SUNBURST Additional Technical Details | Mandiant

By Mandiant

Published: 2020-12-24 · Archived: 2026-04-02 10:35:41 UTC

Written by: Stephen Eckels, Jay Smith, William Ballenthin

FireEye has discovered additional details about the SUNBURST backdoor since our initial publication on Dec. 13, 2020. Before diving into the technical depth of this malware, we recommend readers familiarize themselves with our blog post about the SolarWinds supply chain compromise, which revealed a global intrusion campaign by a sophisticated threat actor we are currently tracking as UNC2452.

SUNBURST is a trojanized version of a digitally signed SolarWinds Orion plugin called SolarWinds.Orion.Core.BusinessLayer.dll. The plugin contains a backdoor that communicates via HTTP to third party servers. After an initial dormant period of up to two weeks, SUNBURST may retrieve and execute commands that instruct the backdoor to transfer files, execute files, profile the system, reboot the system, and disable system services. The malware's network traffic attempts to blend in with legitimate SolarWinds activity by imitating the Orion Improvement Program (OIP) protocol and persistent state data is stored within legitimate plugin configuration files. The backdoor uses multiple obfuscated blocklists to identify processes, services, and drivers associated with forensic and anti-virus tools.

In this post, the following topics are covered in greater detail:

- Anti-Analysis Environment Checks and Blocklists
- Domain Generation Algorithm and Variations
- Command and Control (C2) behaviors for DNS A and CNAME records
- Malware modes of operation

Anti-Analysis Environment Checks

Before reaching out to its C2 server, SUNBURST performs numerous checks to ensure no analysis tools are present. It checks process names, file write timestamps, and Active Directory (AD) domains before proceeding. We believe that these checks helped SUNBURST evade detection by anti-virus software and forensic investigators for seven months after its introduction to the SolarWinds Orion supply chain.

First, the backdoor verifies that the lowercase name of the current process is solarwinds.businesslayerhost. UNC2452 avoided including this string directly in the source code by computing a hash of the string and comparing the result to the 64-bit number 17291806236368054941. The hash value is calculated as a standard FNV-1A 64-bit hash with an additional XOR by the 64-bit number 6605813339339102567. The additional XOR operation forces malware analysts to develop custom tools to brute force the hash preimage.

Next, the backdoor only executes if the filesystem last write time of the .NET assembly SolarWinds.Orion.Core.BusinessLayer.dll is at least 12 to 14 days prior to the current time. The exact threshold is selected randomly from this interval. In other words, SUNBURST lays low for almost two weeks before raising its head. If the timestamp check fails, the backdoor will execute again at a random later time when it is invoked by a legitimate recurring background task. Once the threshold is met, the sample creates the named pipe 583da945-62af-10e8-4902-a8f205c72b2e to ensure only one instance of the backdoor is running. If the named pipe already exists, the malware exits.

SUNBURST stores its configuration in the legitimate SolarWinds.Orion.Core.BusinessLayer.dll.config file. It repurposes two existing settings in the appSettings section: ReportWatcherRetry and ReportWatcherPostpone. During initialization, the backdoor determines if the ReportWatcherRetry setting is the value 3. This value indicates the malware has been deactivated and will no longer perform any network activity. As we describe later, UNC2452 can command the backdoor to disable itself. This feature may be utilized when the operator determines the victim is not of interest or that they've completed their mission. When investigating a system compromised by SUNBURST, review this setting to determine if the backdoor has been disabled. Note, the presence of this value does not offer proof the actor did not further compromise the environment before disabling SUNBURST.

The backdoor also determines if the system is joined to an Active Directory (AD) domain and, if so, retrieves the domain name. Execution ceases if the system is not joined to an AD domain. SUNBURST checks the AD domain name against a blocklist and halts execution if it contains one of the following values:

swdev.local	emea.sales	pci.local	apac.lab
swdev.dmz	cork.lab	saas.swi	dmz.local
lab.local	dev.local	lab.rio	lab.brno
lab.na	test	solarwinds	

We suspect these hard-coded AD domains may be SolarWinds internal domains that UNC2452 wanted to avoid.

Finally, SUNBURST verifies the system has internet connectivity by ensuring it can resolve the DNS name api.solarwinds.com. Otherwise, execution stops and retries at a random later time.

