps.apple.com/us/app/twitter/id333903271**

DEVELOPER WEBSITE: **http://twitter.com/download/iphone**

APP SUPPORT: **https://support.twitter.com/articles/20169906**

PRIVACY POLICY: **https://twitter.com/privacy**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/com.atebits.tweetie.application-important-state/	*	Various
/Documents/com.atebits.tweetie.application-state/	app.acct.<Twitter_Username>	Plist
/Documents/com.atebits.tweetie.compose.attachments/	*	JPG
/Documents/com.atebits.tweetie.direct-message.attachments/	*	Various
/Library/Caches/com.atebits.Tweetie2/fsCachedData/	*	Various
/Library/Caches/com.atebits.tweetie.direct-message.cache/	<User_ID>-<User_Number>	Plist
/Library/Caches/com.twitter.simple.disk.caches/	*	MP4
/Library/Caches/TIPImagePipeline/	*	PNG
/Library/Preferences/	com.atebits.Tweetie2.plist	Plist

/private/var/mobile/Containers/Shared/AppGroup/<APP_GUID>

Internal App Path	File Name	File Type
/com.atebits.tweetie.scribe/	Scribe.2.sqlite-sqlite	SQLite
/Library/Preferences/	group.com.atebits.Tweetie2.plist	Plist
/TFSModelCache.1/<User_ID>/database/	modelCache.sqlite3	SQLite

REFERENCES:

http://cs.lewisu.edu/mathcs/msisprojects/papers/kevinswartz.pdf

https://www.academia.edu/10726810/Social_Media_Forensics_on_Mobile_Devices

https://www.tandfonline.com/doi/abs/10.1080/00450618.2015.1066854?src=recsys&journalCode=tajf20

Shopping

Amazon Shopping

APPSTORE URL: **https://apps.apple.com/us/app/amazon-shopping/id297606951**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Caches/com.amazon.Amazon/	Cache.db	SQLite
/Library/Caches/com.amazon.Amazon/fsCachedData/	*	Various
/Library/Caches/WebKit/NetworkCache/Version 14/	*	Various
/Library/Preferences/	com.amazon.Amazon.plist	Plist
/Library/WebKit/WebsiteData/LocalStorage/	https_www.amazon.com_0.localstorage	SQLite

Health & Fitness

Fitbit

APPSTORE URL: **https://apps.apple.com/us/app/google-translate/id414706506**

APPSTORE URL: **https://apps.apple.com/us/app/fitbit-health-fitness/id462638897**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	fitbit.sqlite	SQLite
/Library/Application Support/Fitbit/Defaults/	UserInfo.plist	Plist
/Library/Caches/com.fitbit.FitbitMobile/fsCachedData/	*	JSON
/Library/Preferences/	com.fitbit.FitbitMobile.plist	Plist

Runtastic

APPSTORE URL: **https://apps.apple.com/us/app/adidas-running-app-runtastic/id336599882**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/raw_traces/<User_ID>/	*	CSV
/Library/Application Support/runtastic/	RTCoreDataAdditionalSessionInfo.sqlite	SQLite
/Library/Application Support/runtastic/	RTCoreDataGeoImageInfo.sqlite	SQLite
/Library/Application Support/runtastic/	RTCoreDataHeartRateInfo.sqlite	SQLite
/Library/Application Support/runtastic/	RTCoreDataLiveTrackingInfo.sqlite	SQLite
/Library/Application Support/runtastic/	RTCoreDataLocationInfo.sqlite	SQLite
/Library/Application Support/runtastic/	RTCoreDataRoute.sqlite	SQLite
/Library/Application Support/runtastic/	RTCoreDataSession.sqlite	SQLite
/Library/Application Support/runtastic/	RTCoreDataSpeedInfo.sqlite	SQLite
/Library/Application Support/runtastic/	RTCoreDataUser.sqlite	SQLite
/Library/Application Support/runtastic/	RTDatabaseEventTrace.sqlite	SQLite
/Library/Application Support/runtastic/	RTDatabaseGoalInfo.sqlite	SQLite
/Library/Application Support/runtastic/	RTDatabaseStepInfo.sqlite	SQLite
/Library/Application Support/runtastic/	RTDatabaseWorkout.sqlite	SQLite
/Library/Preferences/	at.runtastic.gpssportapp.plist	Plist

Strava

APPSTORE URL: **https://apps.apple.com/us/app/strava-run-ride-swim/id426826309**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Application Support/	Strava.sqlite	SQLite
/Library/Application Support/	Strava.sqlite.error	SQLite
/Library/Preferences/	com.strava.stravaride.plist	Plist

REFERENCES:

https://deepsec.net/docs/Slides/2019/Still_Secure_We_Empower_What_We_Harden_Because_We_Can_Conceal_-_Yury_Chemerkin.pdf

Finance

Paypal

APPSTORE URL: **https://apps.apple.com/us/app/paypal/id283646709**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Library/Preferences/	com.yourcompany.PPClient.plist	Plist

Venmo

APPSTORE URL: **https://apps.apple.com/us/app/venmo/id351727428**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	Model.sqlite	SQLite
/Documents/	mParticle28.db	SQLite
/Library/Caches/com.hackemist.SDImageCache/default/	*	Various
/Library/Preferences/	net.kortina.labs.Venmo.plist	Plist
/SplashBoard/Snapshots/	*	KTX

REFERENCES: **https://thebinaryhick.blog/2019/11/07/venmo-the-app-for-virtual-ballers**

Reference

Google Translate

APPSTORE URL: **https://apps.apple.com/us/app/google-translate/id414706506**

/private/var/mobile/Containers/Data/Application/<APP_GUID>

Internal App Path	File Name	File Type
/Documents/	translate.db	SQLite
/Library/Preferences/	com.google.Translate.plist	Plist

SANS DFIR Windows to Unix Cheat Sheet v2.0

This guide was created by SANS Internet Storm Center | isc.sans.edu

Automated Malware Analysis

Virustotal: **http://www.virustotal.com**

Jotti s Malware Scan: **http://virusscan.jotti.org**

Threat Information and Internet Status

Shadowserver: **http://www.shadowserver.org**

National Vulnerability Database: **http://nvd.nist.gov**

Mitre CVE: **http://cve.mitre.org**

Useful Tools

Sysinfo CLSIDs/Startup Info: **www.processlibrary.com**

Services to Block Malicious Web Sites

Siteadvisor	**http://www.siteadvisor.com**
Linkscanner	**http://linkscanner.explabs.com**
OpenDNS	**http://www.opendns.org**

Unix Commands

`cut -f 1 -d ' '`	cut the first column from a file. Consider space ( ) as delimiter
`df`	check free disk space
`du`	list disk usage of each directory
`find`	find files
`grep -a "needle" file`	same, but treat binary file like ASCII
`grep -c "needle" file`	count lines that contain "needle"
`grep -v "needle" file`	return lines that DO NOT contain "needle"
`grep "needle" file`	extract all lines that contain "needle" from file
`history`	review recent commands
`id`	who am I?
`ls`	directory listing
`sort`	sort a file line by line
`sort -u`	sort, and only show unique lines
`strings`	extract printable strings from file
`uniq`	unique lines from sorted file

Bring Out the Big Guns

Top 10 sources from a tcpdump capture:

```
tcpdump -nr dumpfile | cut -f3 -d' ' | cut -f1-4 -d'.' |
sort | uniq -c | sort -nr | head -10
```

Top 10 referrers from Apache access log:

```
cat access_log | cut -d' ' -f11 | cut -d '/' -f3 | sort |
uniq -c | sort -nr | head -10
```

Windows Commands

(Not all commands are available in all windows versions) runas: run a program as a different user

assoc	associate programs with extensions
netsh	Network configuration tool (type "?" to get help)
sc	control services (startup/shutdown/auto-start on boot)
sigverify	Verify driver signatures
find large files	find . -size +5000k -xdev -print
Kill Process	wmic process [pid] delete wmic process where name='cmd.exe' delete
List local users	wmic useraccount list full
Startup Programs	wmic startup list full

use wmic on remote system: wmic /user:userID /password:password / node:hostname share list full

More: http://blogs.technet.com
http://isc.sans.org/diary.html?storyid=1229

Frequently Scanned Ports

Also see **http://isc.sans.org/port.hml?port=[portnumber]**

PORT	SERVICE
21	FTP
23	telnet
25	SMTP (Sendmail)
53	DNS
67, 68	DHCP
80	WEB
113	authd/ident
135	Windows Name Resolution
137	Windows File Sharing
139	Windows File Sharing
143	IMAP
161	SNMP
443	HTTPS
445	Common Internet File System
1026	Windows RPC
1027	Windows RPC

PORT	SERVICE
1028	Windows RPC
1080	Proxy Server
1433	SQL Server (connect)
1434	SQL Server (Slammer Worm)
2967	Symantec System Center
3128	Squid Proxy
3306	MYSQL
3389	MS Terminal Services
4662	eMule P2P file sharing
4672	remote access server
4899	remote admin
5060	SIP (VoIP)
5900	VNC
6881	Bittorrent
8000	irdmi/http
8010	Wingate / HTTP

IPv4 Header

Byte 0								Byte 1								Byte 2								Byte 3							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Version				Length				TOS								Total Packet Length															
IP ID / Fragment ID																X	DF	MF	Fragment Offset											
TTL								Protocol								Checksum															
Source Address																															
Destination Address																															

IPv6 Header

Byte 0								Byte 1								Byte 2								Byte 3							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Version				Traffic Class								Flow Label																				
Payload Length																Next Header								Hop Limit							
Source Address (128 bits, 16 bytes)																															
Destination Address (128 bits, 16 bytes)																															

ASCII/HEX Lookup

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SPC	!	“	#	$	%	&		(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

HEX/DEC Conversion

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Protocols

1	ICMP	51	AH
2	IGMP	55	IP Mobility
4	IP in IP	58	IPv6 ICMP
6	TCP	60	IPv6 no header
8	EGP	88	EIGRP
9	IGP	89	OSPFIGP
17	UDP	111	IPX in IP
41	IPv6 in IPv4	115	L2TP
43	IPv6 Routing	133	FC
44	IPv6 Fragment	135	Mobility
47	GRE	255	Reserved
50	ESP		

http://www.iana.org/assignments/protocol-numbers

Compare Unix to Windows Shell Commands

DOS	Unix	
`assign`	`ln -s`	create symbolic link
`attrib`	`chmod`	change file permissions
`chdir`	`pwd`	display current directory
`chdisk`	`du`	disk usage
`cls`	`clear`	clear screen
`comp /fc`	`diff`	compare files
`copy`	`cp`	
`date/time`	`date`	show date or time
`del/erase`	`rm`	delete file
`dir`	`ls`	directory listing
`doskey /h`	`history`	get recent commands
`find`	`grep`	find matching lines
`format`	`mke2fs`	format a disk
`help`	`man`	get help
`ipconfig`	`ifconfig`	check network configuration
`mem`	`free/top`	check memory usage
`mkdir/md`	`mkdir`	create directory
`netstat`	`netstat`	network info
`pkzip`	`zip`	compress a file with ZIP
`reboot`	`shutdown -r now`	reboot system
`rename/move`	`mv`	rename file or move file
`route print`	`route -n`	check routing configuration
`tasklist`	`ps`	list of running programs
`tracert`	`traceroute`	trace route
`type`	`cat`	print file content
`ver`	`uname -a`	OS version

http://www.yolinux.com/TUTORIALS/unix_for_dos_users.html

Convert Windows text files to Unix (remove CR and ^Z):

```
tr -d '\15\32' < windows.txt > unix.txt
```

Convert Unix text file to Windows:

```
awk 'sub("$", "\r")' unix.txt > windows.txt
```

Note: Unix variation vary. Not all commands are available on all flavors of Unix. Use “man” and “info” to explore.

