Malware Forensics (MalFor)
work
REPORT

Word Count: 1425

Word Count

Section	Word Count
Summary	69
Methodology	12
Malware Details	10
Static Analysis	348
Dynamic Analysis	382
Reverse Engineering	355
Origins and Removal	45
Conclusions and Recommendations	91
Total	1374

Tareq Hasan
B.Sc(Engg.)- Dept. Of ICT, Islamic University, Bangladesh

Contents

Summary:

Previligelogin action breaking with the audit privilege bypassing & Delegate session user Imperonate Privilege.

User with Privileges Logon

Detects logon with "Special groups" and "Special Privileges" can be thought of as Administrator groups or privileges.
Sigma Integrated Rule Set (GitHub) - frack113

Context For The Matching Events

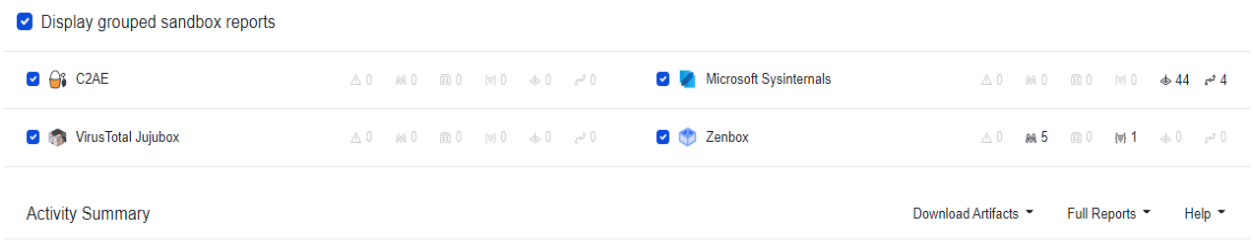
EventID:4672
PrivilegeList:SeAssignPrimaryTokenPrivilege SeTcbPrivilege SeSecurityPrivilege SeTakeOwnershipPrivilege SeLoadDriverPrivilege SeBackupPrivilege SeRestorePrivilege SeDebugPrivilege SeAuditPrivilege SeSystemEnvironmentPrivilege SeImpersonatePrivilege SeDelegateSessionUserImpersonatePrivilege
SubjectUserName:SYSTEM
SubjectLogonId:999
SubjectUserSid:S-1-5-18
SubjectDomainName:NT AUTHORITY

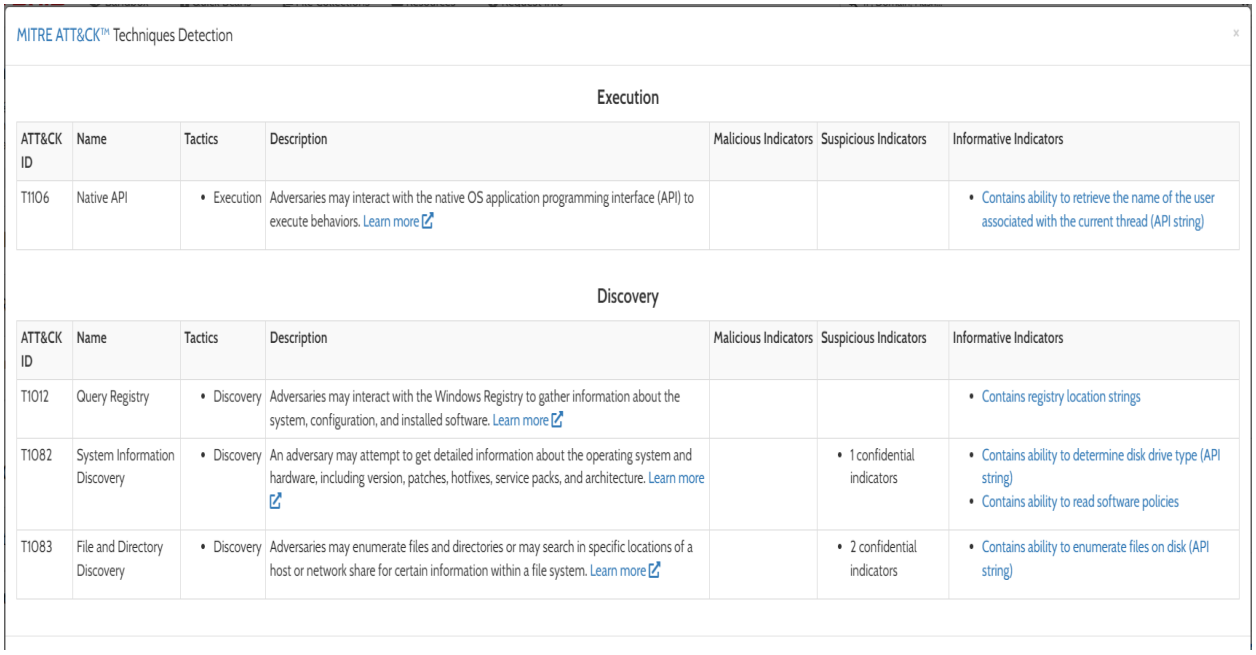
Methodology:

Used Ida Pro, Virustotal, Hybrid-Analysis, HxD & such linux commands for analyze it.

