Suricata - Feature #5245
allow fast_pattern on base64_data strings
04/08/2022 01:32 PM - Brandon Murphy

Status: New
Priority: Normal
Assignee: OISF Dev
Category: TBD
Assignee: OISF Dev

Description
As referenced within issue 5220 - the engine today does not use nor error out when a fast_pattern is provided on a content within the base64_data buffer. 5220 requests that the engine error out when this condition occurs. This ticket requests this as a feature requests and supported in a way which would not result in FNs.

Sometimes when writing rules for traffic which is base64 encoded, this is no strong fast_pattern to use besides that which is base64 encoded. However, due to the nature of base64 encoding, any static string can be found within base64 encoding in 3 variations. These 3 variations of a base64 encoded string, result from the different offsets in which 8 bit values align with 6 bit (base64) groupings, and can be found for testing purposes in this cyberchef recipe.
https://gchq.github.io/CyberChef/#recipe=Show_Base64_offsets('A-Za-z0-9%2B/%3D',true,'Raw')&input=c29tZXRoaW5n
This results in 3 different IDS rules being created. One for each of the "base64 offsets".

However, one way to work with this, would be for the IDS engine to create the 3 different base64 encoded strings for the "plain text" string and use them as a fast_pattern. This likely violates some current constructs such as, a rule can only have one fast_pattern, but it'd be pretty bad ass if this didn't require 3 rules just to get a good fast_pattern, and instead the engine just took care of it.

Current Method of covering this.

alert http $HOME_NET any -> $EXTERNAL_NET any (flow:established,to_server; http.method; content:"POST"; http.request_body; content:"c29tZXRoW5n"; fast_pattern; base64_decode:bytes 28; base64_data; content:"something"; classtype:bad-unknown; sid:1; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (flow:established,to_server; http.method; content:"POST"; http.request_body; content:"NvbWV0aGlz"; fast_pattern; base64_decode:bytes 28; base64_data; content:"something"; classtype:bad-unknown; sid:2; rev:1;)
alert http $HOME_NET any -> $EXTERNAL_NET any (flow:established,to_server; http.method; content:"POST"; http.request_body; content:"zb21ldGhpbm"; fast_pattern; base64_decode:bytes 28; base64_data; content:"something"; classtype:bad-unknown; sid:3; rev:1;)

Proposed Method:

alert http $HOME_NET any -> $EXTERNAL_NET any (flow:established,to_server; http.method; content:"POST"; http.request_body; base64_decode:bytes 28; base64_data; content:"something"; fast_pattern; classtype:bad-unknown; sid:123; rev:1;)

With the proposed method, the engine would determine the 3 different offsets, apply them as a fast_pattern to the rule, and work as expected. In this particular case, the base64_decode as a 28 byte limit. the fast_pattern could be created in a way to leverage this so that it applies a depth:28 to each of the fast_patterns as well.

Related issues:
Related to Bug #5220: fast_pattern specification in base64_data shouldn't be... Assigned

History
#1 - 06/10/2022 09:10 AM - Victor Julien
- Related to Bug #5220: fast_pattern specification in base64_data shouldn't be allowed added

07/17/2022