Outshare the bad guys!

Submit your observations at **https://isc.sans.org/contact.html**

SANS DFIR Memory Forensics Cheat Sheet v2.1

This guide was created by by Chad Tilbury | http://forensicmethods.com

Purpose

This cheat sheet supports the SANS **FOR508: Advanced Incident Response, Threat Hunting, and Digital Forensics** course. It is not intended to be an exhaustive resource for Volatility™ or other highlighted tools. Volatility™ is a trademark of Verizon. The SANS Institute is not sponsored, approved by or affiliated with Verizon.

How To Use This Document

Memory analysis is one of the most powerful tools available to forensic examiners. This guide hopes to simplify the overwhelming number of available options.

Analysis can generally be accomplished in six steps:

1. Identify Rogue Processes
2. Analyze Process DLLs and Handles
3. Review Network Artifacts
4. Look for Evidence of Code Injection
5. Check for Signs of a Rootkit
6. Extract Processes, Drivers, and Objects

We outline the most useful Volatility™ plugins supporting these six steps here. Further information is provided for:

- Memory Acquisition
- Alternate Memory Locations
- Converting Hibernation Files and Crash Dumps
- Memory Artifact Timelining
- Registry Analysis Plugins

Getting Started with Volatility™

Getting Help

```
# vol.py -h                Show options and supported plugins
# vol.py plugin -h         Show plugin usage
# vol.py plugin --info     Show available OS profiles
```

Sample Command Line

```
# vol.py -f image --profile=profile plugin
```

Identify System Profile

imageinfo Display memory image metadata

```
# vol.py -f mem.img imageinfo
```

Using Environment Variables

Set name of memory image Takes place of **-f**

```
# export VOLATILITY_LOCATION=file:///images/mem.img
```

Set profile type Takes place of **--profile=**

```
# export VOLATILITY_PROFILE=Win10x64_14393
```

Identify Rogue Processes

pslist High-level view of running processes

```
# vol.py pslist
```

psscan Scan memory for EPROCESS blocks

```
# vol.py psscan
```

pstree Display parent-process relationships

```
# vol.py pstree
```

Identify Rogue Processes

dlllist List of loaded dlls by process

-p Show information only for specific processes (PIDs)

```
# vol.py dlllist -p 1022,868
```

getsids Print process security identifiers

-p Show information only for specific PIDs

```
# vol.py getsids -p 868
```

handles List of open handles for each process

-p Show information only for specific PIDs

-t Display only handles of a certain type {Process, Thread, Key, Event, File, Mutant, Token, Port}

```
# vol.py handles -p 868 -t File,Key
```

Review Network Artifacts

netscan Scan for TCP connections and sockets

```
# vol.py netscan
```

Note: Use connscan and sockscan for XP systems

Look for Evidence of Code Injection

malfind Find injected code and dump sections

-p Show information only for specific PIDs

-v Verbose: show full paths from three DLL lists

```
# vol.py ldrmodules -p 868 -v
```

hollowfind Detect process hollowing techniques

-p Show information only for specific PIDs

-D Directory to save suspicious memory sections

```
# vol.py hollowfind -D ./output_dir
```

Check for Signs of a Rootkit

psxview Find hidden processes using cross-view

```
# vol.py psxview
```

modscan Scan memory for loaded, unloaded, and unlinked drivers

```
# vol.py modscan
```

apihooks Find API/DLL function hooks

-p Operate only on specific PIDs

-Q Only scan critical processes and DLLS

```
# vol.py apihooks
```

ssdt Hooks in System Service Descriptor Table

```
# vol.py ssdt | egrep -v '(ntoskrnl|win32k)'
```

driverirp Identify I/O Request Packet (IRP) hooks

-r Analyze drivers matching REGEX name pattern

```
# vol.py driverirp -r tcpip
```

idt Display Interrupt Descriptor Table

```
# vol.py idt
```

Extract Processes, Drivers, and Objects

dlldump Extract DLLs from specific processes

- **-p** Dump DLLs only for specific PIDs
- **-b** Dump DLL using base offset
- **-r** Dump DLLs matching REGEX name

```
# vol.py dlldump --dump-dir ./output -r metsrv
```

moddump Extract kernel drivers

- **-b** Dump driver using offset address (from modscan)
- **-r** Dump drivers matching REGEX name
- **--dump-dir** Directory to save extracted files

```
# vol.py moddump --dump-dir ./output -r gaopdx
```

procdump Dump process to executable sample

- **-p** Dump only specific PIDs
- **-o** Specify process by physical memory offset
- **-n** Use REGEX to specify process
- **--dump-dir** Directory to save extracted files

```
# vol.py procdump --dump-dir ./output -p 868
```

memdump Extract every memory section into one file

- **-p** Dump memory sections from these PIDs
- **-n** Use REGEX to specify process
- **--dump-dir** Directory to save extracted files

```
# vol.py memdump --dump-dir ./output -p 868
```

filescan Scan memory for FILE_OBJECT handles

```
# vol.py filescan
```

dumpfiles Extract FILE_OBJECTs from memory

- **-Q** Dump using physical offset of FILE_OBJECT
- **-r** Extract using a REGEX (add -i for case insensitive)
- **-n** Add original file name to output name
- **--dump-dir** Directory to save extracted files

```
# vol.py dumpfiles -n -i -r \\.exe --dump-dir=./
```

svcscan Scan for Windows Service record structures

- **-v** Show service DLL for svchost instances

```
# vol.py svcscan -v
```

cmdscan Scan for COMMAND_HISTORY buffers

```
# vol.py cmdscan
```

consoles Scan for CONSOLE_INFORMATION output

```
# vol.py consoles
```

Memory Acquisition

Remember to open command prompt as Administrator

winpmem

- **-o** Output file location
- **-p** **<path to pagefile.sys>** Include page file
- **-e** Extract raw image from AFF4 file
- **-l** Load driver for live memory analysis

```
C:\> winpmem_<version>.exe -o F:\mem.aff4
C:\> winpmem_<version>.exe F:\mem.aff4 -e
  PhysicalMemory -o mem.raw
```

DumpIt

- **/f** Output file location
- **/s** **<value>** Hash function to use
- **/t** **<addr>** Send to remote host (set up listener with /l)

```
C:\> DumpIt.exe /f F:\mem.raw /s 1
```

Alternate Memory Locations

Hibernation File

Compressed RAM Image; available in Volume Shadow Copies

```
%SystemDrive%\hiberfil.sys
```

Page and Swap Files

```
%SystemDrive%\pagefile.sys
%SystemDrive%\swapfile.sys (Win8+\2012+)
```

Memory Dump

```
%WINDIR%\MEMORY.DMP
```

Converting Hibernation Files and Crash Dumps

imagecopy Convert alternate memory sources to raw

- **-f** Name of source file
- **-o** Dump DLL using base offset
- **--profile** Source OS from **imageinfo**

```
# vol.py imagecopy -f hiberfil.sys -O hiber.
  raw --profile=Win7SP1x64
# vol.py imagecopy -f MEMORY.DMP -O crashdump.
  raw --profile=Win2016x64_14393
```

How To Use This Document

The **timeliner** plugin parses time-stamped objects found in memory images. Output is sorted by:

- Process creation time
- Thread creation time
- Driver compile time
- DLL/EXE compile time
- Network socket creation time
- Memory resident registry key last write time
- Memory resident event log entry creation time

timeliner

- **--output-file** Optional file to write output
- **--output=body** Bodyfile format (also **text,xlsx**)
- **--type=Registry** Extract registry key last write times

```
# vol.py -f mem.img timeliner --output-file out.
  body --output=body --profile=Win10x64
```

Registry Analysis Plugins

hivelist Find and list available registry hives

```
# vol.py hivelist
```

hivedump Print all keys and subkeys in a hive

- **-o** Offset of registry hive to dump (virtual offset)

```
# vol.py hivedump -o 0xe1a14b60
```

printkey Output a registry key, subkeys, and values

- **-K** Registry key path

```
# vol.py ol.py printkey -K "Microsoft\Windows\
  CurrentVersio
```

dumpregistry Extract all available registry hives

- **-o** Extract using virtual offset of registry hive
- **--dump-dir** Directory to save extracted files

```
# vol.py printkey -K "Microsoft\Windows\
  CurrentVersion\Run"
```

hashdump Dump user NTLM and Lanman hashes

```
# vol.py hashdump
```

autoruns Map ASEPs to running processes

- **-v** Show everything

```
# vol.py autoruns -v
```

SANS DFIR Rekall Memory Forensic Framework Cheat Sheet v3.1

Purpose

The Rekall Memory Forensic Framework is a collection of memory acquisition and analysis tools implemented in Python under the GNU General Public License. This cheat sheet provides a quick reference for memory analysis operations in Rekall, covering acquisition, live memory analysis and parsing plugins used in the 6-Step Investigative Process. For more information on this tool, visit **rekall-forensic.com**.

Rekall Memory Forensic Framework

Memory analysis is one of the most powerful investigation techniques available to forensic examiners. Rekall auto-detects the target system's profile, using a repository of more than 100 kernel versions available either online or stored locally.

When launching Rekall, you can run single commands or drop into an interactive session to take advantage of caching, preventing the need to obtain the same data with subsequent plugin runs. This cheat sheet shows command line examples using both techniques for Rekall version 1.5.3+

Getting Started with Rekall

Single Command Example

```
$ rekal -f be.aff4 pslist
```

Starting an Interactive Session

```
$ rekal -f be.aff4
```

Starting an Interactive Session (sends output to specified tool)

```
$ rekal -f be.aff4 --pager=gedit
```

Memory Analysis Basics

GETTING HELP

```
[1] be.aff4 11:14:35> plugins.<tab>
```
(lists plugins applicable for use for this image)
```
[1] be.aff4 11:14:35> pslist?
```
(lists options available for specific plugin)

COMMON OPTIONS IN INTERACTIVE SESSION

describe(<plugin>) Print the output fields of a plugin `verbosity=#`
Specify amount of output (1-10, default=1)
`proc_regex="process name"`
Regex to select process by name `<pid>`
Positional Argument: Filter by process PID
`dump_dir="path to directory"`
Path to output directory
`output="path to output dir\file"`
Required if outputting to file quit
Exit interactive session

IMAGE DETAILS (list OS version, physical layout, uptime)

```
[1] be.aff4 11:14:35> imageinfo
```

ARTIFACT COLLECTOR (Carving for defined artifacts)

```
[] Live (API) 16:52:10> artifact_list
[] Live (API) 16:52:10> artifact_collector
["WMIProcessList","WMILoggedOnUsers","WMIDrivers"],
output_path="c:\\cases\\exercises"
```

Step 1. Enumerating Processes

PSLIST – Enumerate Processes

```
[1] be.aff4 11:14:35> pslist
```

Customize pslist output with efilters

```
[1] be.aff4 11:14:35> describe(pslist)
[1] be.aff4 11:14:35> select
EPROCESS,ppid,process_create_time from pslist()
order by process_create_time
```

PSTREE (WITH VERBOSITY) – List Processes with path and command line

```
[1] be.aff4 11:14:35> describe(pstree)
[1] be.aff4 11:14:35> select _EPROCESS,ppid,cmd,path
from pstree()
```

PEINFO Display detailed process & PE info

```
[1] be.aff4 11:14:35> procinfo <PID>
```

DESKTOPS Enumerate desktops and desktop threads

```
[1] be.aff4 11:14:35> desktops verbosity=<#>
```

SESSIONS Enumerate sessions and associated processes

```
[1] be.aff4 11:14:35> sessions
```