Anti-Analysis Blocklists

SUNBURST's behavior is affected by the presence of malware analysis and security software. To disguise the strings used to detect these security tools, UNC2452 calculated and embedded a hash value for each string. While it is trivial for the backdoor to check for the existence of a hashed process name, it is computationally expensive to determine what string a hash value corresponds to (the "preimage"). However, thanks to some hard work by members of the information security community, the hashes have been successfully brute-forced. The list of hashes and their corresponding strings can be viewed at this FireEye GitHub page.

SUNBURST uses the aforementioned FNV-1A plus XOR algorithm to compute the hash of each process name, service name, and driver filename on the system.

If a blocklisted process or driver name is found, SUNBURST pauses and tries again later. The backdoor continues past this check only when there are no processes nor drivers from the blocklist present.

If a blocklisted service is found, SUNBURST attempts to disable the blocklisted service by manipulating the service configuration in the Windows Registry. It sets the registry value HKLM\SYSTEM\CurrentControlSet\services\\Start to the value 4, which corresponds to SERVICE_DISABLED. As a result, the blocklisted service is disabled *on the next power cycle*. This means the presence of a blocklisted service on a compromised host does not make a system immune to SUNBURST.

After the registry modification is made, SUNBURST updates the ReportWatcherPostpone configuration value to reflect the service it disabled. Then, the backdoor pauses and retries the process and service blocklist checks at a later time.

Subsequent service blocklist checks skip services already present in the ReportWatcherPostpone configuration key. SUNBURST will not treat the services it has disabled as members of the blocklist anymore. Therefore, during an incident response, forensic teams should consider recovering and decoding this configuration key to parse out which services SUNBURST attempted to disable.

Domain Generation Algorithm

In this section we describe how SUNBURST uses an intermediary command and control (C2) coordinator to retrieve its final C2 server. The C2 coordinator instructs the backdoor to continue or halt beaconing. It also redirects SUNBURST to its final C2 server via DNS CNAME records. We believe this enables UNC2452 to compartmentalize their operations, limiting the network infrastructure shared among victims.

The C2 coordinator is implemented as the authoritative DNS server for the avsvmcloud[.]com domain. To communicate with the C2 coordinator, SUNBURST uses a Domain Generation Algorithm (DGA) to construct subdomains of avsvmcloud[.]com and resolves the fully qualified domain names (FQDN) using the system DNS client. The backdoor interprets the DNS responses in an unusual way to receive orders from the C2 coordinator.

The DGA generates subdomains with the following DNS suffixes to create the FQDN:

- .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

A method named Update is responsible for initializing cryptographic helpers for the generation of these random-looking C2 subdomains. Subdomains are generated by concatenating an encoded user ID with an encoding of the system's domain name. The C2 coordinator can recover the victim domain name from the encoded data and likely uses this to route SUNBURST to its final C2 server.

A user ID is generated based on three values:

- MAC address of the first available, non-loopback network interface
- Domain name

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\MachineGuid value

SUNBURST takes the MD5 hash of these combined values and encodes it using a custom XOR scheme. We believe this value is used by UNC2452 to track unique victims.

SUNBURST uses four different forms of subdomains to signify the operating mode of the backdoor. Each form contains slightly different information. However, in two of the forms, investigators can recover the domain names of victim organizations. We recommend reviewing DNS logs to confirm the presence of a victim's domain in SUNBURST C2 coordinator traffic.

When SUNBURST is in its initial mode, it embeds the domain of the victim organization in its DGA-generated domain prefix. Once the malware transitions to an "active" mode, the malware uses the other two forms of subdomains. These do not include the AD domain, but instead include encodings of either the list of running and stopped services or a timestamp.

The open-source community has done a fantastic job reverse engineering many of the subdomain forms. While we are not aware of any public decoder scripts that reverse all four possible encodings, most decoders focus on recovering the most useful information: the user ID and domain name embedded in the subdomains. We recommend that incident responders for victim organizations with access to DNS logs use these tools to confirm their AD domains are not embedded within SUNBURST generated DNS subdomains. Note that this does not indicate follow-on activity.

