Ransomware

Ryan Warns
Outline

- Introductions
  - Me
  - FireEye
- Malware Analysis process
  - Goals and Process
- Ransomware
  - Implementations and Reverse Engineering
- Conclusion
- Q&A
Introduction

FLARE Offensive Task Force (OTF)
Ryan Warns

- Career trajectory:
  - Reversed malware
  - Wrote malware
  - Reversed malware
  - Repurposed malware
  - ???
- Presented at BlackHat, Infiltrate, CSW
- Fun fact: met wife on Counter-Strike
  - Also almost broke up with wife because of Counter-Strike 😃
- Gen 1 Cyberdawg
- Likes Windows Internals, Binary Exploitation, and long walks on the bach
FireEye/Mandiant

- 3 Main Services
- Consultancy
  - Red Teams
  - Incident Response
- Endpoint Security
  - Tools & Associated Monitoring
- Intelligence
  - Malware families, actors, APT groups, etc.
FLARE Team

- In-depth Reverse Engineering
  - Malware Analysis for hundreds of clients
  - Host and network decoders
  - Break things

- Tools
  - Internal and external
  - Make ourselves and investigators more efficient
  - Advance the overall community (Won Volatility contest, Malware Analysis Tools, CTF Hosting)

- Training – Internal and External
  - Black Hat, Private Companies, Internal Education, etc.

- Improve Detections and Context of Alerts

- FLARE On
Community Presence

- Get our code - https://github.com/fireeye/
- Read our blog - http://www.fireeye.com/blog
- Watch our webinars - https://www.brighttalk.com/search?q=flare
- Read our whitepapers - Synful Knock, WMI Defense and Offense
- Play with our free tools - https://www.fireeye.com/services/freeware.html
  - FLOSS, FakeNet-NG, FLARE VM
Malware Analysis & You
Malware Analysis

- Malware analysis is the **art of dissecting malicious software** to:
  - Understand how it works
  - Discover how to identify it
  - Plan to remove it

- Goals of malware analysis:
  - How can you detect malware within **networks**?
  - How can you tell if a **host** is infected?
  - What are the **general capabilities** of the malicious software?
Malware Indicators – Host-Based

- Host-based indicators (HBIs) describe artifacts that malware leaves on a host
- Used to identify if an individual system is compromised
- HBIs can be anything unique about a sample:
  - File characteristics – size, hashes, names
  - Characteristics unique to the binary – strings, PDB paths
  - Changes made to the system – registry keys, created files, created directories
  - Other changes made to the system – named mutexes, started processes
Malware Indicators – Network-Based

- Network Based Indicators (NBIs) are unique characteristics of malware’s communications over a network

- NBIs are used to identify infected systems on a network

- Network indicators are meant to identify unique details of network traffic
  - Domains and IP addresses
  - Protocols and ports
  - HTTP User-Agent strings
  - Unique patterns or data formats used by the sample
Malware Classifications - Capabilities

When analyzing a sample it is useful to categorize malware into broad categories

Sample categories:

- **Backdoor/Remote Access Trojan (RAT)** – provides an attacker the ability to run commands on the infected system remotely
- **Downloaders** – contacts a remote system to download additional binaries
- **Ransomware** – encrypts files on a system to extort money from the infected
- **Crypto Currency Miner** – uses a system’s resources to mine crypto currency
- **Credential Stealer** – extracts saved passwords from various locations on a system

A sample may implement any combination of the above behaviors
Malware Analysis is **Iterative**

- **Malware Analysis is not:**
  - Static Analysis tells me the first 30%
  - Dynamic Analysis tells me the next 40%
  - Disassembly/Debugging tells me the rest

- **Rather:**
  - Maybe Static Analysis tells me 90%
  - … or Dynamic Analysis tells me nothing
  - … or on my 500th backdoor you open the command handler in IDA and know instantly
Ransomware Anatomy
Ransomware

- Ransomware:
  - Iterates through files on an infected system
  - Encrypts or destroys them
  - May use a custom or standard encryption algorithm
  - May (not) extort victims for decryption keys
Ransomware? Who cares?