Step 2. Analyze Process DLLs and Handles

DLLLIST – List of loaded dlls by process.
Filter on specific process(es) by including the process identifier <PID> as a positional argument

```
[1] image.img 11:14:35> dlllist [1580,204]
```

THREADS – Enumerates process threads

```
[1] be.aff4 11:14:35> threads proc_regex= "chrome"
```

HANDLES List of open handles for each process Include pid or array of pids separated by commas

`object_types="TYPE"` – Limit to handles of a certain type
{Process, Thread, Key, Event, File, Mutant, Token, Port}

```
[1] image.img 11:14:35> handles 868, object_types="Key"
```

FILES CAN – Scan memory for _FILE_OBJECT handles

```
[1] image.img 11:14:35> filescan output="filescan.txt"
```

DUMPFILES – Extract memory mapped files

```
[1] image.img 11:14:35> dumpfiles 1484,dump_dir="."
```

Step 3. Review Network Artifacts

NETSCAN -Scan for connections and sockets in Vista-Win7

```
[1] memory.aff4 11:14:35> netscan
```

NETSTAT -ID active TCP connections in Vista-Win7

```
[1] memory.aff4 11:14:35> netstat
```

DNS_CACHE- Dumps dns resolver cache

```
[1] memory.aff4 11:14:35> dns_cache
```

Step 4. Look for Evidence of Code Injection

MALFIND analysis	Find injected code and dump sections by VAD
<pid> Positional Argument:	Show information only for specific PIDs
phys_eprocess=	Provide physical offset of process to scan
eprocess=	Provide virtual offset for process to scan
dump_dir=	Directory to save memory sections `[1] be.aff4 11:14:35> malfind eprocess=0x853cf460, dump_dir="/cases"`
LDRMODULES	Detect unlinked DLLs
verbosity=	Verbose: show full paths from three DLL lists `[1] be.aff4 11:14:35> ldrmodules 1936`
MESSAGEHOOKS	Enumerates desktop and thread windows message hooks to aid in spotting SetWindowsHookEx code injection

Step 5. Check for Signs of a Rootkit

PSXVIEW MODSCAN	Find hidden processes using cross-view Scan memory for loaded, unloaded, and unlinked drivers
SERVICES	Enumerates services from in-memory registry hive
SVCSCAN	Scans for _SERVICE_RECORD objects
HOOKS_INLINE	Detects API hooks
eprocess=	Filters by virtual address EProcess
phys_eprocess=	Filters by physical address of EProcess
HOOKS_EAT	Detects Export Address Table hooks `[1] be.aff4 11:14:35> hooks_eat 6764`
HOOKS_IAT	Detects Import Address Table hooks
SSDT	Hooks in System Service Descriptor Table
DRIVERIRP	Identify I/O Request Packet (IRP) hooks
regex="drivername"	Filter on REGEX name pattern
OBJECT_TREE	Tracks named objects `[1] be.aff4 11:15:35> object_tree type_regex=”Driver”`
CALLBACKS	Enumerates registered system event callbacks

Step 6. Dump Suspicious Processes and Drivers

DUMP	Hexdump data starting a specified offset [1] be.aff4 11:14:35> dump <virtual offset>

COMMON OPTIONS FOR EXTRACTION

<pid>	Positional Argument: Filter by process
PID proc_regex= "process name"	Regex to select process by name
offset=	Specify process by physical memory offset
dump_dir=	Directory to save extracted files `[1] be.aff4 11:14:35> dlldump 1004,dump_dir="."`
DLLDUMP	Extract DLLs from specific processes
MODDUMP	Extract kernel drivers `[1] be.aff4 11:14:35> moddump regex="tcipip", dump_dir="/tmp"`
PROCDUMP	Dump process to executable sample `[1] be.aff4 11:14:35> procdump proc_regex="csrss", dump_dir="/tmp"`
MEMDUMP	Dump every memory section into a single file `[1] be.aff4 11:15:35> memdump 1004,dump_dir="./output"`

Windows Memory Acquisition (`winpmem`)

CREATING AN AFF4 (Open cmd.exe as Administrator)

```
C:\> winpmem_<version>.exe -o output.aff4
```

*INCLUDE PAGE FILE

```
C:\> winpmem_<version>.exe -p c:\pagefile.sys -o output.aff4
```

EXTRACTING TO RAW MEMORY IMAGE FROM AFF4

```
C:\> winpmem<version>.exe output.aff4 --export PhysicalMemory -o memory.img
```

EXTRACTING TO RAW USING REKALL

```
$ rekal -f win7.aff4 imagecopy --output-image= "/cases/win7.img
```

Live Windows Memory Analysis

(Open cmd.exe as Administrator)
CREATING LIVE REKALL SESSION VIA MEMORY

```
C:\Program Files\Rekall> Rekal --live
```

CREATING LIVE REKALL SESSION VIA API ANALYSIS

```
C:\Program Files\Rekall> Rekal --live API
```

**LIVE WMI COMMANDS

```
[] Live (API) 16:52:10> wmi "select SID,Disabled from Win32_UserAccount"
```

**LIVE GLOB SEARCH

```
[] Live (API) 16:52:10> select * from glob("c:\windows\*.exe")
```

MacOS Memory Live Analysis & Acquisition

MAC OSXPMEM (Run commands with Root privileges)
Extract osxpmem.zip and ensure file/dir permissions are root:wheel

CREATING AN AFF4

```
$ sudo kextload MacPmem.kext
$ sudo ./osxpmem --output test.aff4
$ sudo kextunload MacPmem.kext/
<clean up by removing driver>
```

LIVE OSX MEMORY ANALYSIS

```
$ sudo kextload MacPmem.kext/
$ rekal -f /dev/pmem
<begin interactive session>
$ sudo kextunload MacPmem.kext/
<clean up by removing driver>
```

Registry Analysis Plugins

ENUMERATE AND EXTRACT REGISTRY HIVES

HIVES	Find and list available registry hives `$ rekal -f be.aff4 hives`
REGDUMP	Extracts target hive
--hive_regex	Regex Pattern Matching
- D ”<dir>”	Dump directory `$ rekal -f be.aff4 regdump --hive_regex="SAM" -D "/cases"`
PRINTKEY	Output a registry key, subkeys, and values
-K	“Registry key path” `[1] be.aff4 11:14:35> printkey -K "Software\Microsoft\Windows\CurrentVersion\Run"`
USERASSIST	Find and parse userassist key values

Additional Functionality

ANALYZE_STRUCT	Interprets and identifies windows memory structures when given a virtual offset `[1] be.aff4 11:15:35> analyze_struct 0x8180e6f0`
DT	Displays Specific Kernel Data Structures `[1] be.aff4 11:14:35> dt "_EPROCESS",offset=<virtual offset>`
PTOV	Determine owning process with physical to virtual address translation (decimal offset shown below) `$ rekal -f test.img ptov 21732272`
VMSCAN	Allows for the identification of virtual machines
CERTSCAN	Dumps RSA private and public keys
dump_dir=	Dumps output to a specified directory
MIMIKATZ	Extracts and decrypts credentials from lsass

Linux Memory Acquisition

LINUX PMEM (TO CREATE PROFILE)

```
# tar vxzf linux_pmem_1.0RC1.tgz
# cd linux
# make
```

LINPMEM (TO CREATE IMAGE VIA /proc/kcore)

```
# gzip -d linpmem_2.0.1.gz
# chmod 755 linpmem_2.0.1
# ./linpmem_2.0.1 -o linux.aff4
# cd linux
# rekal convert_profile 3.11.0-26-generic.zip
Ubuntu.zip # rekal --profile=Ubuntu.zip -f ../linux.aff4
```

SANS DFIR Hex File Headers and Regex for Forensics Cheat Sheet v1.0

This guide was created by by Guy Bruneau | gbruneau@sans.org

Purpose

Forensic analysts are on the front lines of computer investigations and this guide aims to support them in their quest to uncover the truth.

How To Use This Document

When performing an investigation, it is helpful to be reminded of the powerful options available to the investigator. This document is aimed to be a reference to the tools that could be used.

The key to successful forensics is minimizing your data loss, accurate reporting, and a thorough investigation.

awk

awk is an extremely useful tool, especially for parsing data structured in columns. It is straightforward to use for simple purposes. Its basic use is to select some particular columns from the output: column 1 is referred to as $1, column 2 as $2, etc.

The space is the default awk separator. However if you want to be able to parse data separated by some other character, e.g., "**:**", you can use the **-F** flag.

Example: **`echo "hello:goodbye" | awk -F: '{print $2}'`**

Would return "**goodbye**" as an output

sed

sed is an excellent command for character substitution. Example: if you want to substitute the first occurrence of the '**a**' character by an '**e**':

```
echo "hallo" | sed 's/a/e/'
```

The output would be: **`hello`**

You can use the **g** modifier to substitute all instances:

```
echo "Hallo Janny" | sed 's/a/e/g'
```

The output would be: **`Hello Jenny`**

uniq

The uniq command reads the input and compares adjacent lines. If two or more adjacent lines are identical, all but one is removed.

Here is a list of the most common options used with uniq:

-c	Prefix line with number of occurrence
-f	Avoid comparing the first N fields
-i	Ignore case
-s	Avoid comparing the first N characters
-u	Only print unique lines

Consider this input file:

```
a
b
c
d
```

Now run uniq on it:
sort testfile | uniq

```
a
b
c
```

Now run uniq -c on it:

```
1 a
2 b
3 c
```

Date

Check the date man page for more options.

Returns the real date from epoch time:
`date –d @1284127201`

Return an epoch time of 1288756800:
`date +%s -d "2010-11-03"`

Return a 2 days old date:
`date --date="-2 days"  +"%Y-%m-%d"`

Return 20:00 hours:
`date -d @1288310401 +%k:%M`

Windows findstr

The Windows findstr has one interesting feature that differs from grep. If you need to search for multiple strings, you need to separate them with a space.

For example, you want or need to look for a match for WHITE or GREEN in a text file, you write your command like this:

```
findstr "WHITE GREEN" textfile
```

To make the search case insensitive, add the /I to print all variant of WHITE or GREEN.

Windows findstr Command List

/B	Matches pattern if at the beginning of a line
/E	Matches pattern if at the end of a line
/L	Uses search strings literally
/R	Uses search strings as regular expressions
/S	Searches for matching files in the current directory and all subdirectories
/I	Specifies that the search is not to be case-sensitive
/X	Prints lines that match exactly
/V	Prints only lines that do not contain a match
/N	Prints the line number before each line that matches
/M	Prints only the filename if a file contains a match
/O	Prints character offset before each matching line
/P	Skip files with non-printable characters

Hex File Header and ASCII Equivalent

File headers are used to identify a file by examining the first 4 or 5 bytes of its hexadecimal content.