The following sources may be referenced for decoding such domains:

- https://securelist.com/sunburst-connecting-the-dots-in-the-dns-requests/99862/
- https://github.com/RedDrip7/SunBurst_DGA_Decode
- https://blog.cloudflare.com/a-quirk-in-the-sunburst-dga-algorithm/

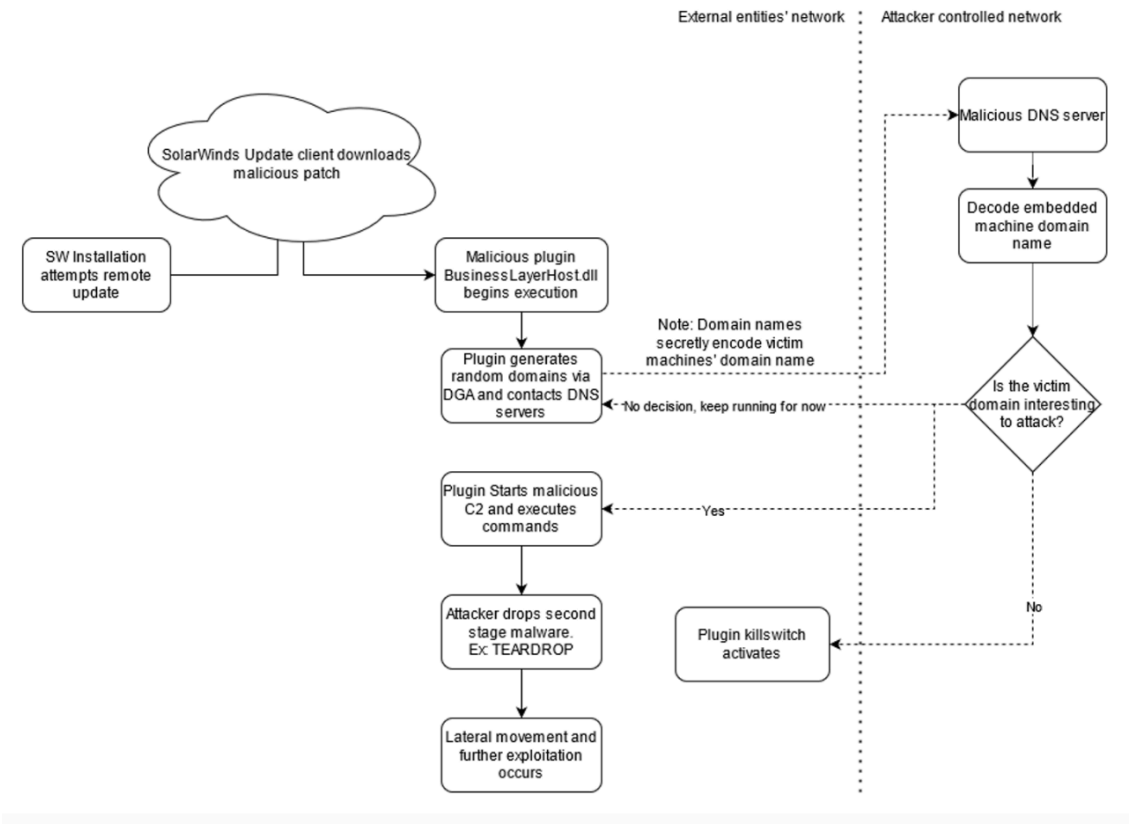


Figure 1: Diagram of actor operations and usage of SUNBURST

Command and Control

SUNBURST uses a two-part C2 protocol that involves both DNS and HTTP. In “passive” mode, the backdoor communicates with its C2 coordinator via DNS and receives high-level updates to its state. For example, the C2 coordinator may tell the backdoor to go to sleep or spring into action. When the backdoor is in “active” mode, it communicates via HTTP to its final C2 server and receives detailed commands such as “spawn a process” or “transfer a file”.

DNS C2 and the C2 Coordinator Protocol

When communicating with the C2 coordinator, the backdoor continuously generates domains via its DGA. The backdoor delays execution for random intervals between generating domains. In some cases, this delay is up to 9 hours.

If the C2 coordinator responds with a DNS A record, SUNBURST checks the resolved address against a hard-coded list of IP address blocks. If the address falls within a block, the backdoor transitions into an associated mode. The backdoor starts in the “passive” mode where it does nothing but check blocklists, sleep, and beacon via DNS until a transition is seen. Other modes are “active”, in which the malware communicates via HTTP, and “disabled”, in which the malware is permanently disabled. These modes and transitions are defined in the Modes of Operation section.

The C2 coordinator may also respond with a DNS CNAME response. In this case, the malware uses the pointed-to domain from the CNAME response for HTTPS C2 communications. SUNBURST starts a thread to handle command execution and further C2 HTTP(S) callouts. As an investigator, if you see CNAME resolutions for subdomains of avsvmcloud[.]com, it‘s possible that UNC2452 initiated follow-on C2 within the environment.