Malware Details

MITRE ATTACK techniques detection with native API execution & CAE connections.





MITRE ATT&CK™ Techniques Detection

Execution

ATT&CK ID	Name	Tactics	Description	Malicious Indicators	Suspicious Indicators	Informative Indicators
T1106	Native API	• Execution	Adversaries may interact with the native OS application programming interface (API) to execute behaviors. Learn more			• Contains ability to retrieve the name of the user associated with the current thread (API string)

Discovery

ATT&CK ID	Name	Tactics	Description	Malicious Indicators	Suspicious Indicators	Informative Indicators
T1012	Query Registry	• Discovery	Adversaries may interact with the Windows Registry to gather information about the system, configuration, and installed software. Learn more			• Contains registry location strings
T1082	System Information Discovery	• Discovery	An adversary may attempt to get detailed information about the operating system and hardware, including version, patches, hotfixes, service packs, and architecture. Learn more		• 1 confidential indicators	• Contains ability to determine disk drive type (API string) • Contains ability to read software policies
T1083	File and Directory Discovery	• Discovery	Adversaries may enumerate files and directories or may search in specific locations of a host or network share for certain information within a file system. Learn more		• 2 confidential indicators	• Contains ability to enumerate files on disk (API string)

Static Analysis

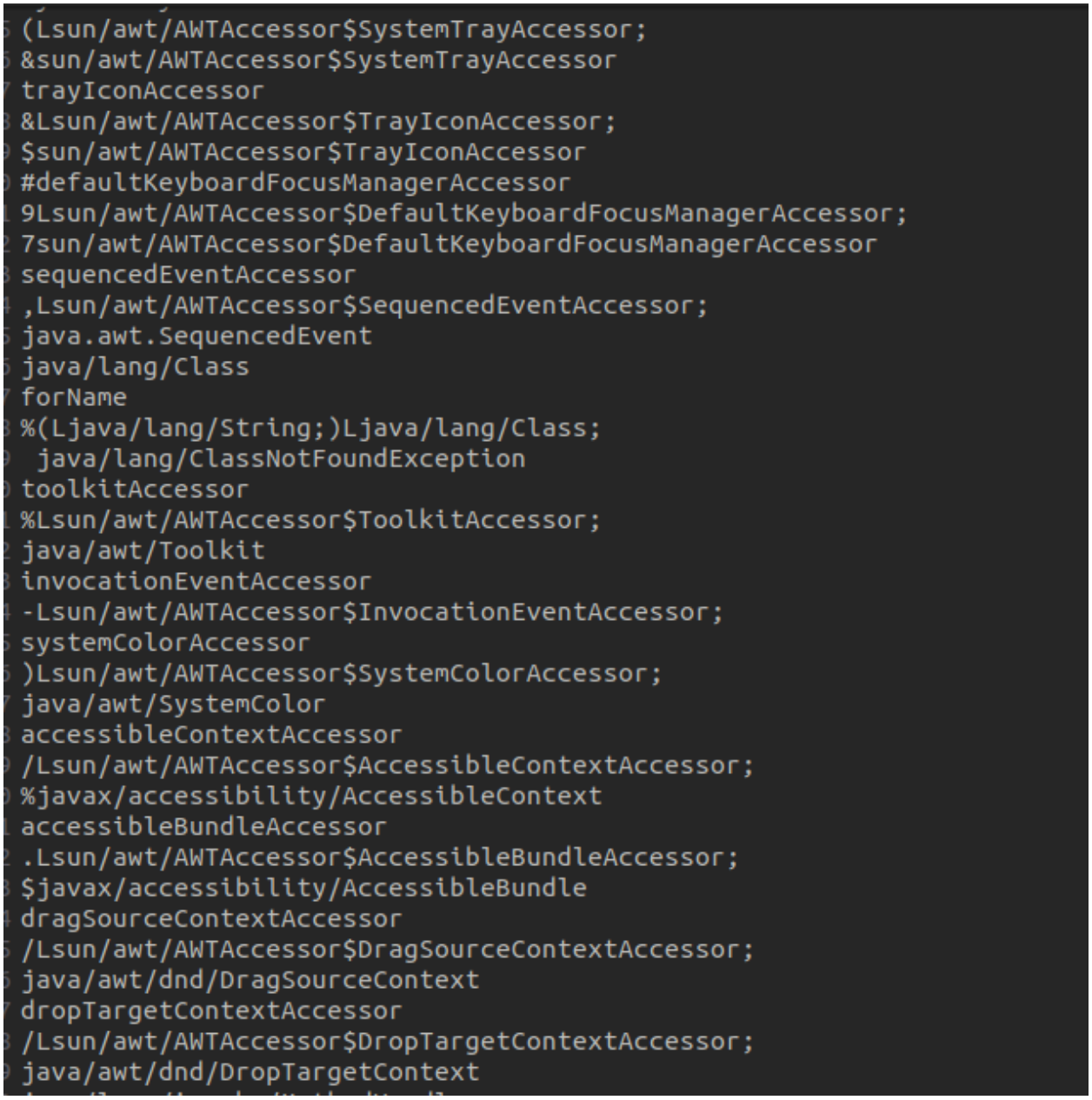
Here used Kernel based API for accessing and bypassing the Privilege logon. API preference all are kernel based.

