SolarWinds: How Sunburst Sends Data Back to the Attackers

symantec-enterprise-blogs.security.com/blogs/threat-intelligence/solarwinds-sunburst-sending-data

Threat Hunter TeamSymantec

In the fourth of a series of follow-up analysis on the SolarWinds attacks, we detail how data is sent to the attackers.

In our previous blog we described how the attackers controlled the Sunburst malware, and detailed a variety of commands that will result in data being sent to the threat actors. The next technique to discuss is how Sunburst sends this data to the attackers.

If data is being sent to the attacker as a result of a command, instead of performing a HTTP(S) GET request, something we described in our last blog, Sunburst initiates a HTTP(S) POST request.

Sunburst uses randomly generated URL paths for HTTP(S) POST requests that are different from HTTP(S) GET requests.

If the data to send is greater than 10,000 bytes, the URL path will be as follows:

/pki/crl/{0}{1}-{2}.crl
- element 0 is a number between 100 and 10,000
- element 1 is optionally one of the following:
  - -root
  - -cert
  - -universal_ca
  - -ca
  - -primary_ca
  - -timestamp
  - -global
  - -secureca
- element 2 is the last error code

A Content-Type header is set to application/octet-stream and the POST data follows. The POST data consists of the data to send UTF8 encoded, concatenated with the last error code, concatenated with the userid, and subsequently compressed. Every byte of the compressed blob is then summed and the lowest byte of the sum value is used as a key. The compressed blob is XOR'd by the key byte and the key byte is prepended to the encrypted data.

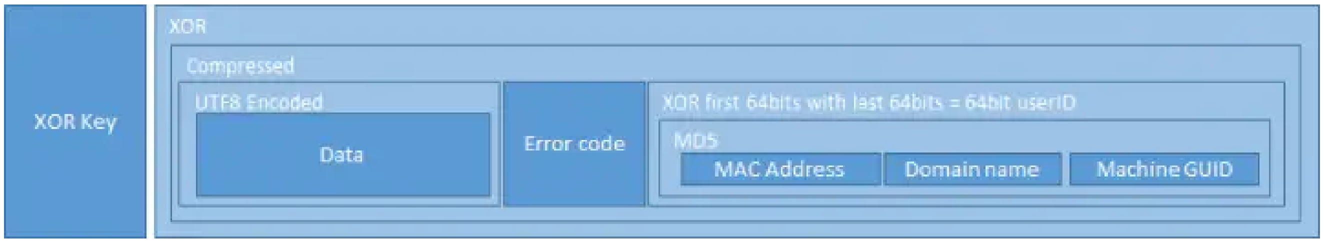


Figure 1. Structure of Sunburst POST data

If the data to send is less than, or equal to, 10,000 bytes, the URL path will take one of two forms as follows:

- /fonts/woff/{0}-{1}-{2}-webfont{3}.woff2
    - element 0 is a random number between 100 and 10,000
    - element 1 is "opensans" or “noto”
    - element 2 is one of the following:
        - bold
        - bolditalic
        - extrabold
        - extrabolditalic
        - italic
        - light
        - lightitalic
        - regular
        - semibold
        - semibolditalic
    - element 3 is the last error code
- or /fonts/woff/{0}-{1}-{2}{3}.woff2
    - element 0 is a random number between 100 and 10,000
    - element 1 is one of the following:
        - freefont
        - SourceCodePro
        - SourceSerifPro
        - SourceHanSans
        - SourceHanSerif
    - element 2 is one of the following:
        - Bold
        - BoldItalic
        - ExtraBold
        - ExtraBoldItalic
        - Italic
        - Light
        - LightItalic
        - Regular
        - SemiBold
        - SemiBoldItalic
    - element 3 is the last error code