Note, the malware must receive a DNS A record response pointing to one of the following subnets immediately prior to receiving a DNS CNAME response. Otherwise, the CNAME resolution will be ignored and treated as an error. In addition, for these subnets, the least-significant bytes from the A record IP address are parsed to obtain configuration data such as the proxy method to use, the URI scheme to use, and a delay value used in the HTTP thread.

18.130.0.0/16	99.79.0.0/16	184.72.0.0/15

Subsequent DNS A record responses in the following subnet ranges cause the DGA beaconing to continue.

8.18.144.0/23	18.130.0.0/16	71.152.53.0/24	99.79.0.0/16
87.238.80.0/21	199.201.117.0/24	184.72.0.0/15	

Otherwise, subsequent DNS A record responses in the following subnet ranges cause SUNBURST to terminate and update the configuration key ReportWatcherRetry to a value that prevents further execution. This is the killswitch.

10.0.0.0/8	fc00:: - fe00::	96.31.172.0/24
172.16.0.0/12	fec0:: - ffc0::	131.228.12.0/22
192.168.0.0/16	ff00:: - ff00::	144.86.226.0/24
224.0.0.0/3	20.140.0.0/15	

While in “active” mode, the DGA subdomain encoding does not include victim machine domain name information. Instead, the DGA subdomains encode a timestamp or the list of stopped/running services. To transition back to “passive” mode, the malware operator may set the DNS server to resolve A records within the following IP ranges:

41.84.159.0/24	74.114.24.0/21	154.118.140.0/24	217.163.7.0/24

If the malware is transitioned from the “active” mode back to “passive”, the C2 coordinator must repeat the activation process if the operator wants to re-activate the sample.

HTTP(S) C2

After receiving a CNAME DNS response with a new host to communicate with, SUNBURST starts a new thread to execute the method HttpHelper.Initialize. This method is responsible for the C2 communications and

dispatching. The HTTP thread begins by delaying for a configurable amount of time that is controlled by the SetTime command. The HTTP thread delays for a minimum of one minute between callouts. The malware uses HTTP GET or POST requests. The sample disables certificate verification so it is possible to decrypt HTTPS traffic if SSL man-in-the-middle is performed.

The malware constructs HTTP GET or POST request URIs from random selections of substrings. The scheme used depends on the A record received just prior to the CNAME DNS response. When sending a GET request, the malware adds an "If-None-Match" HTTP header that includes the encoded user ID. This likely allows the C2 server to determine which SUNBURST installation generated the request, further enabling multiplexing of C2 streams on a single server.

In observed traffic, the C2 server employs steganography to hide data within HTTP response bodies and attempts to appear as benign XML related to .NET assemblies. Command data is spread across many GUID and hexadecimal strings. Commands are extracted from HTTP response bodies by searching for hexadecimal strings using the following regular expression: "\{[0-9a-f-]{36}\}"|"[0-9a-f]{32}"|"[0-9a-f]{16}". Matched substrings in the response are filtered for non-hex characters, joined together, and hex-decoded. Depending on the mode of operation, the malware may skip the steganography and send the encoded response in an HTTP response body.