- Suspicious Indicators4
- **Unusual Characteristics**
    - Input file contains API references not part of its Import Address Table (IAT)
    - **details**
      Found string "getsockname" (Source: 582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin, API is part of module: WS2_32.DLL)
      Found string "getsockopt" (Source: 582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin, API is part of module: WS2_32.DLL)
      Found string "GetFinalPathNameByHandleW" (Source: 582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin, API is part of module: KERNELBASE.DLL)
      Found string "getEncodingFromLangID" (Source: 582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin, API is part of module: FAKEBANKLOGIN.EXE)

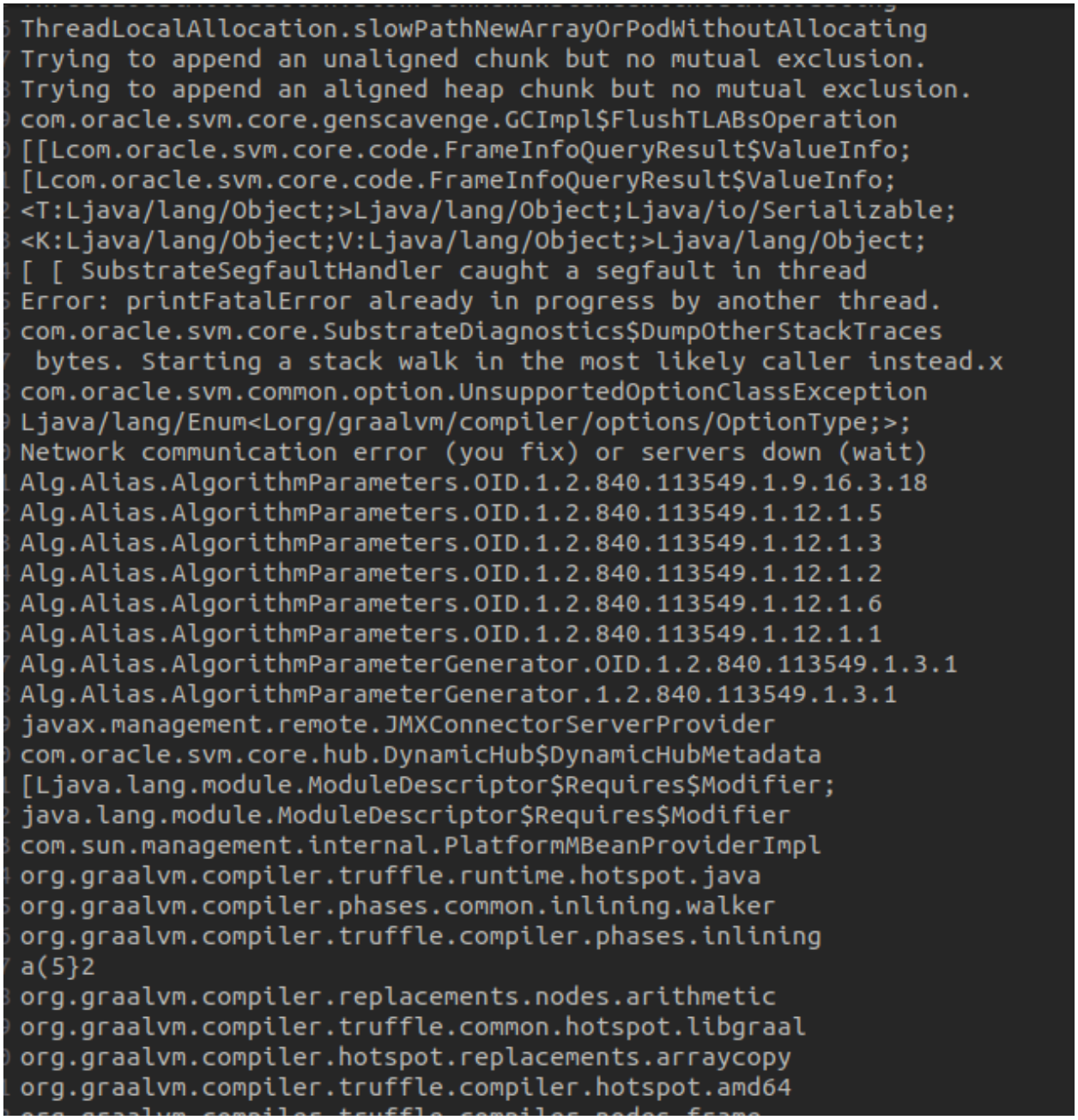
Found string "getJavaIDFromLangID" (Source:
582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin,
API is part of module: FAKEBANKLOGIN.EXE)
Found string "initialize" (Source:
582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin,
API is part of module: FAKEBANKLOGIN.EXE)
Found string "GetFileVersionInfoSizeW" (Source:
582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin,
API is part of module: VERSION.DLL)
Found string "GetFileVersionInfoW" (Source:
582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin,
API is part of module: VERSION.DLL)
Found string "VerQueryValueW" (Source:
582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin,
API is part of module: VERSION.DLL)
Found string "GetUserNameW" (Source:
582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin,
API is part of module: ADVAPI32.DLL)
Found string "OpenProcessToken" (Source:
582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin,
API is part of module: ADVAPI32.DLL)
Found string "GetUserProfileDirectoryW" (Source:
582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin,
API is part of module: USERENV.DLL)
Found string "CloseHandle" (Source:
582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin,
API is part of module: KERNELBASE.DLL)
Found string "CreateEventA" (Source:
582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin,
API is part of module: KERNELBASE.DLL)
Found string "CreateFileMappingW" (Source:
582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin,
API is part of module: KERNELBASE.DLL)
Found string "CreateFileW" (Source:
582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin,
API is part of module: KERNELBASE.DLL)
Found string "DuplicateHandle" (Source:
582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin,
API is part of module: KERNELBASE.DLL)
Found string "FlushFileBuffers" (Source:

582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin, API is part of module: KERNELBASE.DLL)
Found string "GetCurrentDirectoryW" (Source: 582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin, API is part of module: KERNELBASE.DLL)
Found string "GetCurrentProcess" (Source: 582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin, API is part of module: KERNELBASE.DLL)
source
String
relevance
10/10

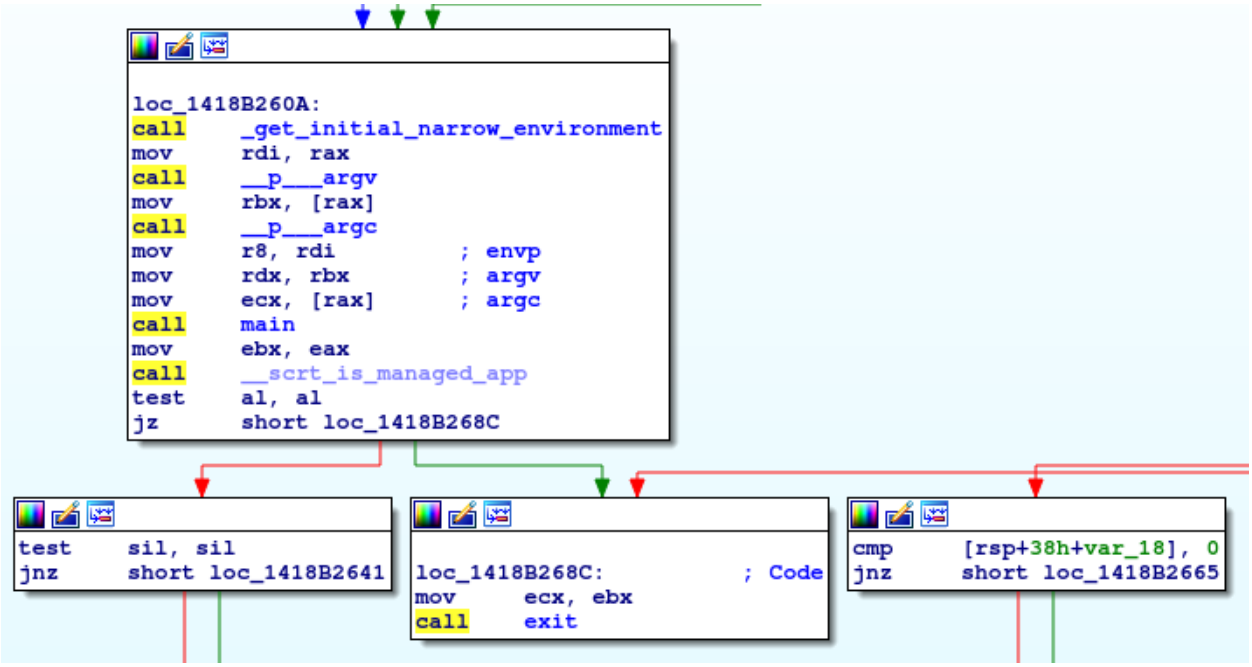
Using String analysis, Located can accessor about executable logon file. There is many sequences defect by malicious section.



& some local parameters setup detected from strings like remote connection Alogithm parameters & their connections in a thread.



& The functions Overlapping-



Need to change the accessor mode & the privilege logon.

Dynamic Analysis:

Mode & Ability-

Environment Awareness

- Contains ability to read software policies
- **details**
  "fakebanklogin.exe" (Path: "HKLM\SOFTWARE\POLICIES\MICROSOFT\WINDOWS\SAFER\CODEIDENTIFIERS"; Key: "TRANSPARENTENABLED")
  source
  Registry Access
  relevance
  1/10
  ATT&CK ID
  T1082

General

- Contains registry location strings

- **details**
  "System\CurrentControlSet\Control\TimeZoneInformation"
  "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Time Zones"
  "SOFTWARE\Microsoft\Windows\CurrentVersion\Time Zones"
  source
  String
  relevance
  1/10
  ATT&CK ID
  T1012

Found API related strings

details

"IP Helper Library GetIfTable function failed" (Indicator: "GetIfTable")

"IP Helper Library GetIpAddrTable function failed" (Indicator: "GetIpAddrTable")

"Software caused connection abort" (Indicator: "connect")

"Socket is already connected" (Indicator: "connect")

"Too many open files" (Indicator: "open")

"Network dropped connection on reset" (Indicator: "connect")

"No buffer space available (maximum connections reached?)" (Indicator: "connect")