File Type	Start	Start ASCII Translation
ani	52 49 46 46	RIFF
au	2E 73 6E 64	snd
bmp	42 4D F8 A9	BM
bmp	42 4D 62 25	BMp%
bmp	42 4D 76 03	BMv
cab	4D 53 43 46	MSCF
dll	4D 5A 90 00	MZ
Excel	D0 CF 11 E0	
exe	4D 5A 50 00	MZP (inno)
exe	4D 5A 90 00	MZ
flv	46 4C 56 01	FLV
gif	47 49 46 38 39 61	GIF89a
gif	47 49 46 38 37 61	GIF87a
gz	1F 8B 08 08	
ico	00 00 01 00	
jpeg	FF D8 FF E1	
jpeg	FF D8 FF E0	JFIF
jpeg	FF D8 FF FE	JFIF
Linux bin	7F 45 4C 46	ELF
png	89 50 4E 47	PNG
msi	D0 CF 11 E0	
mp3	49 44 33 2E	ID3
mp3	49 44 33 03	ID3
OFT	4F 46 54 32	OFT2
PPT	D0 CF 11 E0	
PDF	25 50 44 46	%PDF
rar	52 61 72 21	Rar!
sfw	43 57 53 06/08	cws
tar	1F 8B 08 00	
tgz	1F 9D 90 70	
Word	D0 CF 11 E0	
wmv	30 26 B2 75	
zip	50 4B 03 04	PK

grep/egrep

grep's strength is extracting information from text files. grep operates on one or multiple files when provided with a command line argument(s) that can also include wildcards:

Example: **`grep "John" addressbook`**
Would return the lines that contained the **`"John"`** string in the addressbook text file

Some useful flags:

- **`-A`** Print number of lines after the match
- **`-B`** Print number of lines before match
- **`-c`** Report number of occurrences
- **`-f`** Reads one or more patterns from a file – pattern is terminated by a newline
- **`-h`** Suppress the file names on the output
- **`-i`** Ignore case
- **`-l`** Report matching files, not matching lines
- **`-P`** Interpret pattern as a Perl Regex
- **`-v`** Reverse operation: return the lines **not** matching the string

The **`egrep`** (extended **`grep`**) utility can be useful to match several possible strings at the same time (in an OR mode):

```
egrep "John|Peter" addressbook
grep "John\|Peter" addressbook
```

SANS DFIR Eric Zimmerman's Tools Cheat Sheet v1.0

This cheat sheet covers the basics of using several command line programs by Eric Zimmerman.

Purpose

Incident responders are on the front lines of intrusion investigations and this guide aims to support them in their quest to uncover the truth.

IT'S TIME TO GO HUNTING!

Lnk files with LECmd

Type of artifact: Document creation and opening

Basic usage

```
LECmd.exe -f <file>
LECmd.exe -d <directory>
```

Key data

Target timestamps, Volume information, Absolute file path, Target ID information

Advanced usage

Use the **--all** switch to process all files in a directory vs. only those ending in '**.lnk**'.

Prefetch Files with PECmd

Type of artifact: Evidence of execution

Basic usage

```
PECmd.exe -f <file>
PECmd.exe -d <directory>
```

Default output is to standard out. Data can be exported to several formats such as csv, json, HTML, etc.

```
PECmd.exe -d <directory> --csv c:\temp
```

Key data

Execution timestamps, total number of executions, and files/directories referenced

Advanced usage

To display higher precision timestamps, use the **--mp** switch. When **--mp** is used, the higher precision timestamps will be reflected in any exported data as well.

Jump Lists with JLECmd

Type of artifact: Document creation and opening

Basic usage

```
JLECmd.exe -f <file>
JLECmd.exe -d <directory>
```

Key data

Same as **LECmd** key data plus Application ID and DestList entry information (for automaticDestinations jump lists)

Advanced usage

The **--ld** and **--fd** switch can be used to display more information about each embedded lnk file.

Use the **--appIDs** switch to supply a list of application IDs that will be added to the internal list of over 375 appIDs.

In some cases, an automaticDestinations jump list can contain additional lnk files tracked in its Directory that are not accounted for in DestList entries. When this happens, a warning will be given and the **--withDir** switch can be used to process all available lnk files regardless of them being present in the DestList.

JLECmd also allows for exporting out all available lnk files from a jump list to a directory via the **--dumpTo** switch. Once lnk files have been dumped from a jump list, they can be investigated using any parser that understands lnk files (**LECmd** for example).

String Searching with bstrings

Type of artifact: Any

Basic usage

```
bstrings -f <file>
```

To search for specific strings, use **--ls**

```
bstrings -f <file> --ls "forensics"
```

Use the **-x** and **-m** switches to set maximum and minimum string lengths

Use **--off** to show the offset for each search hit

Advanced usage

In addition to Unicode strings, **bstrings** looks for strings encoded using Western (1252) code page. Use the **--cp** switch to search in any other code page supported by .net. A full listing of available code pages is available at **https://goo.gl/ig6DxW**

bstrings also supports regular expression searches via the **--lr** switch. **bstrings** also contains over a dozen built in regular expression patterns for things like credit card numbers, social security numbers, IP addresses, email addresses, and more. To see a list of all built-in regular expressions, use the **-p** switch. When using a built-in expression, use the value in the Name column. For example, to look for email addresses, use this command:

```
bstrings -f <some file> --lr email
```

bstrings also allows searching for several strings or regular expressions at once using the **--fr** and **--fs** switches

Filtering: Just the Field You Want

Type of artifact: Evidence of execution

Basic usage

```
AppCompatCacheParser.exe -f <path to>\SYSTEM --csv
c:\temp
```

Output file is a tab separated file that can be imported into Timeline Explorer or Excel.

Key data

Full path for executable and execution flag

Advanced usage

```
AppCompatCacheParser.exe --csv c:\temp
```

Omitting **-f** switch pulls AppCompatCache data from the Registry hive loaded into memory.

The **-d** switch can be used to inspect all available details for an entry

Common Functionality

Most tools have common options for exporting data, displaying higher precision timestamps, using custom date formats, etc.

When **--mp** is used, higher precision timestamps are displayed and will also be reflected in any exported data.

Data can be exported to several formats such as csv, json, HTML, etc. at the same time.

```
PECmd.exe -d <directory> --csv c:\temp --html
c:\temp\html
```

Amcache.hve with `AmcacheParser`

Type of artifact: Evidence of execution

Basic usage

```
AmcacheParser.exe -f <path to>\Amcache.hve –csv c:\temp
```

Output file is a tab separated file that can be imported into Timeline Explorer or Excel.

Key data

FullPath: The full path to the executed file

SHA-1: The SHA-1 hash of the file FileIDLastWriteTimestamp: First executed timestamp MFTEntryNumber: NTFS entry number from FILE record

MFTSequenceNumber: NTFS sequence number from FILE record

Advanced usage

Use the **`-b`** and **`-w`** switches to blacklist or whitelist SHA-1 hashes to further reduce the amount of data to review

Use **`-i`** to generate a list of associated program/file entries

Download Location

Individual tools are available at **https://ericzimmerman.github.io/**.

Chocolatey packages for each are also available.

To get all tools at once, use chocolatey to install the EricZimmermanTools package

SANS DFIR JSON and jq Quick Start Guide v1.2

This guide was created by Phil Hagen & David Szili — lewestech.com | alzetteinfosec.com

Purpose

This guide is a supplement to the SANS **FOR572: Advanced Network Forensics: Threat Hunting, Analysis, and Incident Response** course. It covers the basics of JSON and some of the fundamentals of the **jq** utility. The **jq** utility filters, parses, formats, and restructures JSON—think of it as **sed**, **awk**, and **grep**, but for JSON. Given the trend toward logs being generated in JSON, easily accessing and molding that data is increasingly important for the forensicator. This document is not intended to replace **jq**'s extensive documentation. It is only a quick reference resource.

Acquiring the jq Utility

Stephan Dolan is the developer of **jq**, which is free and open-source software. You can install **jq** in most *NIX-based operating systems using the distribution's software/package management system, or you can download the binary for your operating system or the source code from the main project page. Visit **for572.com/jq** for details, as well as to learn how to contribute to the project through the author's GitHub repository.

A web-based version of **jq** is hosted at **jqplay.org**. This is especially helpful while learning **jq**, as you can experiment with filters and options in a graphical interface. The **jqplay.org** web implementation can also be hosted in your own environment, making it suitable for air-gapped networks.

JSON: Structure for Humans and Machines

JSON (JavaScript Object Notation) is an open standard data interchange format, which is both machine-parsable and (mostly) human-readable. JSON is built on two primary structures: Key/value pairs and ordered lists of values. Data types for values can consist of strings, numbers, booleans, nested JSON objects, or the null value.

A single-field JSON object might look like this:

```
{ "name": "Lance" }
```

A more complex JSON object might look like this:

```
{ "name": "Lance", "age": 42,
  "active": true, "tags": [ "tag1",
  "tag2" ], "address": { "street":
  "123 Main", "city": "Lewes",
  "state": "DE", "postalCode":
  "19958"}, "pet": null }
```

JSON can be represented in compact form, as shown above with one object per line, or expanded as shown below – both formats are considered equivalent. Note that **tags** is an array and **address** is a nested JSON object.

```
{
  "name": "Lance",
  "age": 42,
  "active": true,
  "tags": [
    "tag1",
    "tag2"
  ],
  "address": {
    "street": "123 Main",
    "city": "Lewes",
    "state": "DE",
    "postalCode": "19958"
  },
  "pet": null
}
```

Sample JSON Record

The following JSON object will be used for all examples. This reflects a single entry from a **dns.log** file created by the **Zeek Network Security Monitoring** platform. All examples assume the compact version of this record in a file named dns.log. If you'd like to test on your own, download the sample record from **for572.com/dnslog-sample**.

```
{
  "ts": 1602265824.123071,
  "uid": "CHFRflzsgM15k9et4",
  "id.orig_h": "192.168.75.169",
  "id.orig_p": 58506,
  "id.resp_h": "192.168.75.1",
  "id.resp_p": 53,
  "proto": "udp",
  "trans_id": 50763,
  "rtt": 0.022633075714111328,
  "query": "www.sansgear.com",
  "qclass": 1,
  "qclass_name": "C_INTERNET",
  "qtype": 1,
  "qtype_name": "A",
  "rcode": 0,
  "rcode_name": "NOERROR",
  "AA": false,
  "TC": false,
  "RD": true,
  "RA": true,
  "Z": 0,
  "answers": [
    "vhost1.identityvector.com",
    "70.32.97.206"
  ],
  "TTLs": [
    3600,
    3600
  ],
  "rejected": false
}
```

To learn what these fields mean, see **for572.com/dnslog-fields**

Fundamental Usage: Pretty Print

In its simplest usage, jq will format compact JSON objects into their expanded form, as shown in the panels to the left. There are several fundamental command line options that will help you as well. The **'.'** filter represents the root of each JSON object and will simply display all fields in the object when used.

jq Format, filter, and transform JSON data

```
$ jq [options] <filter> <input_file>
```

-c Compact output instead of "pretty-printed"

-r Raw output instead of quoted JSON text

-s Sort output lexically based on key names

For example, to print the sample **dns.log** entry shown in expanded form on the left:

```
$ jq '.' dns.log
```

Filtering: Just the Field You Want

Many times, the user only needs to display specific fields from each JSON object instead of the entire set. This requires a more detailed filter statement.

To display the value for just one field, identify the field with the **'.<fieldname>'** syntax (note the leading dot).

```
$ jq '.query' dns.log
"www.sansgear.com"
```

To display resulting values in their non-quoted raw form, use the **-r** option to the **jq** command.

```
$ jq -r '.query' dns.log
"www.sansgear.com"
```

When referencing a field name that contains any non-alphanumeric character, double quotation marks must be used. This is common with the dot character, which designates nested JSON objects. However, some JSON logs such as Zeek's use it as a part of the field name which then requires double quoting.

```
$ jq '."id.orig_h"' dns.log
"192.168.75.169"
```

Accessing Array Elements

To access a particular element from an array, familiar **[<element_number>]** notation is used. Remember that array indices are zero-based.