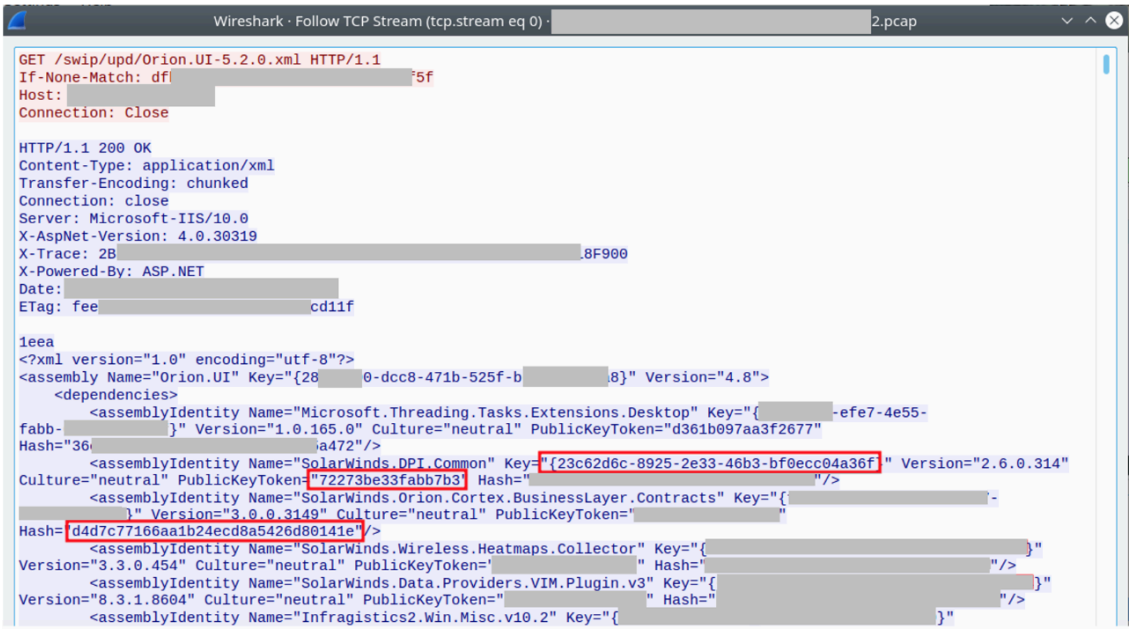


Figure 2: C2 response embeds commands in XML GUIDs

In the received C2 response, the first DWORD value indicates the actual size of the message. This value is followed immediately by the message with optional junk bytes added. The extracted message is single-byte XOR decoded using the first byte of the message. The result is then DEFLATE decompressed. The first character of the decompressed data is an ASCII integer that maps to a command ID, with optional additional command arguments delimited by space characters. The supported commands are shown in the following table.

Command	ID	Operation
Idle	0	No operation

Exit	1	Terminate the current thread.
SetTime	2	Sets the delay time between main event loop executions Delay is in seconds, and varies random between [.9 * , 1.1 *] If the delay is < 300 it is doubled on the next execution through the loop, this means it should settle onto an interval of around [5, 10] minutes o There is a second, unrelated delay routine that delays for a random interval between [16hrs, 83hrs]
CollectSystemDescription	3	Profile the local system including hostname, username, OS version, MAC addresses, IP address, DHCP configuration, and domain information.
UploadSystemDescription	4	Perform an HTTP request to the specified URL, parse the results and send the response to the C2 server.
RunTask	5	Starts a new process with the given file path and arguments
GetProcessByDescription	6	Returns a process listing. If no arguments are provided, returns just the PID and process name. If an argument is provided, it also returns the parent PID and username and domain for the process owner.
KillTask	7	Terminate the given process, by PID.
GetFileSystemEntries	8	Given a path, and a match pattern recursively list files and directories.
WriteFile	9	Given a file path and a Base64-encoded string write the contents of the Base64 string to the given file path. Write using append mode. Delay for [1s, 2s] after writing is done.
FileExists	10	Tests whether the given file path exists.
DeleteFile	11	Deletes the specified file path.
GetFileHash	12	Compute the MD5 of a file at a given path and return the result as a hex string. If an argument is provided, it is the expected MD5 hash of the file and returns an error if the calculated MD5 differs.
ReadRegistryValue	13	Arbitrary registry read from one of the supported hives.
SetRegistryValue	14	Arbitrary registry write from one of the supported hives.

DeleteRegistryValue	15	Arbitrary registry delete from one of the supported hives.
GetRegistrySubKeyAndValueNames	16	Returns listing of subkeys and value names beneath the given registry path.
Reboot	17	Attempts to immediately trigger a system reboot.

SUNBURST sends messages back to the C2 server using a protocol that consists of UTF-8 strings and an appended 4-byte error DWORD and 8-byte userID. These are bare DEFLATE compressed and single-byte XOR encoded with the XOR key prepended to the message. Message payloads of length 10,000 or less are turned into JSON documents that resemble the Orion Improvement Program (OIP) messages used legitimately by SolarWinds. Messages larger than 10,000 bytes are sent as-is in the HTTP request body.

The fake OIP document contains the following fields:

- userId – a unique identifier for the victim machine
- sessionID – stores a new random GUID for each HTTP thread
- steps – an array of objects that contain the following keys: Timestamp, Index, EventType, EventName, DurationMs, Succeeded, and Message.

The JSON key EventType is hard-coded to the value Orion and the EventName is hard-coded to EventManager. The encoded message to send to the C2 server is split up into random ranges, Base64-encoded, and stored in the Message field of the objects in the steps array. Not all objects in the steps array contribute to the malware response. The integer in the Timestamp field must have the 0x2 bit set to indicate the contents of the Message field are used in the malware response. steps objects whose 0x2 bit is clear in the Timestamp field contain random data and are discarded when assembling the malware response. This may be an attempt to hinder analysis in the absence of the malware sample.