"Socket is not connected" (Indicator: "connect")

"Socket operation on nonsocket" (Indicator: "socket")

"Protocol wrong type for socket" (Indicator: "socket")

"Cannot send after socket shutdown" (Indicator: "send")

"Successful WSAStartup not yet performed" (Indicator: "WSAStartup")

"IP Helper Library GetAdaptersAddresses function failed with ERROR_INSUFFICIENT_BUFFER" (Indicator: "GetAdaptersAddresses")

"IP Helper Library GetAdaptersAddresses function failed with ERROR_ADDRESS_NOT_ASSOCIATED" (Indicator: "GetAdaptersAddresses")

"IP Helper Library GetAdaptersAddresses function failed with error == %d" (Indicator: "GetAdaptersAddresses")

"IP Helper Library GetAdaptersAddresses function failure" (Indicator: "GetAdaptersAddresses")

"SetFilePointerEx failed" (Indicator: "SetFilePointer")

"getsockname" (Indicator: "getsockname")

"GetFullPathNameW failed" (Indicator: "GetFullPathNameW")

"Could not open file" (Indicator: "open")

Source String

Relevance 1/10

Spyware/Information Retrieval

- Contains ability to determine disk drive type (API string)

- **details**
  Observed api string:"GetDriveTypeW" [Source: 582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin]
  source
  String
  relevance
  1/10
  ATT&CK ID
  T1082

Contains ability to retrieve the name of the user associated with the current thread (API string)

details

Observed api string:"GetUserNameW" [Source: 582a5a8f175b49dffb9a0d63a6c334ac914d4b3e0320a4c6ee33f7fbef7267bb.bin]

Source String

Relevance 1/10

ATT&CK ID : T1106

Unusual Characteristics

- Possibly uses a Windows Server utility
- **details**

  "JVM_GetMethodIxExceptionIndexes called: Unimplemented" (Indicator: "exes")

  "JVM_GetMethodIxExceptionIndexes" (Indicator: "exes")

  source

  String

  relevance

  10/10

IP Traffic

192.168.0.1:137 (UDP)
20.99.133.109:443 (TCP)
20.99.184.37:443 (TCP)
23.216.147.76:443 (TCP)

Process And Service Actions

Processes Tree

2240 - %windir%\System32\svchost.exe -k WerSvcGroup

2700 - %SAMPLEPATH%

2740 - %WINDIR%\explorer.exe

2956 - wmiadap.exe /F /T /R

3000 - %windir%\system32\wbem\wmiprvse.exe

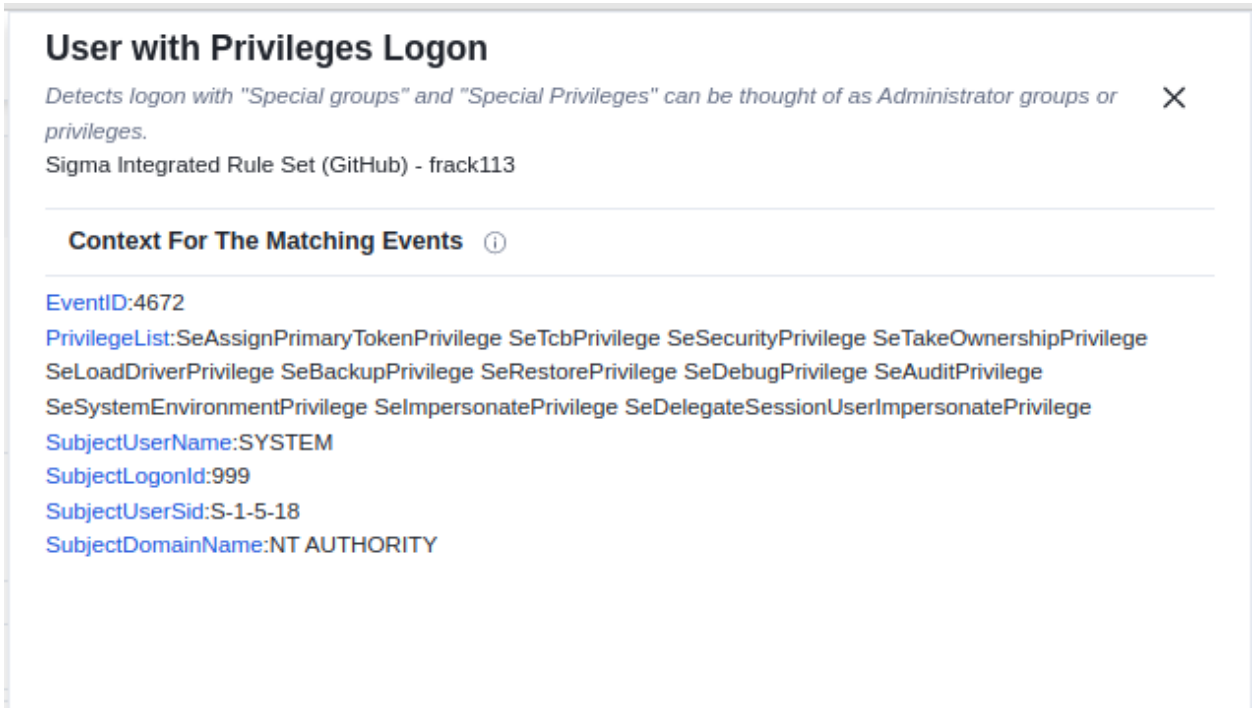
3388 - %SAMPLEPATH%\fakebanklogin.exe

3972 - C:\Windows\System32\wuapihost.exe

616 - C:\Windows\System32\svchost.exe

7464 - "C:\Users\user\Desktop\fakebanklogin.exe"