Accessing Nested JSON Objects

Nested objects can be accessed by using the dot separator. While the sample Zeek **dns.log** entry does not contain these, the below example uses the original JSON object from this handout with the shell's pipe operator to show that like many other command-line tools, jq can be used with data on STDIN instead of a file.

```
$ echo '{ "name": "Lance", ⏎
    "age": 42, "active": true, ⏎
    "tags": [ "tag1", "tag2" ], ⏎
    "address": { "street": "123 Main", ⏎
    "city": "Lewes", "state": "DE", ⏎
    "postalCode": "19958"}, ⏎
    "pet": null }' | ⏎
  jq '.address.city'
"Lewes"
```

Complex Filtering: Build New JSON Objects

To select more than one field, the syntax reflects assembling a new JSON object. No leading dot is used in this filter syntax. Note that the **-r** option has no effect.

```
$ jq '{ "id.orig_h", query }' dns.log
{
  "id.orig_h": "192.168.75.169",
  "query": "www.sansgear.com"
}
```

Selecting Records Based on Content

By default, jq will process all JSON objects in the input data set. Using the select operator allows you to limit the records processed based on their values.

```
$ jq 'select(.rcode_name == "NOERROR")'
```

This command will produce all records with a value of "**NOERROR**" in the **rcode_name** field.

The additional operators **contains**, **startswith**, and **endswith** are also useful ways to select records.

```
$ jq 'select(.query | ⏎
   contains("sans"))' dns.log
$ jq 'select(.query | ⏎
  endswith(".com"))' dns.log
```

Note however, that the **select()** operator can be very slow, especially on large data sets. Using a preprocessor such as grep (or **zgrep** for **gzip**-compressed JSON) can provide a dramatic performance improvement.

Chaining jq Operations

The pipe symbol, **|**, can be used in the filter statement to pass the output of one operation to the input of the next.

```
$ jq 'select(.rcode_name == "NOERROR") ⏎
  | .query' dns.log
"www.sansgear.com"
```

Reformatting Time Stamps

Many logs use a form of the UNIX Epoch timestamp. Rather than use external conversion utilities, **jq** can convert these natively. (The **|** operator passes output from one part of the filter as input to the next and **|=** replaces a field's value in place.)

```
$ jq '.ts |= todate | .ts' dns.log
"2020-10-09T17:50:24Z"
```

SANS DFIR Linux Shell Survival Guide v2.3

This guide was created by Phil Hagen — lewestech.com

Purpose

This guide is a supplement to SANS FOR572: Advanced Network Forensics: Threat Hunting, Analysis, and Incident Response. It covers some of what we consider the more useful Linux shell primitives and core utilities. These can be exceedingly helpful when automating analysis processes, generating output that can be copied and pasted into a report or spreadsheet document, or supporting quick-turn responses when a full tool kit is not available.

Remember: If you can make it happen in a shell over a lag-ridden SSH connection, there is a better chance of being the lethal forensicator when it really matters!

How To Use This Document

Linux has been around since 1991, and its *NIX parents since 1969. This handout cannot begin to scratch the surface of the great and powerful things you can do with nothing more than a shell prompt and some moxie. Use this document as a "memory jog" for some of the capabilities of the more commonly used tools in this course and in the forensic workflow in general.

Dig into the details of each tool's features through its manual pages (aka "man pages") and other online and offline references. We think you will find the shell to be as powerful as the GUI, and in some cases a far superior alternative – especially for scalability and automation.

No Packets, No Party

tcpdump – Dump network traffic

```
$ sudo tcpdump -n -s 0 -i eth0 '<BPF filter>'
$ tcpdump -n -r input.pcap -w output.pcap '<BPF filter>'
```

- **-n** Prevent DNS lookups on IP addresses. Use twice to also prevent port-to-service lookups
- **-r** Read from pcap file instead of the network
- **-w** Write packet data to a file
- **-D** Enumerate network interfaces
- **-i** Specify the network interface on which to capture
- **-s** Number of bytes per packet to capture
- **-C** Number of megabytes to save in a capture file before starting a new file
- **-G** Number of seconds to save in each capture file (requires time format in output filename)
- **-W** Used with the -C or -G options, limit the number of rotated files (see man page for detailed usage)
- **-x** Display packet contents in hex

tcpdump *requires root privileges to capture network traffic promiscuously. User-level permissions are sufficient for manipulating existing capture files.*

See the ***pcap-filter*** *man page for information on building BPFs to control captured traffic.*

Index That pcap File!

nfpcapd – Generate **nfdump**-compatible NetFlow records from **pcap** file

```
$ nfpcapd -r in.pcap -l ./netflow/ -S 1 -z
```

- **-r** pcap file to read
- **-l** Output directory
- **-s** Output directory format (0=flat, 1=yr/mo/day; see **nfcapd** man page for more)
- **-z** Compress output flows

Fundamental Usage: Pretty Print

nfdump – Process NetFlow data from files on disk

```
$ nfdump -R ./ -O tstart -o extended
```

- **-R** Recursively read data from the specified directory
- **-r** Read data from a single nfcapd file
- **-a** Aggregate by src+dst IP, src+dst port, protocol
- **-A** Specify custom aggregation
- **-t** Time window, in "**YYYY/MM/DD.hh:mm:ss**" format (See man page for additional details)
- **-s** Generate "TopN" statistics
- **-O** Specify output ordering
- **-o** Specify output format (**line**, **long**, **extended**, or custom). Custom formatting uses "**fmt:<format string>**" syntax, where "**<format string>**" defines values displayed (see man page for full list).

%ts	Start time	**%te**	End time
%td	Duration	**%pr**	Protocol
%sa	Source address	**%da**	Destination address
%sap	Source IP:port	**%dap**	Destination IP:port
%sp	Source port	**%dp**	Destination port
%sas	Source ASN	**%das**	Destination ASN
%pkt	Packet count	**%byt**	Byte count
%fl	Flow count	**%flg**	TCP flags
%bps	Bits per second	**%pps**	Packets per second
%bpp	Bytes per packet		

What's in a Name?

passivedns – Generate normalized records for all DNS queries and responses

```
$ passivedns -r input.pcap ⏎
  -l pdnslog.txt -L pdns_nxdomain.txt
```

- **-r** Specify pcap file to read
- **-l** Log for normal (non-error) queries
- **-L** Log for SRC error queries
- **-i** Specify interface for live DNS observation

GUI-less Packet Spelunking

tshark – Dump and analyze network traffic (aka "Wireshark in the shell")

```
$ tshark -n -r in.pcap -Y '<disp filter>'
```

-n Prevent DNS and port lookups

-r Read from pcap file instead of the network

-w Write output to a pcap file instead of the terminal

-T Output f ormat (text, fields, etc.)

-e With" -T fields", add a field to the output

-Y Protocol-aware display filter to apply

-z Statistical output modes – see man page

*See the **wireshark-filter** man page for information on building protocol-aware display filters.*

Bring Out the Big Guns

awk – Pattern scanning and processing language
Seriously powerful stuff™!

```
$ awk -F ',' '{ print $1,$6,$3 }' in.txt
```

-F Specify input field separator (default is space)

Input and output field separators can be specified in the awk script itself with the FS and OFS variables:

```
$ awk '{ FS = ","; OFS = "\t"; print $2,$4 }' in.txt
```

For More About These Fine Commands...

Use the built-in reference manual:

man Interface to the on-line reference manuals

```
$ man find
```

-k Perform keyword search through all man pages

*Use inline command help where available – many commands provide brief usage statements with the "**--help**" or "**-h**" options*

```
$ tcpdump --help
```

The Grymoire - home for UNIX wizards:

http://www.grymoire.com/Unix/

Use the Force

BASH provides many functions to improve your accuracy, speed, and efficiency – know and use them!

Tab Completion
Hit the <TAB> key to expand the first few characters of a command, directory name, filename, or variable name. If there is more than one possible option, it will complete as far as possible. Press <TAB> again to see the possible completion options.

Standard Variables

~ An alias for the current user's home directory (also available as $HOME)

$PATH The command search path

$? The exit value of the previous command

$PWD The current working directory

Command History
Cycle through previous commands by pressing the up and down arrows. Use the *history* command to see a list of the command history buffer. (BASH writes this buffer is **~/.bash_history** upon exiting, overwriting any existing contents.) Press *Ctrl-R* to search through history for commands that match a search string.

Searching: Where For Art Thou?

grep – Print lines matching a pattern

```
$ grep pattern input.txt
```

-i Case-insensitive pattern matching

-v Print lines that do not match

-c Count matching lines instead of printing them

-l Print filenames containing matching lines

-h Do not include filenames when searching multiple input files (e.g., **output*.txt**)

Working on the Chain Gang

Linux prefers small, single-purpose functions and utilities. Chain them together with the "pipe", which sends the output of one command into the next as input.

```
$ grep pattern input.txt | sort | uniq -c
```

Iteratively build a series of commands to create output that definitively addresses your requirements.

Order in the Court

sort – Sort lines alphabetically or numerically

```
$ sort input.txt
```

-n Sort numerically (5 before 10)

-r Reverse sort order

-k Specify an alternate sort field

-t Specify a field delimiter for **-k**

De-Duplication and De-Duplication

uniq – Only print consecutive matching lines once

```
$ grep pattern input.txt | uniq
```

-c Print the count of consecutive lines

Remember: Only finds consecutive matching lines! Most useful with input piped from the sort command.

```
$ grep pattern input.txt | sort | uniq
```

Redirect Output: I Don't Want to Hear You

Redirect output to a file instead of the shell itself with the "greater than" character. (Warning: It overwrites any existing contents!)

```
$ grep pattern1 input.txt > results.txt
```

Append to existing files with "double greater than"

```
$ grep pattern2 input.txt >> results.txt
```

SANS DFIR SIFT Workstation Cheat Sheet v3.0

http://computer-forensics.sans.org | http://blogs.sans.org/computer-forensics

Purpose

DFIR Forensic Analysts are on the front lines of computer investigations and this guide aims to support them in their quest to uncover the truth.

How To Use This Document

When performing an investigation it is helpful to be reminded of the powerful options available to the investigator. This document is aimed to be a reference to the tools that could be used. Each of these commands runs locally on a system.

TIME TO GO HUNTING

Mounting DD Images

mount -t *fstype [options] image mountpoint*

The *image* can be a disk partition or a dd image file

Useful options:

ro .....mount as read only

loop.....mount on a loop device

noexec .....do not execute files

offset=<BYTES> .....logical drive mount

show_sys_files.....show ntfs metafiles

streams_interface=windows.....use ADS

Example: Mount an image file at mount_location

```
# mount -o
loop,ro,show_sys_files,streams_interface=windows
imagefile.dd /mnt/windows_mount
```

Mounting E01 Images

```
# ewfmount image.E01 mountpoint
```

```
# mount -o
loop,ro,show_sys_files,streams_interface=windows
/mnt/ewf/ewf1 /mnt/windows_mount
```

Mounting Volume Shadow Copies

Stage 1 – Attach local or remote system drive:

```
# ewfmount system-name.E01 /mnt/ewf
```

Stage 2 – Mount raw image VSS:

```
# vshadowmount ewf1 /mnt/vss/
```

Stage 3 – Mount all logical filesystem of snapshot:

```
# cd /mnt/vss
# for i in vss*; do mount -o
ro,loop,show_sys_files,streams_interface= windows $i
/mnt/shadow_mount/$i; done
```

Creating Super Timelines

```
# log2timeline -r -p -z <system-timezone> -f
<type-input> /mnt/windows_mount -w timeline.csv
```

file|dir ..... artifact target

-f *<TYPE-INPUT>*.....input format

-o *<TYPE-OUTPUT>* .....output format: default csv file

-w *<FILE>*.....append to log file

-z *<SYSTEM TIMEZONE>*

-Z *<OUTPUT TIMEZONE>*

-r .....recursive mode

-p .....preprocessors

```
# mount -o
loop,ro,show_sys_files,streams_interface=windows
imagefile.dd /mnt/windows_mount
```

```
# log2timeline -z EST5EDT -p -r -f win7
/mnt/windows_mount -w /cases/bodyfile.txt
```

```
# l2t_process -b /cases/bodyfile.txt -w whitelist.txt
04-02-2012 > timeline.csv
```

