



A glance into the Eye Pyramid

RĂZVAN OLTEANU
Security Reasercher



We keep you safe and we keep it simple.



Introduction

On January 11, 2017 Italian news agency AGI, published a court order regarding cyber-attacks against high ranking Italian government members and Italian institutions. The attacks were conducted by two Italian brothers to get financial information that would help them gain an advantage when trading on financial markets.

Overview

The campaign was carried out over several years starting in 2008 and continuing into 2010, 2011, 2012 and 2014.

The mechanism the brothers used to distribute their malware was simple; targeted spear-phishing emails aimed at victims who had already been selected. The emails contained a malware attachment, which once opened harvested information from the victims' computers. This information consisted of **pictures, documents, archives, presentations, email contacts, email bodies, usernames, passwords, keystrokes, web pages content and databases.**

Technical details

The malware was written in VisualBasic.net and was obfuscated twice using common obfuscators: **Dotfuscator** and **Skater .NET** which can be easily reversed. The malware stored its sensitive data – license keys, URLs and paths – by encrypting with the **Triple DES** algorithm using the MD5 of a provided password as key and SHA256 of the password as initialization vector.





```
public static byte[] Decrypt_DES_array(ref byte[] data, ref byte[] tdeskey, ref byte[] tdesiv)
{
    byte[] result;
    try
    {
        TripleDESCryptoServiceProvider tripleDESCryptoServiceProvider = new TripleDESCryptoServiceProvider();
        ICryptoTransform transform = tripleDESCryptoServiceProvider.CreateDecryptor(tdeskey, tdesiv);
        MemoryStream memoryStream = new MemoryStream();
        CryptoStream cryptoStream = new CryptoStream(memoryStream, transform, CryptoStreamMode.Write);
        cryptoStream.Write(data, 0, data.Length);
        cryptoStream.FlushFinalBlock();
        memoryStream.Position = 0L;
        byte[] array = new byte[checked(((int)(memoryStream.Length - 1L) + 1)];
        memoryStream.Read(array, 0, array.Length);
        memoryStream.Close();
        cryptoStream.Close();
        result = array;
    }
}
```

Figure 1

Security applications

To remain unnoticed, it tried to disable any security application installed on the victim's computer. Targeted security applications were: *Ad-Aware Antivirus, Alice Total Security, AhnLab, Alwil Software, Ashampoo, AVAST Software, AVG, Avira, Bitdefender, BullGuard Ltd, CA, CCleaner, ClamAV for Windows, Comodo, DriveSentry Security Suite, DrWeb, Emsisoft Anti-Malware, Eset, Faronics, FRISK Software, Fortinet, F-Secure Internet Security, G Data, GFI Software, Grisoft, IKARUS, Immunet Protect, INCAInternet, IObit, K7 Computing, Kaspersky lab, Lavasoft, Malwarebytes' Anti-Malware, McAfee Security Scan, Microsoft Security Client, Microsoft Security Essentials, network associates, Norman, Norton AntiVirus Corporate Edition, Norton internet security, Norton security scan, Norton 360, Panda Security, PC Tools Antivirus, Quick Heal, Rising, SafeCentral, Sophos, SPAM-fighter, Spybot - Search & Destroy, SpyShredder, spyware doctor, Spyware Terminator, Sunbelt Software, Symantec AntiVirus, ThreatFire, Trend Micro, Trojan Remover, Trust-Port, UAV, Vba32, Virusbuster, Webroot, Windows Defender, Zone labs.* The method used to neutralize security software will be discussed later.

To allow network communication, the malware added itself to the firewall permission list both locally and in domain by directly setting the registry.





Persistence

For assuring its execution on system startup, the malware used the well-known method of registry **HKEY_LOCAL_MACHINE\\Software\\Microsoft\\Windows\\CurrentVersion\\Run**. This method will be discussed later.

Exfiltration

Several methods were used to exfiltrate data from the victims: by mail, by HTTP and by FTP. With each method the exfiltrated data was encrypted before being sent to the hackers. The algorithm used for encryption was **Triple DES**. The email addresses used for communication are shown in Table 1.

Emails
gpool@hostpenta(dot)com
hanger@hostpenta(dot)com
hostpenta@hostpenta(dot)com
purge626@gmail(dot)com
tip848@gmail(dot)com
dude626@gmail(dot)com
octo424@gmail(dot)com
tim11235@gmail(dot)com
plars575@gmail(dot)com

Table 1

Mail Servers
eyepyrasid(dot)com
hostpenta(dot)com
ayaxisfitness(dot)com
enasrl(dot)com
eurecoove(dot)com
marashen(dot)com
millertaylor(dot)com
occhionero(dot)com
occhionero(dot)info
wallserv(dot)com
westlands(dot)com

Table 2

The servers used for email communication are presented in Table 2.