User with Privileges Logon



Reverse Engineering:

There are two kinds of disassembly syntax. They are Intel and ATT Intel, respectively. Both of them do not change the code, only the way it is displayed. The images below depict source code and equivalent assembly instructions.

```
; Exported entry  12. IsolateEnterStub__JavaMainWrapper__run__5087f
; Exported entry 247. main



; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_F8= qword ptr -0F8h
var_EC= dword ptr -0ECh
var_E8= xmmword ptr -0E8h
var_D8= xmmword ptr -0D8h
var_C8= xmmword ptr -0C8h
var_B8= xmmword ptr -0B8h
var_A8= xmmword ptr -0A8h
var_98= xmmword ptr -98h
var_88= xmmword ptr -88h
var_78= xmmword ptr -78h
var_68= xmmword ptr -68h
var_58= xmmword ptr -58h
var_40= qword ptr -40h
var_38= qword ptr -38h
var_30= qword ptr -30h
var_28= qword ptr -28h
var_20= qword ptr -20h
var_18= qword ptr -18h
var_10= qword ptr -10h
var_8= qword ptr -8
```

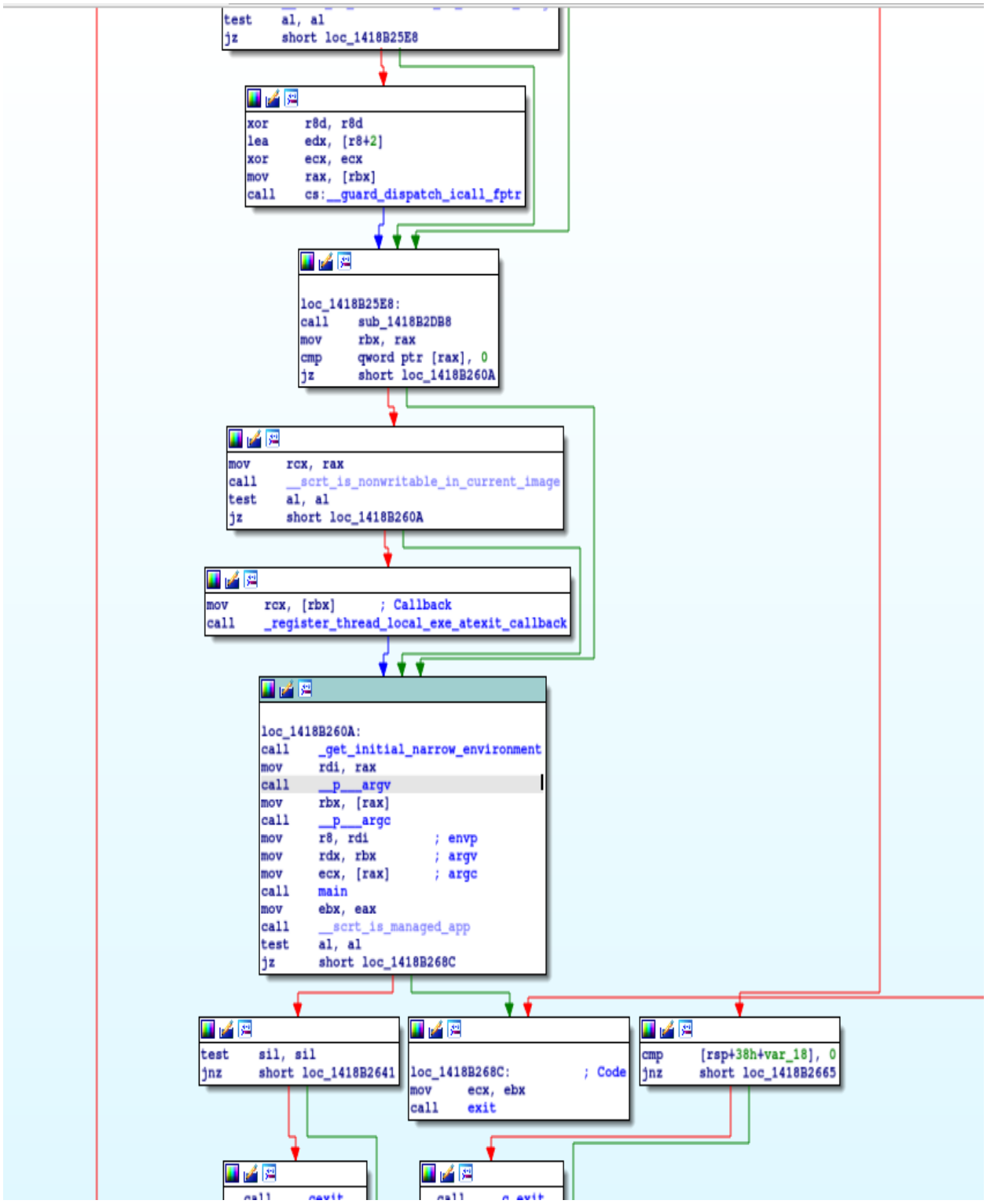
Assembly syntax is divided into two parts. The opcode is a part of the instruction that instructs the processor on what to do (MOV, PUSH). The operand is a component of the instruction that contains the data to be acted on, or the data's memory location in a register (eax 0, esp 10h).