Stream Extraction

```
# bulk_extractor <options> -o output_dir image
```

Useful options:

-o outdir

-f <regex> ..... regular expression term

-F <rfile> ..... file of regex terms

-Wn1:n2..... extract words between n1 and n2 in length

-q nn ..... quiet mode

-e scanner ..... enables a scanner

-e wordlist.... enable scanner wordlist

-e aes ..... enable scanner aes

-e net ..... enable scanner net

```
# bulk_extractor -F keywords.txt -e net -e aes -e
wordlist -o /cases/bulk-extractor-memory-output /
cases/memory-raw.001
```

Registry Parsing – Regripper

```
# rip.pl -r <HIVEFILE> -f <HIVETYPE>
```

Useful options:

-r..... Registry hive file to parse <HIVEFILE>

-f..... Use <HIVEFILE> (e.g., **sam, security, software, system, ntuser**)

-l..... List all plugins

```
# rip.pl -r
/mnt/windows_mount/Windows/System32/config/SAM -f sam
/cases/windowsforensics/SAM.txt
```

Recover Deleted Registry Keys

```
# deleted.pl <HIVEFILE>

# deleted.pl
/mnt/windows_mount/Windows/System32/config/SAM  >
/cases/windowsforensics/SAM_DELETED.txt
```

Recovering Data

Create Unallocated Image Using dls (for FAT and NTFS)

```
# blkls imagefile.dd >
unallocated_imagefile.blkls
```

Create Slack Image (deleted data) using **blkls**

```
# blkls -s imagefile.dd >
imagefile.slack
```

foremost Carves out files based on headers and footers

data_file.img = raw data, slack space, memory, unallocated space

```
# foremost -o outputdir -c
/path/to/foremost.conf  data_file.img
```

sigfind Search for a binary value at a given offset (-o)

-o <offset> Start search at byte **<offset>**

```
# sigfind <hexvalue> -o <offset>
```

Shadow Timeline Creation

Step 1 – Attach Local or Remote System Drive

```
# ewfmount system-name.E01 /mnt/ewf
```

Step 2 – Mount VSS Volume

```
# cd /mnt/ewf
# vshadowmount ewf1 /mnt/vss
```

Step 3 – Run fls across ewf1 mounted image

```
# cd /mnt/ewf
# fls -r -m C: ewf1 >> /cases/vss-bodyfile
```

Step 4 – Run fls Across All Snapshot Images

```
# cd /mnt/vss
# for i in vss*; do fls -r -m C: $i >>
/cases/vss-bodyfile; done
```

Step 5 – De-Duplicate Bodyfile using sort and uniq

```
# sort /cases/vss-bodyfile | uniq >
/cases/vss-dedupe-bodyfile
```

Step 6 – Run mactime Against De-Duplicated Bodyfile

```
# mactime -d -b /cases/vss-dedupe-bodyfile -z EST5EDT
MM-DD-YYYY..MM-DD-YYYY > /cases/vss-timeline.csv
```

Memory Analysis

```
vol.py command -f
/path/to/windows_xp_memory.img
--profile=WinXPSP3x86
```

Supported commands:

connscan..... Scan for connection objects

files..... List of open files process

imagecopy ..... Convert hibernation file

procdump..... Dump process

pslist..... List of running processes

sockscan..... Scan for socket objects

Sleuthkit Tools

File System Layer Tools (Partition Information)

fsstat..... Displays details about the file system

```
# fsstat imagefile.dd
```

Data Layer Tools (Block or Cluster)

blkcat..... Displays the contents of a disk block

```
# blkcat imagefile.dd block_num
```

blkls ..... Lists contents of deleted disk blocks

```
# blkls imagefile.dd > imagefile.blkls
```

blkcalc..... Maps between dd images and blkls results

```
# blkcalc imagefile.dd -u blkls_num
```

blkstat..... Displays allocation status of block

```
# blkstat imagefile.dd cluster_number
```

MetaData Layer Tools (Inode, MFT, or Directry Entry)

ils..... Displays inode details

```
# ils imagefile.dd
```

istat ..... Displays information about a specific inode

```
# istat imagefile.dd inode_num
```

icat..... Displays contents of blocks allocated to an inode

```
# icat imagefile.dd inode_num
```

ifind..... Determines which inode contains a specific block

```
# ifind imagefile.dd -d block_num
```

Filename Layer Tools

fls ..... Displays deleted file entries in a directory inode

```
# fls -rpd imagefile.dd
```

ffind ..... Finds the filename that's using the inode

```
# ffind imagefile.dd inode_num
```

SANS DFIR SQLite Reference Guide

Lee Crognale, Sarah Edwards and Heather Mahalik — sans.org/for518 | sans.org/for585

Purpose

This guide is a supplement to the SANS **FOR518: Mac and iOS Forensic Analysis and Incident Response** and **FOR585: Smartphone Forensic Analysis In-Depth** courses, and enhances concepts covered in other courses. It covers some of the core methods to extracting data from SQLite databases. Definitions, sample queries, and SQLite terminology will help you conduct manual extractions from databases of interest found on Macs, smartphones, and PCs. *Commercial tools will only get you so far. Manual recovery and extracting contents **is** what really matters!*

How To Use This Document

SQLite databases are used to store a vast amount of data recovered from digital media. This handout cannot begin to scratch the surface of the great and powerful things you can do with SQL queries. Use this document as a "memory jog" for some of the queries, tools available and examples for normalizing data formats recovered from databases.

We think you will find this reference as a place to start or simply help remember which operators and queries will assist you in your investigation. The tools listed provide support for CLI and GUI.

SQLite References & Tutorials

- **Official SQLite Documentation:** https://sqlite.org
- **Great Tutorials:**
  - https://www.tutorialspoint.com/sqlite/
  - http://www.sqlitetutorial.net/
  - http://zetcode.com/db/sqlite/
  - http://sandersonforensics.com/forum/content.php?2 75-How-NOT-to-examine-SQLite-WAL-files
- **FOR518: Mac and iOS Forensic Analysis and Incident Response** – FOR518.com
- **FOR585: Smartphone Forensic Analysis In-Depth** – FOR585.com

Table Joins

Taking data from two (or more!) tables that have a column in common and joining them into one table. Identify tables of interest that contain unique values.

LEFT JOIN – Resulting rows are returned from the **LEFT** table even if there are no matches in the right. Using the **LEFT JOIN** produced all the text messages including those with and without attachments.

```
SELECT
ZVIBERMESSAGE.ZTEXT AS "Message Text",
ZATTACHMENT.ZNAME AS "Attachment Filename",
datetime(ZVIBERMESSAGE.ZDATE+978307200,'unixepoch',
'localtime') AS "Message Date",
ZVIBERMESSAGE.ZSTATE AS "Message Direction/State"
FROM
ZVIBERMESSAGE
LEFT JOIN ZATTACHMENT on
ZATTACHMENT.Z_PK=ZVIBERMESSAGE.ZATTACHMENT
```

INNER JOIN – Resulting rows are returned when both items are a match. Using the **INNER JOIN** (also achieved by typing "**JOIN**" in the query) returned just the messages that included attachments.

Useful Stuff

Column Renaming:
```
A_TABLE.ZAWKWARDCOLUMNNAME AS "Chat Messages"
```

Counting:
```
SELECT COUNT(*) FROM A_TABLE;
```

Aggregating with GROUP BY and COUNT (Count chat messages per contact):
```
SELECT MESSAGES,COUNT(*) FROM CHAT GROUP BY CONTACT;
```

Sorting with ORDER BY:
```
SELECT * FROM CHAT ORDER BY A_TIMESTAMP ASC
```
ASC = Ascending
DESC = Descending

Searching with WHERE and LIKE:
```
SELECT CONTACT, MESSAGE FROM CHAT WHERE CONTACT LIKE
'%Hank%'
```

Timestamp Conversion

Timestamps are stored in the databases as one of several numerical representations. *(Timestamps are assumed to be stored in UTC, you may need to verify this.)*

UNIX Epoch (10-digit number - number of seconds since 01/01/1970 00:00:00):

- `SELECT datetime(TS_COLUMN,'unixepoch')`
  Or in **local time** as suggested by the device settings (this can be done for all the following timestamps):
- `SELECT datetime(TS_COLUMN,'unixepoch', 'localtime')`

UNIX Epoch MILLISECONDS (13-digit number - number of milliseconds since 01/01/1970 00:00:00):

- SELECT datetime(TS_COLUMN/1000,'unixepoch');

Mac Absolute time, number of seconds since 01/01/2001 00:00:00. To correctly convert this timestamp, first, add the number of seconds since UNIXEPOCH time to Mac Absolute Time (978307200), then convert.

- `SELECT datetime(TS_COLUMN + 978307200, 'unixepoch');`

Chrome time accounts for time accurate to the MICROSECOND, which requires dividing the number by 1,000,000:

- `SELECT datetime(TS_COLUMN/1000000 + (strftime('%s','1601-01- 01')),'UNIXEPOCH');`

`sqlite3` CLI Options

Use the command line version of the **`sqlite3`** program (sqlite.org/cli.html) either in a SQLite shell, or just query via CLI:

```
$ sqlite3 <db_file>
$ sqlite3 <db_file> 'select * from a_table'
```

- **`.help`** – Provides a list of these 'dot-commands'
- **`.tables`** – Show the table names in the database
- **`.headers on`** – Show the column names in the output
- **`.mode column`** – Show left-aligned columns
- **`.mode tabs`** – Show tab separated columns
- **`.output <filename>`** – Send output to file
- **`.dump`** – Dump database contents (use with .output)
- **`.quit`** – Quit sqlite3 shell

Is the Database Using WAL or Journaling

The SQLite header for every database will contain offsets enabling you to differentiate if a journal or WAL is being used to support the database.

- File Offset 18 (1 byte) = x01 = Journaling
- File Offset 19 (1 byte) = x01 = Journaling

OR

- File Offset 18 (1 byte) = x02 = WAL
- File Offset 19 (1 byte) = x02 = WAL

SQLite Deletion

Examine the table to determine if data is moved to the free pages or a Boolean value is entered to mark the data deleted.

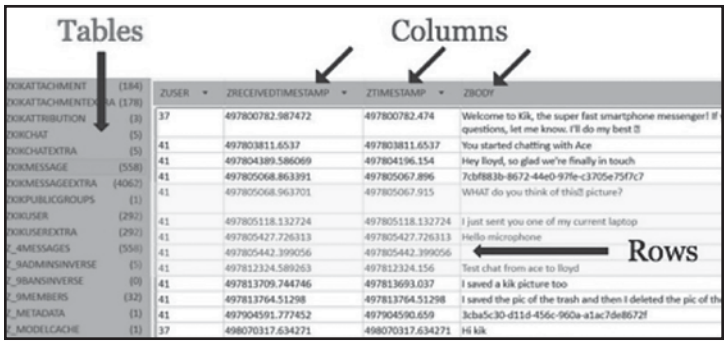
Use a SQLite Editor to examine the free pages in a Hex view to carve for deleted artifacts.

Use scripts and tools available to conduct a cursory scan of the free pages for deleted SQLite entries.

SQLite Deleted Records Parser – Python script and GUI to parse deleted data from SQLite databases – https://github.com/mdegrazia/SQLite-Deleted-Records-Parser/tree/master/Old_Versions/sqlparse_v1_1

SQLite Database Basics

SQLite databases are a self-contained database stored as a file system file (but may have a few supporting files that will also be needed for analysis!) Files have the magic number "**`SQLite format 3`**". SQLite files correspond to a database that contains tables. **Tables** contain **rows** of data with corresponding columns that describe the data in the row.