- FLREQ-9937 - ELF backdoor (with full symbols), using fake TLS handshake.
- FLREQ-9939 - DRAGONJUICE recon tool.
- FLREQ-9918 - MILUM backdoor
- FLREQ-9944 - BOOSTPIPE/SANPIPE/THINPIPE code comparisons
- FLREQ-9953 - NSIS installer, acts as CobaltStrike Beacon dropper
- FLREQ-9955 - PROLOCK ransomware
- FLREQ-9961 - PROLOCK ransomware
- FLREQ-9958 - PROLOCK Unlocker/decryptor
- FLREQ-9974 - Prolock Decryptor re-implementation request
- FLREQ-9810 - GINGERYUM dropper
- FLREQ-9952 - SADFLOWER/LOWKEY/FRONTMAN code comparison
- FLREQ-9929 - MAZE ransomware
- FLREQ-9962 - MAZE ransomware
- FLREQ-9959 - MAZE ransomware decryptor utility
- FLREQ-9981 - MAZE ransomware
- FLREQ-9984 - MAZE ransomware
- FLREQ-9977 - doppelpaymer ransomware
- FLREQ-9954 - Ploutus-D ATM malware
- FLREQ-9538 - Tyupkin ATM malware
Ransomware? Who cares?

Baltimore, $18 Million Later: 'This Is Why We Didn't Pay the...
Jun 12, 2019 - ... the Baltimore ransomware attack. Why didn't Baltimore pay the ransom? How much did hackers demand? Mayor defend the city's stance.

https://en.wikipedia.org › wiki › 2019_Baltimore_ransomware_attack
The Baltimore ransomware attack occurred in May 2019, in which the American city of Baltimore, Maryland had its servers largely compromised by a new strain ...

Baltimore Ransomware Attack Update: City CIO Exits - MSSP...
https://www.msspalert.com › ransomware › baltimore-cio-departs
Oct 9, 2019 - Baltimore CIO Frank Johnson and the city part ways. Baltimore ransomware attack earlier this year damaged city's technology infrastructure.

Baltimore City Ransomware Attack Knocks City Services...
https://www.npr.org › 2019/05/21 › ransomware-cyberattacks-on-baltimore...
May 21, 2019 - The hackers used a ransomware called RobinHood — an extremely powerful and malicious program that makes it impossible to access server ...

Ransomware FAQ | City of Baltimore
https://www.baltimorecity.gov › ransomware-faq
Ransomware FAQ. Click here for a YouTube playlist of videos addressing Baltimore At Work. Question: Why don't we just pay the ransom? Answer: I know a lot ...
Ransomware - Analysis

- We see ransomware all the time
  - All the time

- Common questions:
  - What extension does the ransomware use when renaming files
  - Can we extract the ransom note
  - Can the files be decrypted after the sample has run?
Ransomware - Analysis

- Q&A Time

- You’re given a sample with our usual questions, and are asked to write a report

- You, probably, don’t have an encyclopedic knowledge of encryption algorithms

- Where do you start?
Ransomware - Analysis

- 1) Iterates through files on an infected system
- 2) Opens, encrypts, and writes files

Think: If I were writing a ransomware, how would I do this?
  - Then think: where do I look to find this in the binary?
Ransomware – File Encryption

- Multiple approaches to encrypting files
  - Creates new files, adds extension, deletes or overwrites the old
    - ReadFile
    - WriteFile
    - DeleteFile
    - MoveFile
  - In-place encryption, no new file or extension
    - CreateFileMapping
    - MapViewOfFile
    - FlushViewOfFile
A Quick Detour – The Windows API

- **Application Programming Interface (API)**
  - The source code interface provided by an operating system or library that defines the services provided by said operating system or library
  - **TL;DR** This is how you talk to Windows