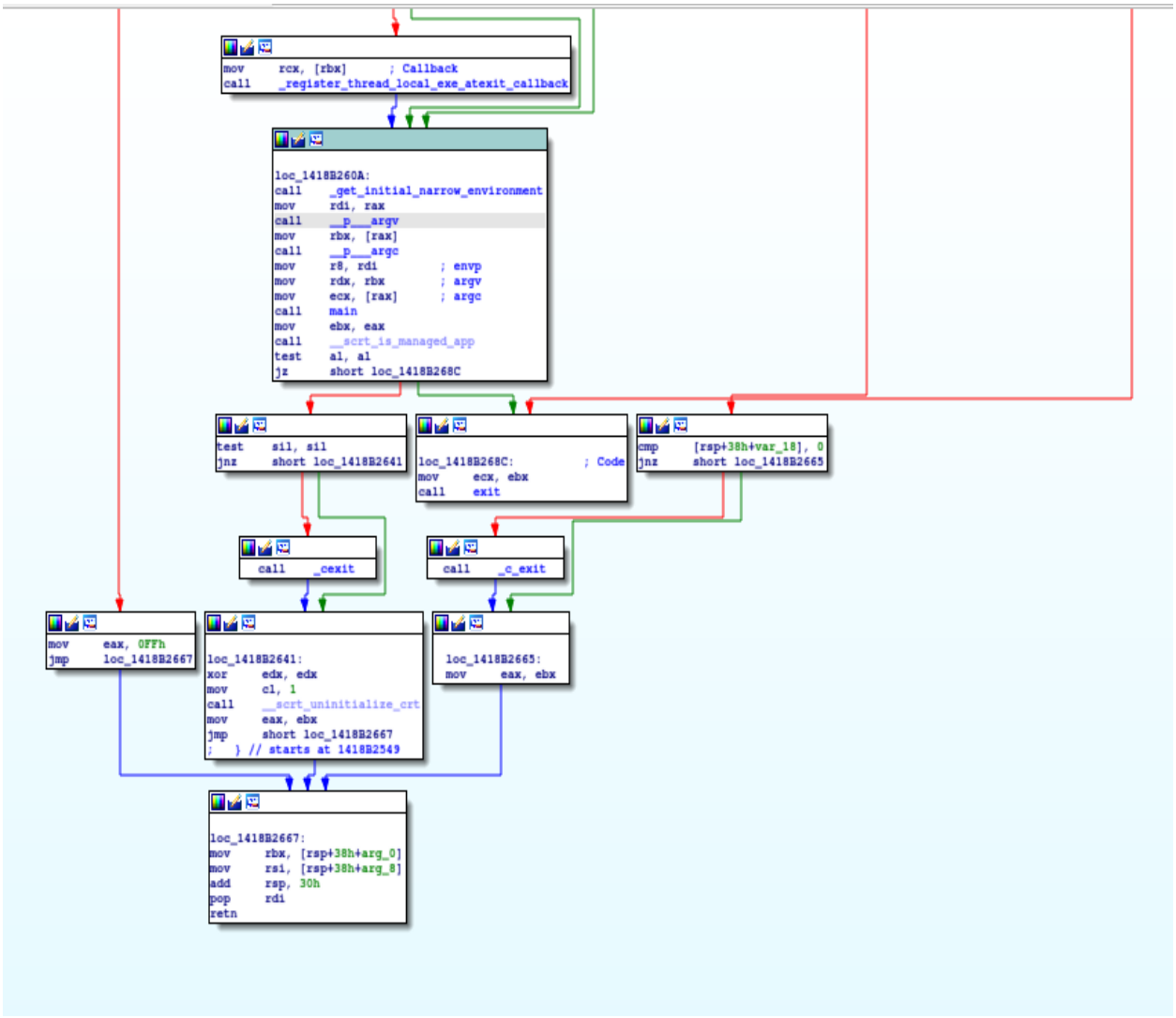
In this algorithm, machine code in executable PE sections is disassembled sequentially. It begins with the first byte in the.text section and decodes each byte until it encounters an illegal instruction. It does not support control flow features such as branches. The main issue with the algorithm is that it does not take control of the program flow and is vulnerable to errors intentionally left in the instruction stream to derail the algorithm from its path. Another issue is that this algorithm cannot distinguish between code and data in a binary file because it decodes each byte as code as long as it appears to be a legitimate code byte. Many unnecessary data bytes are interpreted as assembly instructions as a result.

```
mov     [rsp+108h+var_20], r12
mov     [rsp+108h+var_28], r13
mov     rax, r14
mov     [rsp+108h+var_30], rax
mov     r9, r15
mov     [rsp+108h+var_38], r9
mov     r10, rbp
mov     [rsp+108h+var_40], r10
movdqu  [rsp+108h+var_58], xmm6
movdqu  [rsp+108h+var_68], xmm7
movdqu  [rsp+108h+var_78], xmm8
movdqu  [rsp+108h+var_88], xmm9
movdqu  [rsp+108h+var_98], xmm10
movdqu  [rsp+108h+var_A8], xmm11
movdqu  [rsp+108h+var_B8], xmm12
movdqu  [rsp+108h+var_C8], xmm13
movdqu  [rsp+108h+var_D8], xmm14
movdqu  [rsp+108h+var_E8], xmm15
mov     [rsp+108h+var_EC], ecx
mov     [rsp+108h+var_F8], rdx
mov     r11, rdx
mov     edx, ecx
mov     r8, r11
call    sub_14000F8A0
nop
mov     rbx, [rsp+108h+var_8]
mov     rbp, [rsp+108h+var_40]
mov     rsi, [rsp+108h+var_18]
mov     rdi, [rsp+108h+var_10]
mov     r12, [rsp+108h+var_20]
mov     r13, [rsp+108h+var_28]
mov     r14, [rsp+108h+var_30]
mov     r15, [rsp+108h+var_38]
movdqu  xmm6, [rsp+108h+var_58]
movdqu  xmm7, [rsp+108h+var_68]
movdqu  xmm8, [rsp+108h+var_78]
movdqu  xmm9, [rsp+108h+var_88]
movdqu  xmm10, [rsp+108h+var_98]
movdqu  xmm11, [rsp+108h+var_A8]
movdqu  xmm12, [rsp+108h+var_B8]
movdqu  xmm13, [rsp+108h+var_C8]
movdqu  xmm14, [rsp+108h+var_D8]
movdqu  xmm15, [rsp+108h+var_E8]
add     rsp, 108h
retn
main endp
```