Some temporary files may also be created, including **Journal files** and **Write Ahead Logs**. **Journal files** store original data before a transaction change so the database can be restored to a known state if an error occurs. They are created by default.

Write Ahead Logs (WAL) contain new data changes, leaving original database untouched. After a set number of page changes, the WAL is used to update the actual database. Write ahead logs are optional. Journal files – stores original data before a transaction change so the database can be restored to a known state if an error occurs. (created by default)

SQLite Analysis Tools

- **DB Browser for SQLite** – Free GUI utility available for Mac/Windows/*nix – sqlitebrowser.org
- **sqlite3** – Free CLI utility available for (or native to) Mac/iOS/Android/*nix/Windows - sqlite.org/cli.html
- **SQLite Spy** – Free GUI utility for Windows – https://www.yunqa.de/delphi/products/sqlitespy/index
- **Sanderson Forensic Toolkit for SQLite** – Commercial GUI utility for Windows – sandersonforensics.com
- **SQLite Miner** – Free script to extract blobs from databases – https://github.com/threeplanetssoftware/sqlite_miner
- **Commercial tools and Browser Plug-ins** – A SQLite editor is available in most forensic tool suites

Basic Analysis Query Structure

Get everything from a single table:

```
SELECT * FROM A_TABLE;
```

Get two columns from a single table:

```
SELECT COLUMN_A, COLUMN_B FROM A_TABLE;
```

Data Types

- **NULL** – NULL Value
- **INTEGER** – Signed Integer
- **REAL** – Floating Point Number
- **TEXT** – Text String (UTF-8, UTF-16BE or UTF-16LE)
- **BLOB** – (**B**inary **L**arge **OB**jects) to store large chunks of data. This data may be a picture, video, audio, or archive (Gzip) files. This data is not defined in the database, it may contain anything an app developer desires. This data is often overlooked but may contain forensic nuggets of gold!

Operators

Arithmetic:

- Addition [+]
- Subtraction [-]
- Multiplication [*]
- Division [/]
- Modulus [%]

Comparison:

- Equal [==] or [=], Not Equal [!=] or [<>]
- Greater Than [>] or Greater Than or Equal [>=]
- Less Than [<] or Less Than or Equal [<=]

Logic:

- **IS/IS NOT** – Equal/Not Equal
- **IN/NOT IN** – Is value in (or not) a list of values?
- **LIKE (Case Insensitive)/GLOB (Case Sensitive)** – Is value like this other value? Uses wildcards.
- **AND/OR** – Use with WHERE clause to create complex logic.
- **BETWEEN** – Look for values between two values.

SANS DFIR SOF-ELK Reference Guide

This guide was created by SANS Instructor Phil Hagen with support from SANS DFIR Faculty.

Purpose

SOF-ELK is a VM appliance with a preconfigured, customized installation of the Elastic Stack. It was designed specifically to address the ever-growing volume of data involved in a typical investigation, as well as to support both threat hunting and security operations components of information security programs. The SOF-ELK customizations include numerous log parsers, enrichments, and related configurations that aim to make the platform a ready-to-use analysis appliance. The SOF-ELK platform is a free and open-source appliance, available for anyone to download. The configuration files are publicly available in a GitHub repository and the appliance is designed for upgrades in the field. The latest downloadable appliance details are at **for572.com/sof-elk-readme**.

What is "ELK" and the "Elastic Stack"?

The Elastic Stack consists of the Elasticsearch search and analytics engine, the Logstash data collection and enrichment platform, and the Kibana visualization layer. It is commonly known as "ELK", named for these three components.

The broader Elastic Stack includes other components such as the Elastic Beats family of log shippers, and various security and performance monitoring components.

All of the ELK components and the Beats log shippers are free and open-source software. Some other components of the Elastic Stack are commercially-licensed.

Booting and Logging into SOF-ELK

The SOF-ELK VM is distributed in ready-to-boot mode. You may want to add additional CPU cores and RAM if available. Do not decrease the CPU or RAM. After the VM boots, its IP address is displayed on the pre-authentication screen. This IP address is needed for both remote shell access (SSH) and web access to the Kibana interface.

Log in with the "**elk_user**" user account and password "**forensics**". The elk_user has administrative access using the sudo utility. The password should be changed after first login using local preferences or policies. The SSH server is running on the default port, 22. Access this with your preferred SSH/SCP/SFTP client software. The Kibana interface is running on port 5601. Access this with your preferred web browser. (Note that Microsoft Edge is known to be problematic.)

Updating With Git

The SOF-ELK VM uses a clone of the GitHub-based repository containing all configuration files. This allows the user to update an operational install's configuration files without needing to download a new copy of the VM itself. ALWAYS check the current GitHub repository for any notes or special instructions before updating an operational SOF-ELK platform.

To update the VM, ensure it has Internet connectivity and run the following command:

```
$ sudo sof-elk_update.sh
```

Clearing and Re-Parsing Data

Removing data from SOF-ELK's Elasticsearch indices as well as forcing the platform to re-parse source data on the filesystem itself have both been automated with a shell script. Removal is done by index, and optionally allows a single source file to be removed. The index name is required.

Get a list of currently-loaded indices:

```
$ sof-elk_clear.py -i list
```

Remove all data from the `netflow` index:

```
$ sof-elk_clear.py -i netflow
```

Remove all data from the `syslog` index and reload all source data:

```
$ sudo sof-elk_clear.py -i syslog -r
```

Remove all data from the index that was originally loaded from the `/logstash/httpdlog/access_log` file:

```
$ sof-elk_clear.py ⏎
  -f /logstash/httpdlog/access_log
```

Kibana Query Language Syntax

The Kibana user interface uses the Kibana Query Language (KQL) syntax for searching the data contained in Elasticsearch. Below are some of the basic syntaxes that will help you search data that has been loaded to SOF-ELK.

Basic Searching

The most basic search syntax is "`fieldname:value`", which will match all documents with a "`fieldname`" field set to a value of "`value`". Searches can be negated by prefixing them with "not". Some examples:

- **`hostname:webserver`**
- **`not querytype:AAAA`**

Logical Construction

Multiple searches can be combined using "`and`" and "`or`".

- **`destination_geo.asn:Amazon.com and in_bytes > 1000000`**

Numerical Ranges

Fields containing numerical values can be searched with standard range operators.

- **`total_bytes > 1000000`**
- **`return_code >= 200 and return_code <300`**

Partial String Searches

The "`*`" is used as a wildcard character.

- **`username:*admin*`**
- **`query:*.cz.cc`**

IP Addresses and CIDR Blocks

IP address fields can be searched for specific values or can use CIDR notation for netblocks.

- **`source_ip:172.16.7.11`**
- **`destination_ip:172.16.6.0/24`**

Loading Data to SOF-ELK

SOF-ELK can ingest several data formats, including:

• Syslog (many different log types supported)

• HTTP server access logs

• NetFlow

• Selected Zeek logs

• Selected EZ Tools JSON files

More sources are being tested and added to the platform and can be activated through the GitHub repository. See the "Updating With Git" section for more details on how to do this.

All source data can be loaded from existing files (DFIR Model) as well as from live sources (Security Operations Model).

DFIR Model

Place source data onto the SOF-ELK VM under the `/logstash/` directory tree.

Syslog data: `/logstash/syslog/`
Since syslog entries often do not include the year, subdirectories for each year can be created in this location – for example, `/logstash/syslog/2018/`

HTTP server logs: `/logstash/httpd/`
Supports common, combined, and related formats

PassiveDNS logs: `/logstash/passivedns/`
Raw logs from the `passivedns` utility

NetFlow from nfcapd-collected data stores:
`/logstash/nfarch/`
Use the included `nfdump2sof-elk.sh` or v`pcflow2sof-elk.sh` scripts to create SOF-ELK-compatible NetFlow ASCII files.

Zeek NSM logs: `/logstash/zeek/`
Supports multiple different log types, based on default Zeek NSM filenames

EZ Tools JSON Files: `/logstash/kape/`
Supports multiple files from the KAPE family of Eric Zimmerman's tools in JSON format. Open the necessary firewall port(s) to allow your preferred network-based ingest to occur.

Security Operations Model

Syslog: TCP and UDP syslog protocol

```
$ sudo fw_modify.sh -a open -p 5514 -r tcp
$ sudo fw_modify.sh -a open -p 5514 -r udp
```

Syslog: Elastic Filebeat shipper

```
$ sudo fw_modify.sh -a open -p 5044 -r tcp
```

NetFlow: NetFlow v5 and v9 protocols

```
$ sudo fw_modify.sh -a open -p 9995 -r udp
```

HTTP Server logs: TCP and UDP syslog protocol

```
$ sudo fw_modify.sh -a open -p 5515 -r tcp
$ sudo fw_modify.sh -a open -p 5515 -r udp
```

Configure the log shipper or source to send data to the port indicated above.

SOF-ELK Dashboards

Several Kibana dashboards are provided, each designed to address basic analysis requirements. Open the Kibana interface in a web browser using the SOF-ELK VM's IP address on port 5601.

The following dashboards are included:

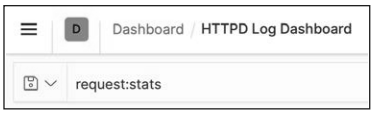
• SOF-ELK VM Introduction Dashboard • HTTPD Log Dashboard

• Syslog Dashboard• NetFlow Dashboard

Additional dashboards will be distributed through the GitHub repository. (See the "Updating With Git" section.)

The Kibana dashboards allow the analyst to interact with and explore the data contained in the underlying Elasticsearch engine. Several features provide a level of interactivity that allows dynamic analysis across vast volumes of data.

Querying Available Data



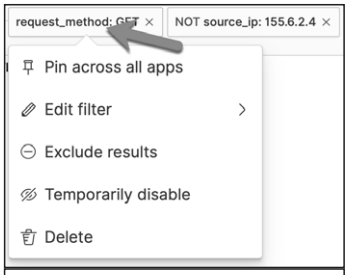
The top of each dashboard allows the user to input KQL queries, detailed in the "Kibana Query Language Syntax" section. Elasticsearch determines how well its documents match, including a "`_score`" field that indicates how well each document matches the query.

Filtering

Filters can also be applied in the Kibana interface. These are similar to queries, but are a binary match/non-match search without a "`_score`" field. Elasticsearch caches frequently-used filters to optimize their performance.

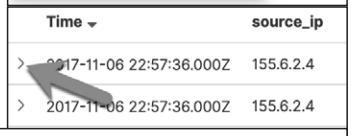
Kibana shows filters as boxes below the query field. "`Must have`" and "`must not have`" are differentiated with the red "`NOT`" text.

Filters can be modified with the drop-down menu displayed after clicking on a filter.

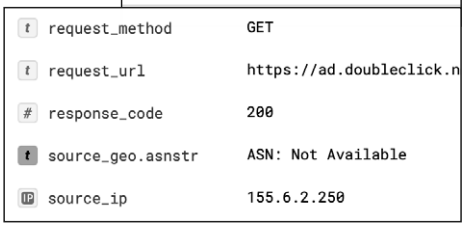


Document Expansion

When a dashboard includes a document listing panel, each document can be expanded by clicking the triangle icon on the left.



This will show all fields for the document.



Interactive Filter Generation

Each field displayed in the record details can be interactively built into a filter with the magnifying glass icons displayed when hovering over the field. The plus sign magnifying glass creates a "must have" filter, the minus sign magnifying glass creates a "must not have" filter. The table icon adds the field to the document listing panel and the final icon creates a "field must be present" filter.