- Every user application (**including malware**) must leverage the Windows API to do work

- Windows has evolved over the years, but many of the core Windows APIs have been largely unchanged for 30 years
  - Microsoft’s commitment to backwards compatibility

- The Windows API is documented in the MSDN Library.
  - Available online or can be installed locally through versions of Visual Studio
A Quick Detour – Subsystem DLLs

- Different subsystem DLLs provide different groups of functionality
- The main ones you will see are:
  - msvcr1.dll, msvcr1XX.dll – C runtime libraries, POSIX APIs
  - kernel32.dll – most vanilla APIs to manage files, processes, etc
  - kernelbase.dll – internal implementations for kernel32 APIs
  - wininet.dll and ws2_32.dll – upper- and lower-level networking functions
  - user32.dll and gdi32.dll – upper- and lower-level GUI functions
  - ntdll.dll – low-level OS management functions
A Quick Detour – The Windows API

- Internally, the Windows Kernel uses Unicode strings (UTF16-LE)
  - For international language support
- Many functions found in the Windows API have two versions:
  - An “A”-suffix version that uses narrow (ASCII) strings
  - A “W”-suffix version that uses wide (Unicode) strings
- Examples:
  - CreateFileA
  - CreateServiceW
  - RegSetValueExA
- Application developers choose to support Unicode in the program code
  - The same goes for malware authors
Life of a Windows API Call – Opening a File

kernel32.CreateFileA

KernelBase.CreateFileA

KernelBase.CreateFileW

KernelBase.CreateFileInternal

ntdll.NtCreateFile

SYSCALL/SYSENTER

• Transition to kernel mode
Life of a Windows API Call – CreateFileA

### Disassembly of CreateFileA()
- Windows 7 x86, KernelBase.dll

#### The ASCII version is a wrapper around the Unicode version

#### On Windows 7 and higher, core Windows APIs are forwarded to a new subsystem DLL, KernelBase.dll

```c
__stdcall CreateFileA(x, x, x, x, x, x) proc near

; DATA XREF: .text:100014C70
; .text:0ff_1000205470

UnicodedString - LSA_UNICODE_STRING ptr -8
lpFileName - dword ptr 8

Disassembly of CreateFileA()
- Windows 7 x86, KernelBase.dll

The ASCII version is a wrapper around the Unicode version

On Windows 7 and higher, core Windows APIs are forwarded to a new subsystem DLL, KernelBase.dll

```
Ransomware - Analysis

- 1) Iterates through files on an infected system
- 2) Opens, encrypts, and writes files

Think: If I were writing a ransomware, how would I do this?
  - Then think: where do I look to find this in the binary?
Ransomware Analysis

- Ransomware can iterate files using the following APIs:
  - FindFirstFile
  - FindNextFile