Figure 3: HTTP callout masquerades as OIP protocol

Modes of Operation

As detailed in the DGA section, the malware has multiple modes of operation configured by the IP block that A records resolve to as well as depending on if CNAME records exist. These modes of operation are stored in internal enumerations. These mappings and values are described next.

Internal Modes

The following modes govern internal operations of the malware:

Mode Name	Value	Description
Truncate	3	Disabled; the malware killswitch has been activated and the sample may never run again without external modification to the XML configuration on-disk.
New	4	Passive mode; DGA subdomains encode the system's domain name

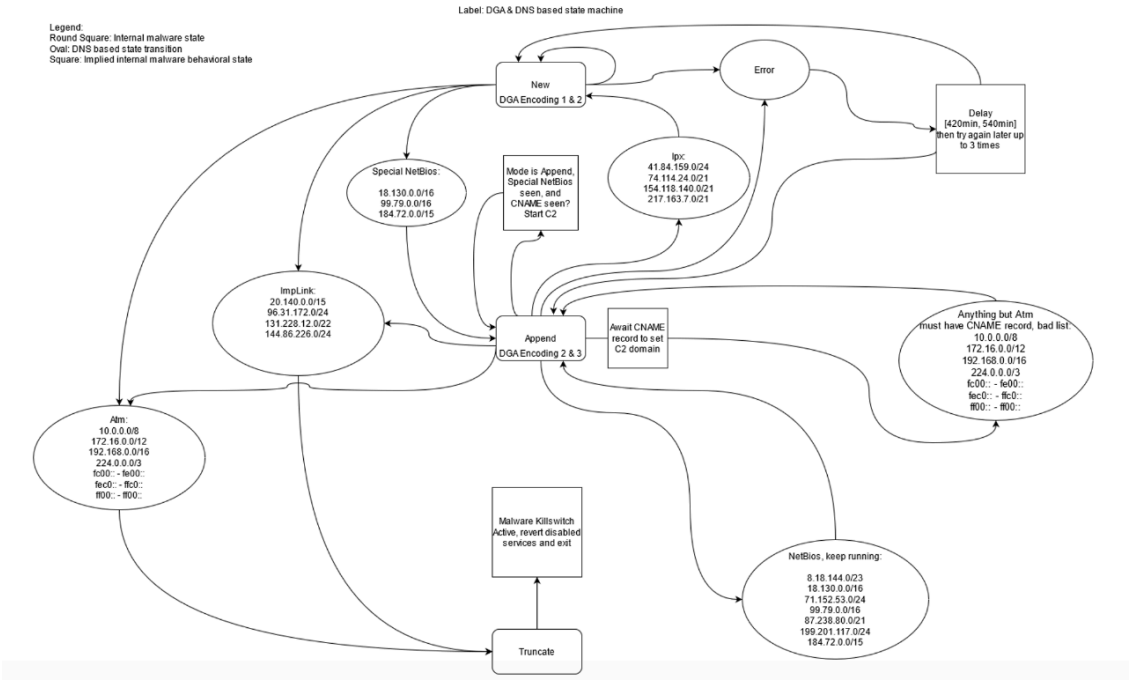
Append	5	Active mode; C2 beaconing will occur on next CNAME resolve or is already actively occurring. DGA domains encode either the userID or the list of services

The mode values are written into the backdoor configuration key ReportWatcherRetry. Investigators may consider recovering the malware configuration and inspecting this configuration value to determine the last running mode of the malware.

The following transitions govern how IP block ranges are translated into running modes:

Transition Name	**Description**
Atm	Transition to Truncate
Ipx	Transition from Append to New
ImpLink	Transition to Truncate
NetBios	Transition to Append, either start or continue C2 beaconing
Unknown	Not an IPv4 or IPv6 address, exit and retry DGA later

The following diagram describes how the SUNBURST's DGA DNS responses act as mode transitions to control the malware before HTTP-based C2 beaconing has even begun:



Additionally, here is an annotated network log showing how a sequence of DNS resolutions can transition the malware mode:

```
<subdomain>.<DGAROOT>  [184.72.193.54]          <- Current Mode: New, special A Record to prepare C2
<subdomain>.<DGAROOT>  nil                      <- Current Mode: Append, await CNAME record
<subdomain>.<DGAROOT>  [deftsecurity.com 13.59.205.66] <- Current Mode: Append, C2 SET!
<subdomain>.<DGAROOT>  [deftsecurity.com 13.59.205.66] <- Current Mode: Append, C2 SET!
<subdomain>.<DGAROOT>  [99.79.180.20]           <- Current Mode: Append, Keep Running!
<subdomain>.<DGAROOT>  [41.84.159.2]            <- Current Mode: New
<subdomain>.<DGAROOT>  [18.130.0.7]             <- Current Mode: New, special A Record to prepare C2
<subdomain>.<DGAROOT>  [websitetheme.com 76.98.15.1] <- Current Mode: Append, C2 SET!<subdomain>.<DGAROOT>
[20.140.0.1]            <- Current Mode: Truncate, killswitch!
```

To end this discussion of modes and transitions, a small note about the choices of these IP blocks. In cases such as the ImpLink IP blocks that activate the killswitch, it’s likely that the ranges were specifically chosen by the attacker to avoid being discovered by security researchers. In other cases, such as the NetBios and "special" NetBios IP blocks, the companies these blocks resolve to is likely irrelevant or at least beyond what can be definitively said without speculation.

Malware Flow Diagram

The following diagram provides a full picture of the malware's execution. Internally, SUNBURST uses a set of modes and transitions as described earlier. The names of these modes and transitions have no meaning. The malware authors purposely chose them as a form of obfuscation. When diagraming the malware's execution, these names were reused for clarity.