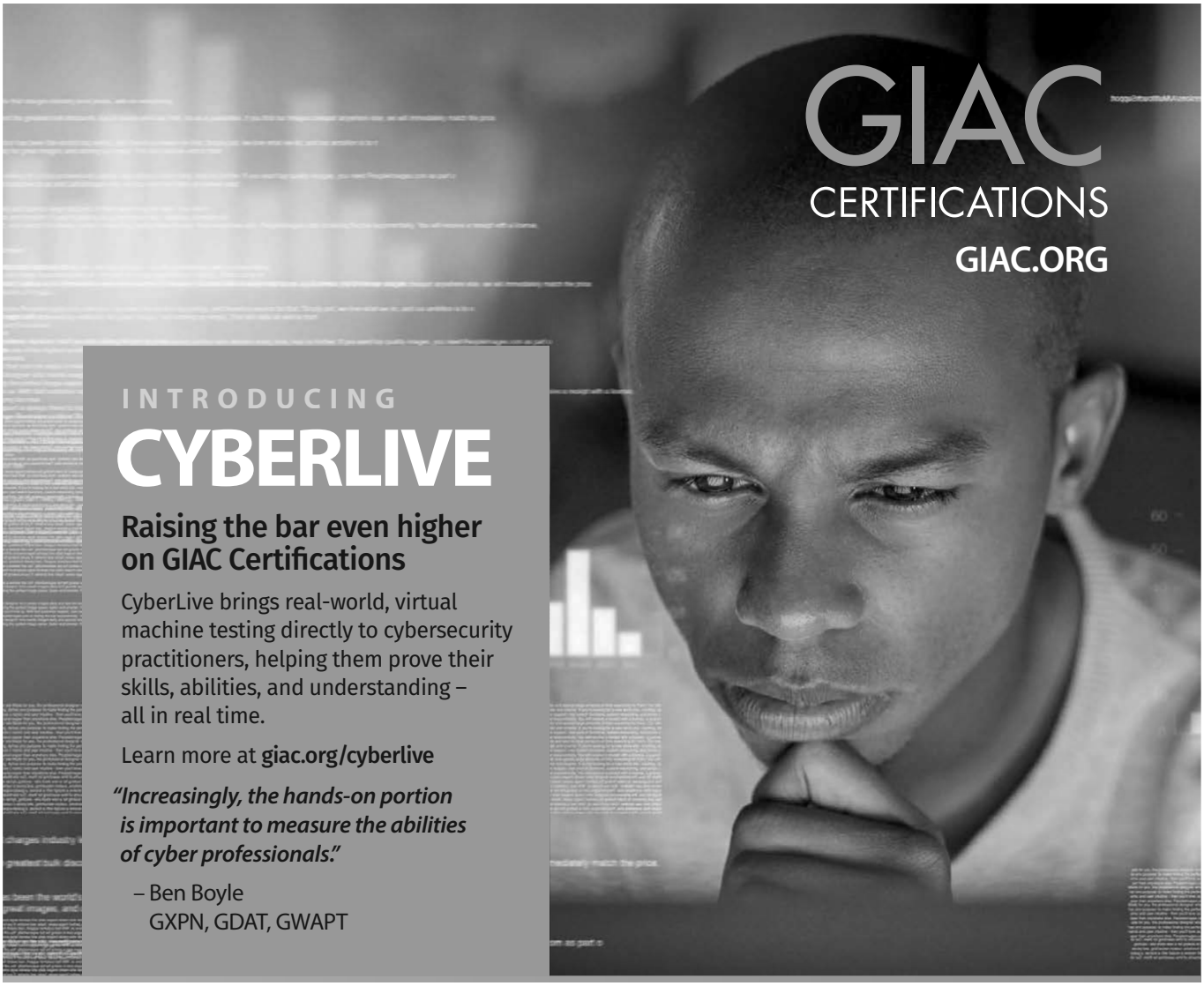
"My GIAC certification doesn't mean I'm the fount of all security wisdom, but it does assure management that security concerns brought to their attention need serious consideration."

– Jenet Hensley,
GCED, GWAPT, GSEC, GISP

GIAC develops and administers premier, professional cybersecurity certifications. Each certification aligns with SANS training and ensures mastery in critical, specialized InfoSec domains – providing the highest, most rigorous assurance of cybersecurity knowledge and skill available to industry, government, and military clients across the world.

Learn more at **GIAC.ORG**

GIAC
The Highest Standard in Cybersecurity Certification



SANS **Cyber Ranges**

A Continuum of Hands-On Learning

sans.org/cyber-ranges

SANS offers a comprehensive suite of hands-on ranges with industry-leading interactive learning scenarios:

- Develop and practice the real-world cybersecurity skills your team needs
- Available online or in-person, any time, anywhere

"NetWars is challenging for all levels of expertise, has great hints if you get stuck, and promotes continuous education."

—Jon-Michael Lacek,
Wegmans Food Markets

Why Utilize SANS Cyber Ranges?

- SANS is the most trusted name in cybersecurity
- World-class SANS instructors create our Cyber Ranges for all skill levels
- SANS Cyber Ranges can help your team assess candidates, build useful skills, and simulate real-world scenarios
- Your team members will be able to apply what they learn on SANS Cyber Ranges as soon as they return to work
- SANS Cyber Ranges are an ideal way to invest in your team's skills, enhancing retention and preparing team members to defend your environment

Assessment

Basic/Intermediate Levels

BootUp CTF

- Beginner to intermediate levels
- A wide spectrum of cybersecurity disciplines
- Individual or team-based
- Self-paced
- 1 to 2 days, scheduled a week in advance

Event/Competition

Basic/Intermediate/Expert/Pro Levels

NETWARS

- Cutting-edge challenges
- Compelling integrated storyline
- Led by SANS instructors and teaching assistants
- Individual or team-based
- Multiple versions: Core, Cyber Defense, DFIR, ICS and Power Grid
- One day, two days, or four months – scheduled a month in advance

Simulation

Intermediate/Expert Levels

CYBER42

- Built for cybersecurity leaders and managers
- Real-world decision-making scenarios
- Play as an individual or team
- Compete to build your security capabilities

NETWARS CYBERCITY

- 1:87 scale miniature physical city
- Real-world ICS assets
- Emulates commercial/residential power, transportation, water, and defense sectors
- Individual or team-based
- One to five days – scheduled a month in advance

Expert/Pro Levels

Jupiter Rockets Range

- Real-world simulation of enterprise environment
- Expert-level penetration test/ offensive skill development
- Individual or team-based
- Self-paced
- One to two days – scheduled a month in advance

Cyber STX

- Red-on-blue range emulating an advanced persistent threat
- Protect IT & OT infrastructure under active attack
- Teams of 25 to 100+ people
- One week, but can be customized
- Please schedule six months in advance

For custom Cyber Range options and the schedule of upcoming Community CTF events and NetWars Tournaments, visit **sans.org/cyber-ranges**

Free Cybersecurity Resources

sans.org/free

SANS instructors and analysts produce thousands of free resources and tools for the cybersecurity community, including more than **150 free tools and hundreds of white papers authored annually**. SANS remains committed to providing free education and capabilities to the cyber communities we serve, train, and certify.

Free Cybersecurity Community Resources

- **Internet Storm Center** – Free Analysis and Warning Service
- **White Papers** – Community InfoSec Research
- **Blog** – Cybersecurity Blog
- **Newsletters** – Newsbites; @Risk; OUCH!
- **Webcasts** – Live and Archived
- **Posters** – Job-Focused Resources
- **SANS Holiday Hack Challenge**
- **Critical Security Controls** – Recommended Actions for Cyber Defense
- **Podcasts** – Internet Storm Center Daily Stormcast; Trust Me, I'm Certified; Blueprint

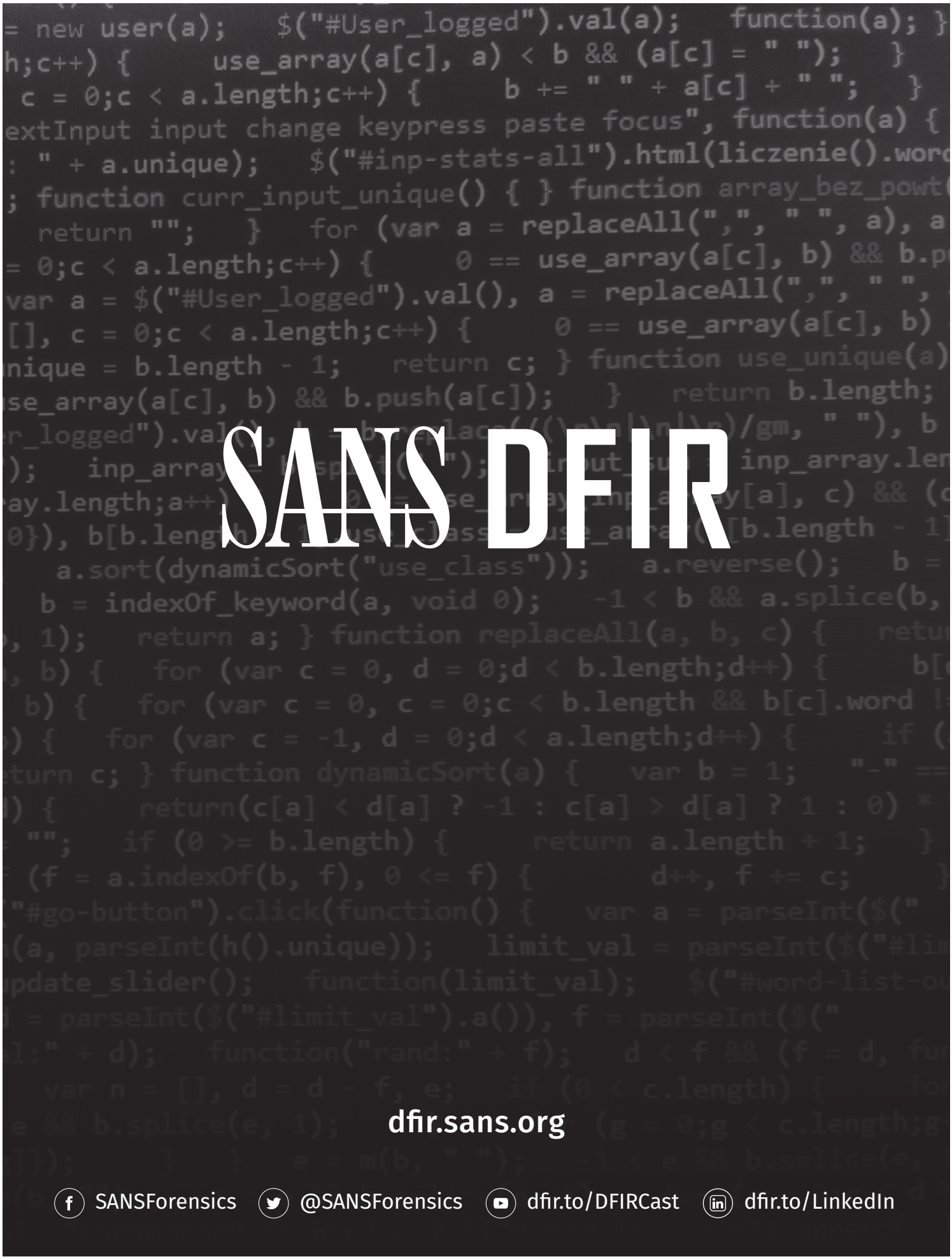
SANS Free Tools

SANS Instructors have built more than 150 open-source tools that support your work and help you implement better security.

Free Training and Events

- Test Drive 45+ SANS Courses
- Free SANS Summits & Forums
- Capture-the-Flag Cyber Challenges
- Cyber Aces

sans.org/free


SANS DFIR
dfir.sans.org
SANSForensics
@SANSForensics
dfir.to/DFIRCast
dfir.to/LinkedIn