coreruleset / coreruleset Public

Code Issues 94 Pull requests 25 Actions Projects Wiki

New issue

Jump to bottom

SQLi using set var at PL2 #1793

Merged dune73 merged 2 commits into `coreruleset:v3.3/dev` from `theMiddleBlue:fix-sqli-bypass-set-var` on Dec 7, 2020

Conversation 10 Commits 2 Checks 0 Files changed 2

theMiddleBlue commented on Jun 8, 2020 Member

Referring to #1727 this new rule tries to block SQLi payload that uses MySQL set variable syntax at PL1.

`detect sqli using set var` 1a6e9e0

fzipi mentioned this pull request on Jul 4, 2020

Monthly Chat Agenda July (2020-07-06) #1836

Closed

Taiki-San commented on Jul 8, 2020 Contributor

Looked a bit into this rule and ran it on some trafic: it doesn't trigger too much but still raise a few false positives on passwords. We're also seeing a few false positives on semi-formatted data from the POST body but overall, it looks fine on our end.
I feel the rule could be more specific (doesn't catch all `set bla = 0` syntaxes, ignore the `set` part) but it tackle the original report and isn't harmful on the way.

theMiddleBlue commented on Jul 13, 2020 • edited Member Author

> still raise a few false positives on passwords

Sorry, I can't understand this statement. Can you attach an example? A password should be a free input text and if a user chooses a password like `exec(/bin/bash);'+OR+1=1--` it may trigger the whole rule-set...

> We're also seeing a few false positives on semi-formatted data from the POST body

Please, if you can share them it would be really helpful to improve this rule.

> doesn't catch all set bla = 0 syntaxes, ignore the set part

Yes, because (IMO) `set bla = 0` isn't something we would like to block at PL1. As you can see on #1727 is easy to bypass libinjection and our PL1 by using both `@var:=<sql>` and ``{`label`<sql>}`` syntaxes. Based on my tests, this is not true by using `set variable = <sql>` syntax. If you can provide a different example of exploiting 1727 PoC by using this syntax it would be really helpful.

Taiki-San commented on Jul 13, 2020 • edited Contributor

We run it a bit further and it actually started to trigger on scans :D Appear to overlap slightly with 942190, but it may be because it was a complex payload.

> Sorry, I can't understand this statement. Can you attach an example?

The actual exemples are redacted, but not too hard to imagine: you just need to following expression somewhere in the password: `@[\w\d]+\=\S` . For exemple, `@4=d` which could quite likely appear in a randomly generated password. The password you suggested is deliberately malicious, the exemple we're having here is similar to `sh` somewhere in a randomly generated string triggering an SHI rule.
I agree that free text is a challenge but we can't completely ignore it: contextualizing the rules enable fewer edge cases and wider usage. For some rules, it's highly complex, for this one it doesn't feel so.

> Please, if you can share them it would be really helpful to improve this rule

I'm seeing payloads like this, although not sure where they come from: `redacted@blablav=website.com`

> Based on my tests, this is not true by using set variable = syntax

Thanks a lot for the rationale, that makes a ton of sense! Could you add it in the rule's header to explain where this design came from? Will likely be useful in the future!

theMiddleBlue commented on Jul 13, 2020 Member Author

> The password you suggested is deliberately malicious

Yes but this is not only related to this rule. It is true for **all** CRS libinjection rules or XSS rules when a random string or base64 encoded string is provided. For example `random/onrandom==` triggers 941100 and 941120 at PL1. Those two rules should have the same behavior for you.

Taiki-San commented on Jul 13, 2020 Contributor

I agree, but when possible we should try to minimize it.
`941120` is a good exemple of a rule that we deemed too sensitive and since disabled by default to our users.

My point isn't to be completely resilient from false positives in free text (I agree it'll be tricky) but to contextualize patterns a bit when there are low hanging fruits.

Is your point that we should keep the patterns as broad as possible, or that what I'm identifying as a low hanging fruit (looking for a `set` prefix) actually isn't?

dune73 commented on Aug 3, 2020 Contributor

I'm a bit late to the show, but here is the summary of the discussion about this PR during the project chat in July 2020.

"We want better documentation, better / more tests and the rule stays in PL2 kindergarden until it is proven to have only very few FPs. It can still be in PL1 for 3.4, but per default it goes to PL2."

dune73 mentioned this pull request on Aug 3, 2020

Monthly Chat Agenda August (2020-08-03) #1853

Closed

dune73 commented on Sep 7, 2020 Contributor

@theMiddleBlue : Could you shift this to PL2 please. We'll merge as soon this is done.

franbuehler mentioned this pull request on Sep 7, 2020

Monthly Chat Agenda September (2020-09-07) #1869

Closed

dune73 added the 👀 **Needs action** label on Sep 7, 2020

dune73 mentioned this pull request on Oct 5, 2020

Monthly Chat Agendas October (2020-10-05 and 2020-10-19) #1892

Closed

lifeforms assigned **theMiddleBlue** on Oct 5, 2020

lifeforms commented on Oct 5, 2020 Member

TODO from meeting:

- theMiddleBlue can shift to PL2 and we merge on next meeting.

`move 942520 to PL2` `909cab5`

theMiddleBlue changed the title ~~SQLi using set var at PL1~~ **SQLi using set var at PL2** on Oct 19, 2020

theMiddleBlue commented on Oct 19, 2020 Member Author

rule moved to PL2

theMiddleBlue removed the 👀 **Needs action** label on Oct 19, 2020

dune73 mentioned this pull request on Nov 2, 2020

Monthly Chat Agendas November (2020-11-02 and 2020-11-16) #1916

Closed

franbuehler mentioned this pull request on Dec 5, 2020

Monthly Chat Agendas December (2020-12-07 and 2020-12-21) #1944

Closed

dune73 commented on Dec 7, 2020 Contributor

This was supposed to be merged after the November meeting. Time to do this. Thank you **@theMiddleBlue**.

👍 1

dune73 merged commit **874d81c** into `coreruleset:v3.3/dev` on Dec 7, 2020

Reviewers

No reviews

Assignees

theMiddleBlue

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

4 participants