- File extension comparison occurs after a file is located using these functions

```c++
typedef struct _WIN32_FIND_DATAA {
    DWORD dwFileAttributes;
    FILETIME ftCreationTime;
    FILETIME ftLastAccessTime;
    FILETIME ftLastWriteTime;
    DWORD nFileSizeHigh;
    DWORD nFileSizeLow;
    DWORD dwReserved0;
    DWORD dwReserved1;
    CHAR cFileName[MAX_PATH];
    CHAR cAlternateFileName[14];
    DWORD dwFileType;
    DWORD dwCreatorType;
    WORD wFileAttributes;
} WIN32_FIND_DATAA, *PWIN32_FIND_DATAA, *LPWIN32_FIND_DATAA;
```
Ransomware – Putting it Together

```c
void doBadFileThings(char *dirToSearch)
{
    HANDLE hFind;
    WIN32_FIND_DATAA data;
    hFind = FindFirstFileA(dirToSearch, &data);

    if (hFind != INVALID_HANDLE_VALUE)
    {
        do
        {
            if (strstr(data.cFileName, ".jpg") ||
                strstr(data.cFileName, ".doc") ||
                strstr(data.cFileName, ".docx"))
                doRansomWare(data.cFileName);
            else if((data.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY &&
                    strcmp(".", data.cFileName) &&
                    strcmp("..", data.cFileName))
                doBadFileThings(data.cFileName);
        } while (FindNextFile(hFind, &data));
    
    FindClose(hFind);
}
```
API example – Search a Directory (x86 - 1)
API example – Search a Directory (x86 - 2)
API example – Search a Directory

```c
int cdecl sub_4C0770(int a1, int a2, LPCWSTR lpPathName, int a4, int a5, int a6, int a7, int a8)
{
    .... <snip> ...

    v28 = 0;
    if ( sub_4C0603(&a1) & 0x10 )
    {
        v8 = 1;
        v9 = sub_4C0668((int)&v24, (int)&a1, L".*");
        if ( *(DWORD *)(v9 + 8) )
            v10 = (const WCHAR *)(v9 + 8);
        else
            v18 = *(const WCHAR *)(v9 + 8);
        hFindFile = FindFirstFileW(v18, &FindFileData);
        LOBYTE(v28) = 1;
        sub_4D1660(&v24);
        if ( hFindFile != (HANDLE)-1 )
            {
                while ( v8 )
                {
                    if ( lstrcmp(FindFileData.cFileName, L"." ) )
                        {
                    if ( lstrcmp(FindFileData.cFileName, L".." )
                        {
                        v25 = &v15;
                        sub_4C0668((int)&v15, (int)&a1, FindFileData.cFileName);
                        sub_4C0755(v25, v16, v17, v18, v19, v20, v21, v22);
                        }
                    }
                    v8 = FindNextFileW(hFindFile, &FindFileData);
                }
            }
}
```
Ransomware - Analysis

- 1) Iterates through files on an infected system
- 2) Opens, encrypts, and writes files

Think: If I were writing a ransomware, how would I do this?
  - Then think: where do I look to find this in the binary?
Ransomware - Encryption

- Q&A time:

- If you were writing a ransomware how would you design it? What algorithm(s) would you use? Any other steps you would take?
Encryption Approaches

- **Symmetric**
  - Single private key to encrypt and decrypt (e.g., RC4, AES, DES)
  - Possible scenarios:
    - Generate private key and send to C2 server
    - Request private key from C2 server
  - Advantage
    - Fast
  - Disadvantage
    - Protecting the key is difficult

- **Asymmetric**
  - Uses public key to encrypt, private key to decrypt (e.g., RSA, Diffie-Hellman, ECC)
  - Possible scenarios:
    - Generate public/private key pair on system, transmit to C2
    - Encrypt files with an embedded or downloaded public key
  - Advantage
    - Key protection
  - Disadvantage
    - Slow
Tools – Crypto Identification

- Recognize complex functions that may be part of the encryption/decryption process
  - Avoid reversing these functions and instead hunt for constants and strings
- PEiD
  - KANAL plugin identifies encryption constants and APIs
  - Generate an IDC script to markup IDA disassembly
Tools – Crypto Identification

- PEiD IDA Pro markup

```
.rdata:10007B3C   db  52h ; R ; RIJNDAEL [S-inv] [char]
.rdata:10007B3C   ; RIJNDAEL (AES): inverse SBOX (for decryption)
.rdata:10007B3D   db   9
```

- FindCrypt IDA plugin

```
10007C3C: found const array Rijndael_Te0 (used in Rijndael)
1000803C: found const array Rijndael_Te1 (used in Rijndael)
1000843C: found const array Rijndael_Te2 (used in Rijndael)
1000883C: found const array Rijndael_Te3 (used in Rijndael)
10008C3C: found const array Rijndael_Td0 (used in Rijndael)
1000903C: found const array Rijndael_Td1 (used in Rijndael)
1000943C: found const array Rijndael_Td2 (used in Rijndael)
1000983C: found const array Rijndael_Td3 (used in Rijndael)
Found 8 known constant arrays in total.
```
// This can be useful in (embedded) bootloader applications, where ROM is often limited.

