Due to masked WebSocket usage with Masked payloads and XOR in general used by malware for network "encryption", I'm wondering if it would be possible to add support for XOR similar to the existing base64_decode/base64_data keywords.

The only existing method I am aware of for achieving this outcome using existing features is a Lua script/rule. However this depends heavily on user configuration to be useful. Providing an XOR keyword has the benefit of not requiring Lua support and provides a general purpose function that could be used with Masked Payloads within WebSockets and any other network communications using XOR.

WebSocket support has been requested here - https://redmine.openinfosecfoundation.org/issues/2695, but does not directly address the use of Masked Payloads.

An example of keyword usage might be

    xor:key <xor key in hex>, bytes <value>, offset <value>, relative;
    xor_data;

Thanks

Related issues:
- Related to Feature #2695: websocket support
- Related to Task #4097: Suricon 2020 brainstorm
- Related to Task #4762: Suricon 2021 brainstorm

History

#1 - 10/29/2019 01:10 PM - Brandon Murphy
Having given a bit more thought, this solution would only work where XOR keys are known. This limitation moves the usefulness of this request to address Masked Payloads of WebSockets as the XOR Key is supposed to be randomly per each WebSocket frame.

#2 - 11/01/2019 09:30 AM - Jason Ish
- Related to Feature #2695: websocket support added

#3 - 11/05/2019 12:01 PM - Victor Julien
- Status changed from New to Feedback
- Assignee set to Community Ticket
- Target version set to TBD

I suppose it would be useful to use the result of byte_extract as input to the key.

#4 - 12/12/2019 02:55 PM - Brandon Murphy
- File 8d44f3fd-110c-423f-82be-c09a1a52a0f8.pcap added

Adding a real world example of how this will be helpful.

AZORult 3.2 uses a static XOR key to encode network communications. PCAP is attached and taken from Any_Run
See Packet 74 for the initial checkin via POST. This traffic can be decoded as described in this CyberChef Recipe.

The Initial Checkin of AZORult uses a unique ID generated from system details as documented by Cylance.

Today, as the values change depending on each infected system, detecting the initial checkin is difficult and very prone to false negative, or is based on "circumstantial" detection, or based on post initial checkin activity.

The requested feature would allow for direct detection of this type of CnC communicators.

Using the attached pcap as an example. Here is a rule utilizing the proposed keyword.

```
alert http $HOME_NET any -> $EXTERNAL_NET any (http.method; content:"POST"; http.uri; content:".*"; endswith; http.request_body; content:"4a 2f 2b"; fast_pattern; depth:3; xor: key 0d0ac8, bytes 133, offset 0; xor_data; pcre: "/^G(?:[A-F]|%3[0-9]){7}%2D(?:[A-F]|%3[0-9]){8}%2D(?:[A-F]|%3[0-9]){8}%2D(?:[A-F]|%3[0-9]){8}%2D(?:[A-F]|%3[0-9]){9}%2D$/"; sid:1; rev:1; classtype:command-and-control;)
```

notice the xor keyword is applied to the http.request_body buffer and xor_data is a sticky buffer.

also, while 133 bytes is longer than the buffer, 133 bytes is the longest possible encoded/xor'ed unique ID.

Note - this rule will cover any of the unique IDs that are G[0-9] when fully decoded. Additional rule(s) would be required to match G[A-F] while maintaining a maybe okish fast pattern;

#5 - 12/12/2019 02:57 PM - Brandon Murphy
Victor Julien wrote:

I suppose it would be useful to use the result of byte_extract as input to the key.

Yes, that would be very useful. It would address the WebSockets use case and, i've seen more than one sample where malware configs/c2 comms/stage 2 binaries, etc are XOR'ed but the key is at a specific offset in the stream.

#6 - 06/03/2020 05:35 PM - Simon Dugas
Here is a first attempt to implement this feature:
- https://github.com/OISF/suricata/pull/5015
- https://github.com/OISF/suricata-verify/pull/243

The only difference with the syntax discussed in this issue is the xor key is surrounded in double-quotes when specifying a hex string. This allows us to distinguish it from a byte_extract variable.

#7 - 06/08/2020 01:16 PM - Victor Julien
- Status changed from Feedback to In Review
- Assignee changed from Community Ticket to Simon Dugas

#8 - 11/14/2020 02:25 PM - Jeff Lucovsky
- Related to Task #4097: Suricata 2020 brainstorm added

#9 - 11/19/2020 05:36 PM - Victor Julien
The idea at the 2020 brainstorm call was:

extend byte_extract to allow arbitrary length extract for the key xor transform keyword that can either take the variable name from byte extract or a static key as input

#10 - 11/24/2020 09:47 AM - Victor Julien
- Subject changed from XOR keyword to rules: XOR keyword
- Target version changed from TBD to 7.0rc1

#11 - 08/30/2021 03:45 PM - Victor Julien
Hi Simon, have you looked into doing this as a transform?

#12 - 09/15/2021 07:23 AM - Victor Julien
- Status changed from In Review to In Progress

#13 - 10/21/2021 09:53 PM - Victor Julien
- Related to Task #4762: Suricon 2021 brainstorm added

#14 - 10/22/2021 05:52 PM - Philippe Antoine
- Status changed from In Progress to In Review

https://github.com/OISF/suricata/pull/6513

#15 - 10/27/2021 12:20 PM - Simon Dugas
Last time I was looking into transforms they didn't seem to support a "sticky buffer" on the entire TCP payload or holding on to variables such as keys. That was a while ago and I think the transforms API may have improved since then, I'll have a look at catenacyber's PR. I have plenty of test cases and suricata-verify tests that could be useful.

My apologies for the late response.

#16 - 11/24/2021 03:43 PM - Victor Julien
- Assignee changed from Simon Dugas to Philippe Antoine

#17 - 01/19/2022 12:43 PM - Philippe Antoine
- Status changed from In Review to Closed

https://github.com/OISF/suricata/pull/6579

Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>8d44f3fd-110c-423f-82be-c09a1a52af08.pcap</td>
<td>4.91 MB</td>
<td>12/12/2019</td>
<td>Brandon Murphy</td>
</tr>
</tbody>
</table>