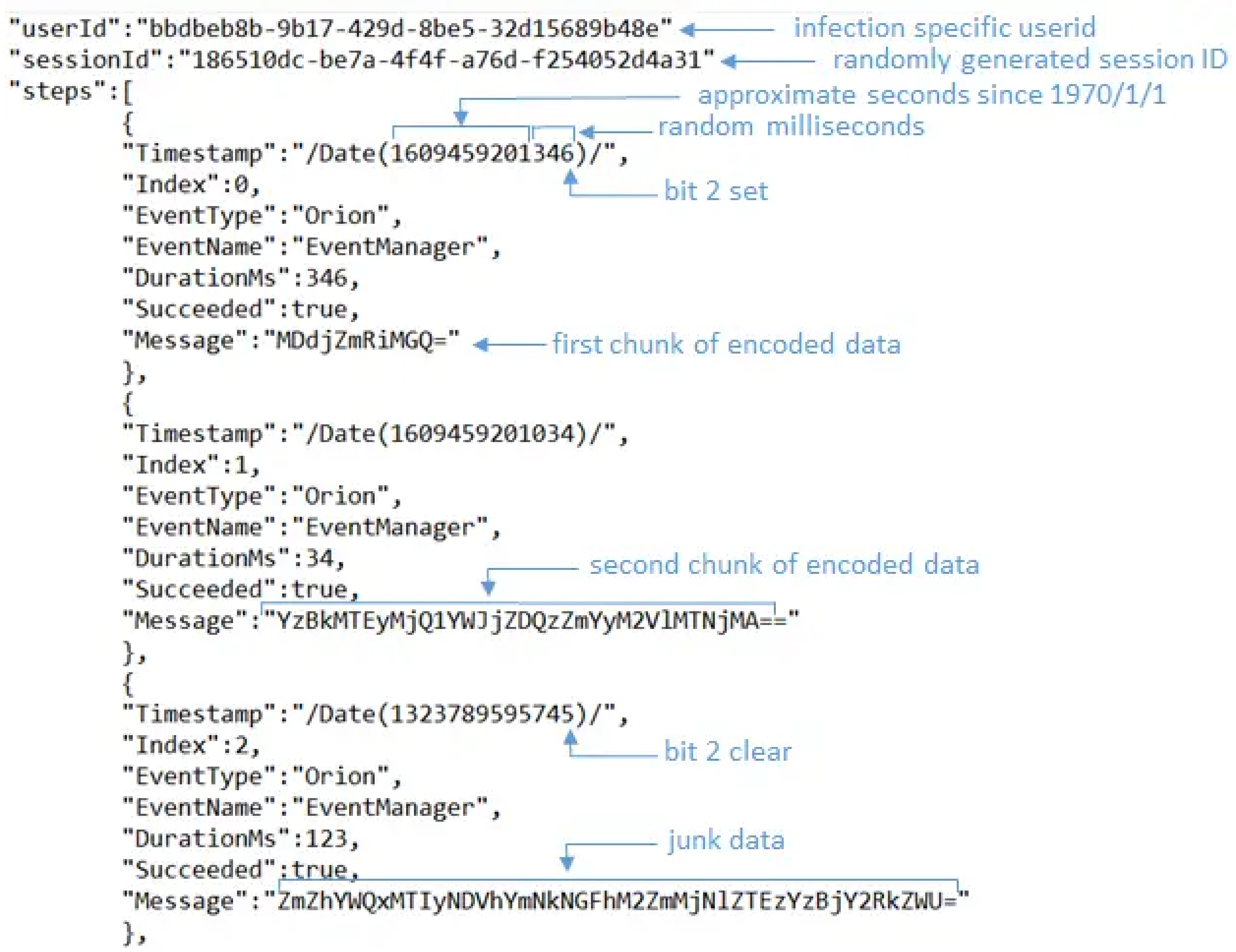
Further, instead of sending the encrypted data directly, as when the data is greater than 10,000 bytes, the data is steganographically sent in a faux JSON blob.

The JSON blob contains the following fields:

- userId - contains the userid obfuscated into a GUID
- sessionId - a randomly generated GUID per HTTP session

- Timestamp - milliseconds since 1970/1/1 minus five minutes plus a random value between 0-512 with the second bit normally set to 1
- Index - an incrementing value
- EventType - set to Orion
- EventName - set to EventManager
- DurationMs - the same random value between 0-512 used in the Timestamp
- Succeeded - set to true
- Message - a chunk of Base64 encoded data

The encrypted data to send is broken into multiple variable sized chunks. The size of each chunk is randomly determined, but generally will go from smaller to larger. If the randomly chosen size is 0, a random chunk between 16 and 28 bytes is generated instead and the Timestamp value is AND'd with the value 18446744073709551613, which more importantly sets the second bit to 0. Each chunk is then encoded and added to the JSON blob and sent as the HTTP(S) POST data with a Content-Type header set to application/json.



```
"userId":"bbdbeb8b-9b17-429d-8be5-32d15689b48e"
"sessionId":"186510dc-be7a-4f4f-a76d-f254052d4a31"
"steps":[
        {
        "Timestamp":"/Date(1609459201346)/",
        "Index":0,
        "EventType":"Orion",
        "EventName":"EventManager",
        "DurationMs":346,
        "Succeeded":true,
        "Message":"MDdjZmRiMGQ="
        },
        {
        "Timestamp":"/Date(1609459201034)/",
        "Index":1,
        "EventType":"Orion",
        "EventName":"EventManager",
        "DurationMs":34,
        "Succeeded":true,
        "Message":"YzBkMTEyMjQ1YWJjZDQzZmYyM2VlMTNjMA=="
        },
        {
        "Timestamp":"/Date(1323789595745)/",
        "Index":2,
        "EventType":"Orion",
        "EventName":"EventManager",
        "DurationMs":123,
        "Succeeded":true,
        "Message":"ZmZhYWQxMTIyNDVhYmNkNGFhM2ZmMjNlZTEzYzBjY2RkZWU="
        },
```

infection specific userid
randomly generated session ID
approximate seconds since 1970/1/1
random milliseconds
bit 2 set
first chunk of encoded data
second chunk of encoded data
bit 2 clear
junk data

Figure 2. A contrived example of a JSON file that would be sent by Sunburst
On receipt, the attacker will need to decode and concatenate all the Message chunks, skipping junk chunks where the Timestamp second bit is not set.

This blog is the final installment of our analysis series concerning the command and control (C&C) process used by the actors behind the SolarWinds attacks. Previous entries have covered how the attackers used Sunburst to disable security software and avoid detection, the malware's use of a rare domain generation algorithm (DGA) to generate domain names to contact for C&C purposes, and Sunburst's control flow and use of IP address values as commands. This final blog, detailing how the malware sends data back to the attackers, rounds out a comprehensive overview of Sunburst's C&C processes.

Protection/Mitigation

Tools associated with these attacks will be detected and blocked on machines running Symantec Endpoint products.

File-based protection:

- Backdoor.Raindrop
- Backdoor.Teardrop
- Backdoor.Sunburst
- Backdoor.Sunburst!gen1
- Backdoor.SuperNova

Network-based protection:

System Infected: Sunburst Malware Activity

For the latest protection updates, please visit the Symantec Protection Bulletin.

About the Author

Threat Hunter Team

Symantec

The Threat Hunter Team is a group of security experts within Symantec whose mission is to investigate targeted attacks, drive enhanced protection in Symantec products, and offer analysis that helps customers respond to attacks.

Want to comment on this post?