static const uint8_t sbox[256] = {
    0x63, 0x7c, 0x77, 0x70, 0xef, 0xe6, 0xf3, 0xe5, 0x3b, 0x83, 0x66, 0xb8, 0xf0, 0x76,
    0xca, 0x82, 0xc9, 0xc4, 0xe1, 0xf0, 0x9f, 0xe2, 0x04, 0x0d, 0x98, 0x0b, 0x02, 0x06, 0x09,
    0x08, 0x03, 0x09, 0x0e, 0x0b, 0x0d, 0x07, 0 sidl, 0x0f, 0x02, 0x30, 0x0f, 0x01, 0x0a, 0x0b,
    0x0d, 0x00, 0x95, 0xb0, 0xb7, 0x9b, 0x6f, 0x82, 0正常的文本内容。
Ransomware - Encryption

- The Windows API provides a cryptographic API implementing different algorithms
  - CryptAcquireContext – creates a crypto “object” to use during a crypto session
  - CryptCreateHash – creates a hash “object” to use during crypto session
  - CryptDeriveKey – key generations
  - CryptHashData – guess
  - CryptEncrypt/CryptDecrypt – guess

- There are other functions available to set:
  - Encryption mode (as applicable)
  - The IV
  - Padding
Ransomware – Encryption (Win API in C)

```c
if (!CryptAcquireContext(
    &hProv,
    NULL,
    MS_ENH_RSA_AES_PROV,
    PROV_RSA_AES,
    CRYPT_VERIFYCONTEXT))
{
    printf("Error %x during CryptAcquireContext!\n", GetLastError());
    goto Cleanup;
}

if (!CryptCreateHash(hProv, CALG_SHA_256, 0, 0, &hHash))
{
    printf("Error %x during CryptCreateHash!\n", GetLastError());
    goto Cleanup;
}

if (!CryptHashData(hHash, (BYTE*)szPassword, cbPassword, 0))
{
    printf("Error %x during CryptHashData!\n", GetLastError());
    goto Cleanup;
}

if (!Crypt DeriveKey(hProv, CALG_AES_256, hHash, CRYPT_EXPORTABLE, &hKey))
{
    printf("Error %x during CryptDeriveKey!\n", GetLastError());
    goto Cleanup;
}
```
Ransomware – Encryption (Win API in IDA)

- The Windows API provides a cryptographic API implementing different algorithms
  - CryptAcquireContext

```
push 0 ; dwFlags
push 18h ; dwProvType
push 0 ; szProvider
push 0 ; szContainer
mov eax, [ebp+phProv]
push eax ; phProv
call ds:CryptAcquireContextW
```

```
push 0 ; dwFlags
push PROV_RSA_AES ; dwProvType
push 0 ; szProvider
push 0 ; szContainer
mov eax, [ebp+phProv]
push eax ; phProv
call ds:CryptAcquireContextW
```

PROV_RSA_AES includes RSA, RC4, AES, MD5, etc.
**MSDN is your Friend**

---

**dwProvType**

Specifies the type of provider to acquire. Defined provider types are discussed in [Cryptographic Provider Types](#).