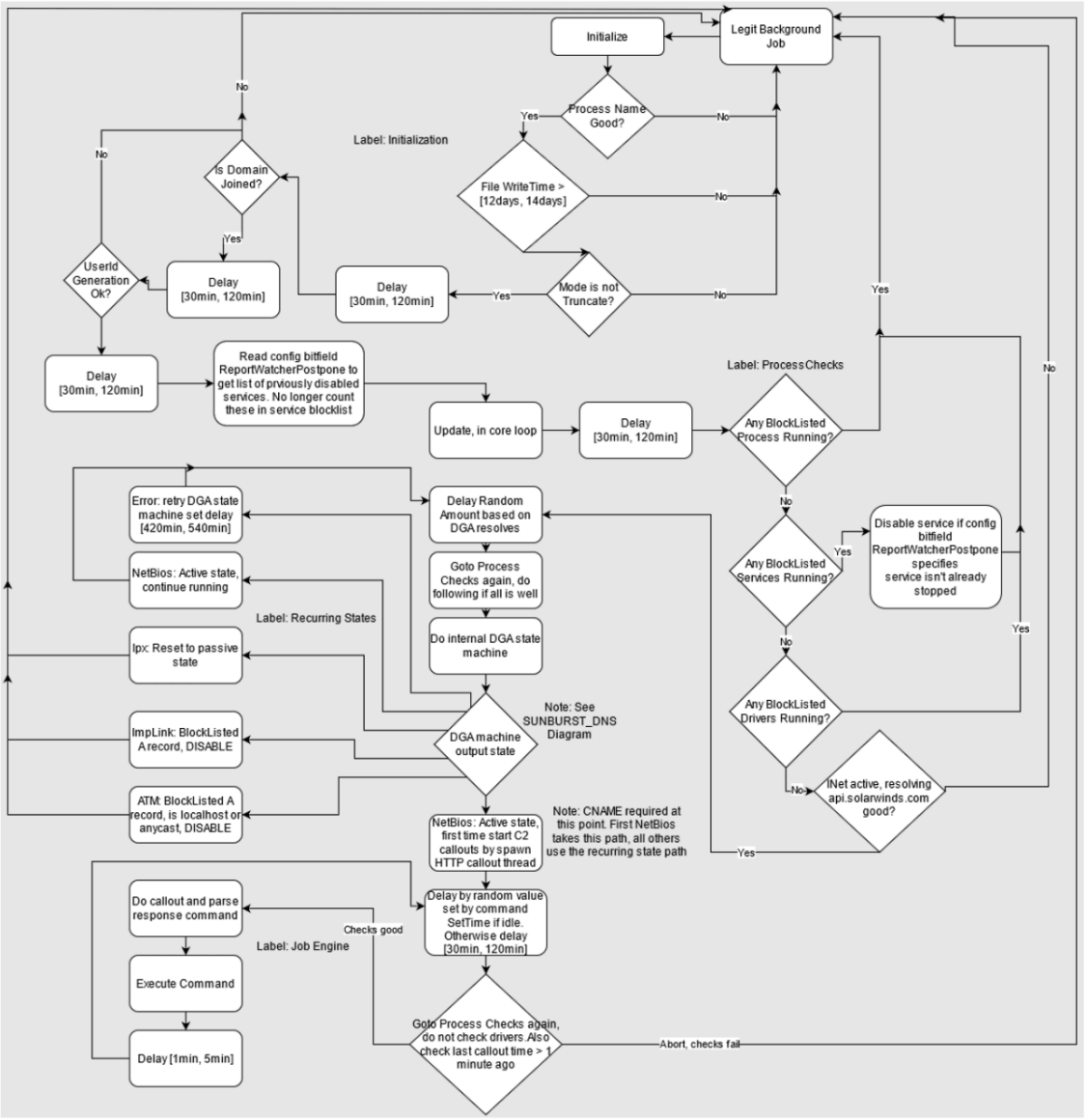


Figure 4: Malware logic and decision states

Q&A

Is a system running blocklisted processes, services, or drivers safe from compromise?

Sometimes, but not always. SUNBURST unconditionally exits if blocklisted processes or drivers are found and will not run until they are no longer detected. On the other hand, services are disabled by setting a registry value that controls startup initialization and are not explicitly stopped. As a result, a blocklisted service may still be running when the malware performs its service checks later. For this reason, it is possible for a victim system to be infected while a blocklisted service is running. Additionally, SUNBURST only attempts to disable a service once and updates its configuration to mark the service as disabled. Once the configuration is updated, the service is not treated as a blocklisted entry during subsequent execution.

Does observing one DGA encoding over another provide any information during incident response?

Short answer: it provides a hint for where to look but isn't a be-all tell-all alone. Noticing the DGA encoding change in network logs is a hint that the malware may have moved from New to Append or Append to New. This puts the malware in a mode where if a CNAME record is seen soon after, then HTTP C2 can begin. Incident response should focus on trying to identify CNAME records being successfully resolved instead of focusing on DGA encodings entirely. Identifying CNAME records is easier than tracking the malware mode through logs and a stronger signal.

What is the "killswitch"?

FireEye discovered that certain DNS responses cause the malware to disable itself and stop further network activity. With the support and help of GoDaddy's Abuse Team and the Microsoft Threat Intelligence Center, the domain used for resolving DGA domains was reconfigured to point to a sinkhole server under Microsoft's control. The IP of this sinkhole server was specially chosen to fall into the range used by the malware to transition from its current mode (New or Append) into Truncate mode where it will be permanently inactive. In other words, SUNBURST infections should now be inoculated due to the killswitch.

When C2 communication occurs, is a CNAME record required?

CNAME records are required for HTTP C2 beaconing to occur and are provided by the C2 coordinator to specify the final C2 server. C2 activity must occur over a domain name provided via a CNAME record. It cannot occur directly via a raw IP. To initialize C2 beaconing, the backdoor first looks for an A record response from one of its special NetBios subnets and subsequently expects to receive a CNAME record.

If a DGA domain is decoded to a company domain name, is that company compromised?

When the backdoor is in "passive" mode it uses the DGA encoding which embeds victim AD domain names. This means that any system where the backdoor is present may have started trying to contact DNS servers where an attacker could then activate the backdoor to begin active C2 communications. In most cases this *did not* occur and backdoors for non-targets were disabled by the operator. Therefore, it cannot be assumed that an organization experienced follow-on activity if their domain is decoded from any DNS logs. Specifically, it's only an indicator that the backdoor code was present and capable of being activated.

Public Contributions

We have seen substantial community contributions to our public SUNBURST GitHub repository.

We would like to publicly thank all contributors to this repository. Specifically, all FNV hashes embedded within SUNBURST have been brute-forced. This is a huge amount of compute power that members of the community provided free-of-charge to help others. We want to thank everyone who contributed hashes and specifically callout the Hashcat community, which organized to systematically break each hash. This was essential for breaking the final few hashes whose preimage were of considerable length.

Acknowledgements

Matthew Williams, Michael Sikorski, Alex Berry and Robert Wallace.

For additional information on UNC2452, register for our webinar, UNC2452: What We Know So Far, on Tuesday, Jan. 12, at 8 a.m. PT/11 a.m. ET.

Posted in

- Threat Intelligence
- Security & Identity