Complexity

Overall, the malware's level of complexity is low. Because it is part of an undiscovered APT that stretches over five years it looks more complex on the surface than it actually is. The programming style is almost non-existent and the first thing that draws attention is the poorly written code. There is a lot of **redundant code**; there are different functions written for the same thing but with other values.

```
// Token: 0x060004F4 RID: 1268 RVA: 0x00007AB3 File Offset: 0x00005CB3
public static string date_format_1()
{
    return string.Format("{0:yyyyMMdd-HH:mm:ss}", DateTime.Now);
}

// Token: 0x060004F6 RID: 1270 RVA: 0x00007AE0 File Offset: 0x00005CE0
public static string date_format_3(ref DateTime dateTime_1)
{
    return string.Format("{0:yyyyMMdd-HH:mm:ss}", dateTime_1);
}
```

Figure 2

Another peculiar thing is the registry manipulation. Instead of using Windows API to read and write from the registry, the malware **creates reg files** and uses them as command line parameters for the Regedit utility as seen in Figure 3.





```

string text = Path.Combine(ConfigClass.Temp_Folder, StrUtils.Gen_Rnd_Str(".reg"));
IL 30:
num2 = 6;
string regedit = "regedit.exe /s \"" + text + "\"";
IL 44:
num2 = 7;
TextWriter textWriter = new StreamWriter(text, false);
IL 4F:
num2 = 8;
textWriter.Write("Windows Registry Editor Version 5.00\r\n\r\n");
IL 5D:
num2 = 9;
textWriter.Write "[" + keypath + "]\r\n");
IL 78:
num2 = 10;
textWriter.Write("\" + StrUtils.Escape_Str(ref keyname) + "\"=-\r\n");
IL 97:
num2 = 11;
textWriter.Flush();
IL A1:
num2 = 12;
textWriter.Close();
IL AB:
num2 = 13;
textWriter = null;
IL B1:
num2 = 14;
Interaction.Shell(regedit, AppWinStyle.Hide, true, 60000);

```

Figure 3

The oddities do not end here. The method used to neutralize security applications is clever but again execution is not so smart – the malware **removes the access rights** for all users to the files by executing Windows utility `cacls.exe` for each file in the application directory as shown in Figure 4.

```

Interaction.Shell("cacls.exe \"" + path + "\" /t /e /c /d system", AppWinStyle.Hide, true, -1);
IL 2C:
num2 = 4;
Interaction.Shell("cacls.exe \"" + path + "\" /t /e /c /d users", AppWinStyle.Hide, true, -1);
IL 47:
num2 = 5;
Interaction.Shell("cacls.exe \"" + path + "\" /t /e /c /d administrators", AppWinStyle.Hide, true, -1);

```

Figure 4





The malware did not end any security application that was running and this can be a possible early termination cause.

Another strange thing is the delay functions for minutes, seconds and milliseconds as can be seen in following images.

```
DateTime t = DateTime.Now.AddSeconds((double)seconds);  
do  
{  
    IL_37:  
    num2 = 4;  
    Application.DoEvents();  
    IL_1C:  
    num2 = 5;  
    Thread.Sleep(10);  
    IL_26:  
    num2 = 6;  
}  
while (DateTime.Compare(DateTime.Now, t) <= 0);
```

Figure 5

```
DateTime t = DateTime.Now.AddMilliseconds((double)milliseconds);  
do  
{  
    IL_36:  
    num2 = 4;  
    Application.DoEvents();  
    IL_1C:  
    num2 = 5;  
    Thread.Sleep(1);  
    IL_25:  
    num2 = 6;  
}  
while (DateTime.Compare(DateTime.Now, t) <= 0);
```

Figure 6





```
DateTime t = DateTime.Now.AddMinutes((double)minutes);
do
{
    IL_37:
    num2 = 4;
    Application.DoEvents();
    IL_1C:
    num2 = 5;
    Thread.Sleep(10);
    IL_26:
    num2 = 6;
}
while (DateTime.Compare(DateTime.Now, t) <= 0);
```

Figure 7

Although for minutes the method seems right – to process events prevents application freezing – for seconds and milliseconds this process seems to be a waste of resources. Again, these functions can easily be replaced by a single one.

Conclusion

The quality of code is the first feature that catches security researchers' attention, or the lack of it.

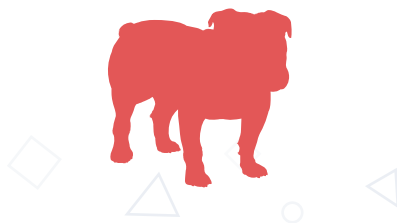
The software was built with no "low profile" in mind. This may be because the software developer has little to no experience in writing malicious programs. Also, the use of a managed programming language, the dot net languages being easier to reverse than native code, indicate little experience in malware development.

Another slip was the use of commercial components – legally purchased under the real names of the attackers – this is what led to the capture of the two brothers.



References

- www.bleepingcomputer.com/news/security/operation-eyepyramid-two-siblings-spied-on-italys-elite/
- cybersecurity.startupitalia.eu/53903-20170111-eye-pyramid-the-italian-job-storia-malware-spionaggio-massoneria
- github.com/eyepyramid/eyepyramid
- securelist.com/blog/incidents/77098/the-eyepyramid-attacks/
- blog.trendmicro.com/trendlabs-security-intelligence/-eye-storm-look-eyepyramid-malware-supposedly-used-high-profile-hacks-italy/



BullGuard Ltd and BullGuard UK Ltd

9 Devonshire Square
EC2M 4YF
London

www.bullguard.com