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROV_RSA_FULL</td>
<td>0x00000001</td>
<td>Supports both digital signatures and data encryption. It is considered a general purpose CSP. The RSA public key algorithm is used for all public key operations.</td>
</tr>
<tr>
<td>PROV_DSS</td>
<td>0x00000003</td>
<td>Supports hashes and digital signatures. The signature algorithm specified by the PROV_DSS provider type is the <a href="#">Digital Signature Algorithm (DSA)</a>.</td>
</tr>
<tr>
<td>PROV_RSA_AES</td>
<td>0x00000018</td>
<td>Supports the same as PROV_RSA_FULL with additional AES encryption capability.</td>
</tr>
<tr>
<td>PROV_DSS_DH</td>
<td>0x0000000D</td>
<td>A superset of the PROV_DSS provider type with Diffie-Hellman key exchange.</td>
</tr>
<tr>
<td>PROV_DH_SCHANNEL</td>
<td>0x00000012</td>
<td>Supports both Diffie-Hellman and Schannel protocols.</td>
</tr>
<tr>
<td>PROV_RSA_SCHANNEL</td>
<td>0x0000000C</td>
<td>Supports both RSA and Schannel protocols.</td>
</tr>
<tr>
<td>PROV_MS_EXCHANGE</td>
<td>0x00000005</td>
<td>Designed for the cryptographic needs of the Exchange mail application and other applications compatible with Microsoft Mail.</td>
</tr>
</tbody>
</table>
Ransomware - Encryption

- The most common implementation for ransomware:
  - C2 server generates a PKI pair (e.g. RSA)
  - Ransomware binary contains the public key
  - Iterate all files on the system
    - Exclude/target certain extensions
  - For each file, generate a unique symmetric (e.g. AES) encryption key
  - Encrypt the file using AES
  - Encrypt the AES key using the RSA key
  - Embed/append the encrypted AES key in the encrypted file
  - Rename the file to the ransom extension
What if we use nonstandard algorithms?

- Malware authors may attempt to “trick” analysts by using custom algorithms
  - These algorithms are not cryptographically sound, obviously
- From one sample I worked on:
  - GetComputerName -> Hash -> Take last 8 bytes
  - GetVolumeSerial -> Hash -> Take last 4 bytes
  - Rolling XOR between these two hashes
  - Concatenate with the target file name
  - Use result from 4 to create an MD5 hash
  - Take 8 characters from an MD5 hash and turn it into an AES key
  - Encrypt
Ransomware – Odds and Ends

- Ransomware commonly use VSSADMIN to...
  - … delete shadow copies (e.g. previous versions)
- All malware commonly uses a named synchronization object (e.g. mutex) to check for infection
- May use bcdedit to modify system restoration
- May beacon to a C2 to note an infection
- May use benign files to checkpoint when a directory has been encrypted
  - May be a way to “vaccinate” a system
  - e.g. .lock files found in a directory
- Common to see target extension strings to be obfuscated
Mutex Creation

```c
v0 = OpenMutexA(0xF0001u, 0, "windows_7_windows_10_check_running_once_mutex");
if (v0)
    return CloseHandle(v0);
CreateMutexA(0, 1, "windows_7_windows_10_check_running_once_mutex");
```
Destroying Backups

Command line:
"C:\Windows\System32\cmd.exe" /c vssadmin.exe Delete Shadows /All /Quiet & bcdedit /set {default}
recoveryenabled No & bcdedit /set {default} bootstatuspolicy ignoreallfailures

Deletion of Shadow copies is consistent with Ransomware execution.

Ransomware – Wrapping up

- The ransomware note is also a source of intelligence
  - Contact information (e-mail? TOR?)
  - Payment information (BTC? FIAT?)
  - Languages
  - Locations
- All of this information is useful for creating a profile against the malware author
Ransomware – Wrapping up

- For ransomware *network based indicators* would include:
  - Any C2s contacted to alert the user of infection
  - If the ransom note contains a URL (e.g. onion address)
Ransomware – Wrapping up

- For ransomware the *host-based indicators* would include:
  - Renamed file extension
  - Any intermediate or .locked files
  - Modifications of registry keys (persistence), or deletion of system restore capability
  - Named mutexes to prevent re-infection
Conclusion

FLARE Offensive Task Force (OTF)
Ransomware - Encryption

- Malware analysis is an iterative process
  - Some tools are specialized for different samples
- Thorough understanding of the Windows API makes reversing easier
- Ransomware analysis is focused on encryption algorithms used and possibility of decryption
- Combining the above allows you to take short cuts to skip over superfluous details when reverse engineering binaries