Linear Sweep is a much more complex and effective approach. This algorithm does not disassemble code in a linear fashion. It is based on the control flow concept. When a branch instruction is identified by the dissembler, the addresses at which the branch instruction blocks begin are determined, and the branch instruction blocks are disassembled.

Jumping to Branch Block. Noted by green arrow in IDA:-





The binary file is executed during the disassembly process, and its execution is monitored to identify the instruction actions and behavior; the execution is made for some input sets, and as a result, some binary file instruction streams can be avoided. An external tool is keeping track of this execution (debugger). The size of the executable file has no effect on the speed of disassembly because it only disassembles parts related to the real-time execution process.

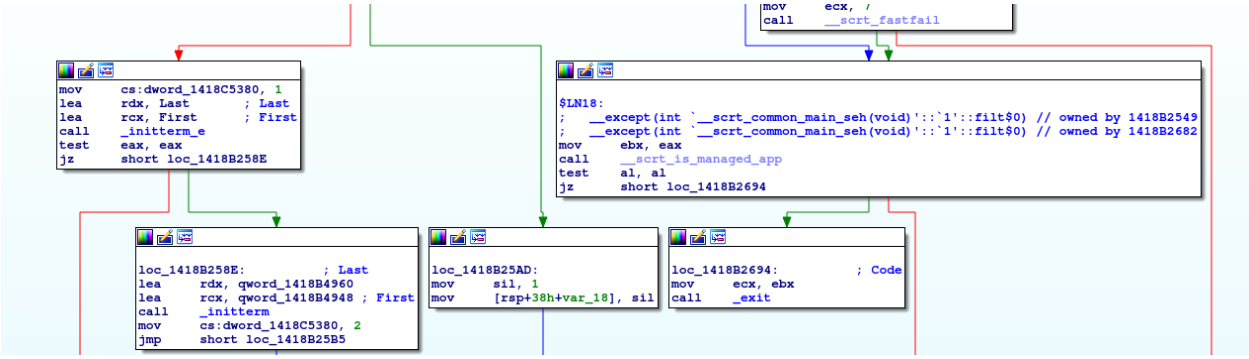
Origins and Removal:

Possibly IP Traffic:

192.168.0.1:137 (UDP)
20.99.133.109:443 (TCP)
20.99.184.37:443 (TCP)

23.216.147.76:443 (TCP)

Manage the memory map & the accessor map for the Privilege logon & their remote connection setup assemble from here-



Conclusions and Recommendations:

Reverse engineering methods have some limitations. The article primarily focused on the disassembly process as a reverse engineering method. During the disassembly process,

It is impossible to completely disassemble an application before it is compiled. The disassembler tool would not generate disassembly comments or textual identifiers such as variable and label names if run on machine code.

Because many disassemblers sequentially disassemble machine code, a single disassembly error can result in many subsequent bytes being incorrectly interpreted, and it can also be very difficult to disassemble an application due to obfuscation.

References:

1. Hoglund, G. & McGraw, G. (2004) Decompiling And Disassembling Software | Reverse Engineering And Program Understanding | Informit [Online] Available from: [20 February 2020].

2. Sikorski, M. & Honig, A. (2012) Practical Malware Analysis. 2nd edn. India:MGHills.
3. Veracode Inc. (2020) Static Testing Vs. Dynamic Testing [Online] Available from: [20 February 2020].
4. Yan, K. (n.d) System — C++ Reference from GeekfromGeek [Online] Available from: [2 May 2020].