Weiterentwicklung des OWASP ModSecurity Core Ruleset

OWASP Stammtisch München – 2018-09-18

Mirko Dziadzka
mirko@owasp.org

Wer bin ich

- Developed WAF like components for banking in 2000
- Worked for WAF vendor(s) as lead engineer from 2005-2017
- Currently work for Avi Network on their WAF
- Co-author of OWASP Paper:
    - Best Practices: Einsatz von Web Application Firewalls
- Diverse Vorträge auf diesem OWASP Stammtisch:
    - https://blog.mirko-dziadzka.de/search/label/OWASP

Worum geht‘s

- Die bekannteste Open Source WAF ist ModSecurity (2.x und 3.x)
- Das Standard Ruleset für diese WAF ist das OWASP Core Rule Set
- Ich würde gerne das Wissen aus diesem Core Rule Set für andere „Web Application Firewalls“ verfügbar machen.
  - Andere OpenSource WAF
  - Kommerzielle WAFs
  - Application Input Filter(Django Middleware)
- Dazu muss dieses in einer anderen Sprache formuliert werden.

OWASP ModSecurity Core Rule Set

OWASP
ModSecurity
Core Rule Set

THE 1ST LINE OF DEFENSE

OWASP ModSecurity Core Rule Set

- Aktive aber kleine Community
- Hat sich erstmals in London am Rande der AppSecEU auf dem CRS Summit getroffen

1st. Core Rule Set Summit

Core Rule Set Summit

- Community wird wieder aktiver
- Regelmäßige Major Releases (3.1 steht vor der Tür)
- Zusammenarbeite mit anderen Organisationen die 0-day patching anbieten
- Ideen zur Weiterentwicklung

OWASP ModSecurity Core Rule Set

Das Core Rule Set besteht im wesentlichen aus zwei Teilen:

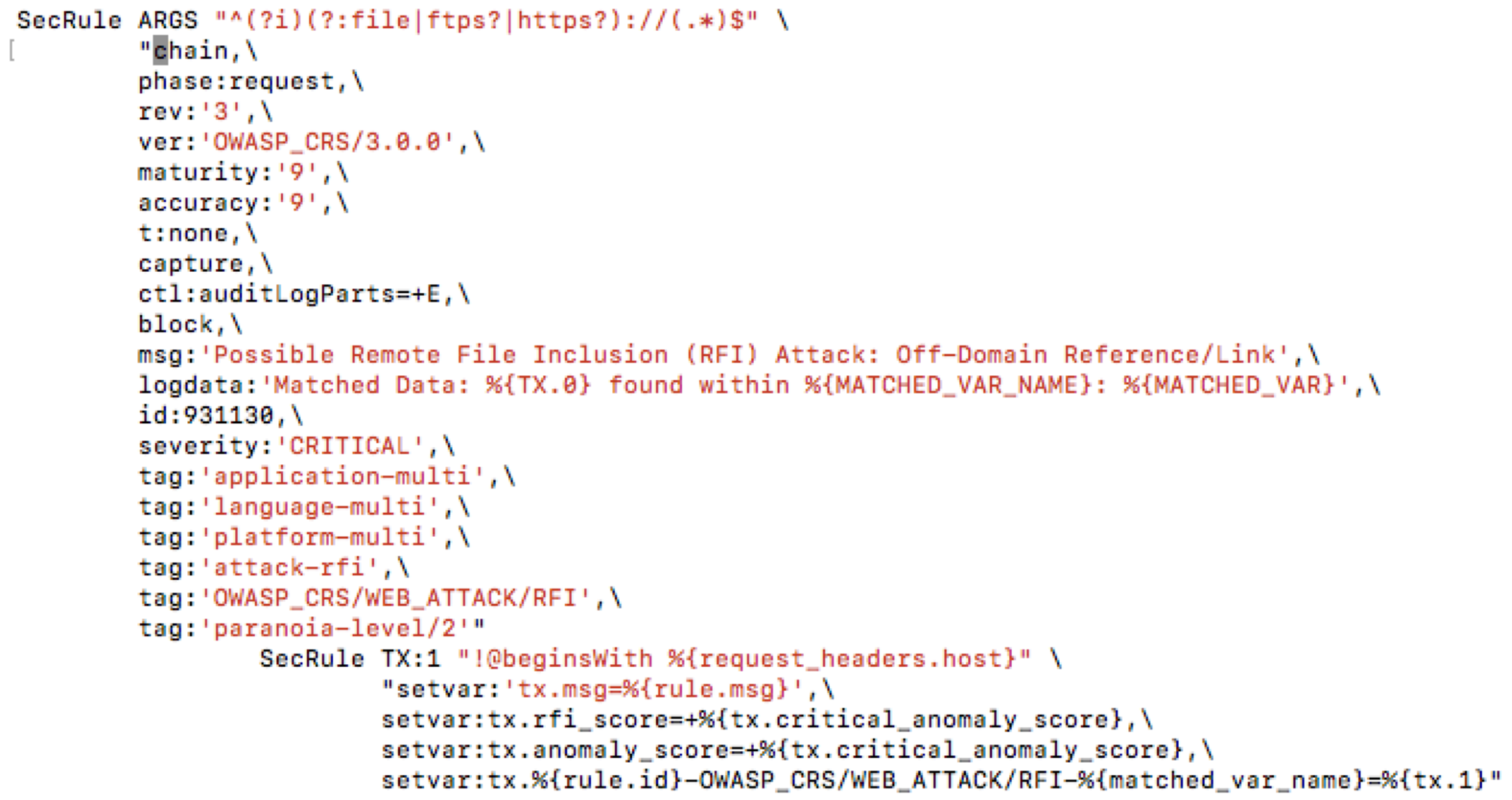
1. Implementiert grundlegende WAF Funktionalität für ModSecurity
2. Implementiert ein Ruleset um Angriffe zu erkennen

Mir geht es hier um den Punkt 2. Das CRS kodiert Wissen der Community wie Angriffe erkannt werden können.

Das Core Rule Set ist in der ModSecurity Sprache verfasst. Diese ist nicht gut gealtert.

Auch wenn es inzwischen alternative Implementationen gibt, ist diese Sprache nicht für generische Application Firewalls geeignet.

OWASP ModSecurity Core Rule Set



```
SecRule ARGS "^(?i)(?:file|ftps?|https?)://(.*)$" \
        "chain,\
        phase:request,\
        rev:'3',\
        ver:'OWASP_CRS/3.0.0',\
        maturity:'9',\
        accuracy:'9',\
        t:none,\
        capture,\
        ctl:auditLogParts=+E,\
        block,\
        msg:'Possible Remote File Inclusion (RFI) Attack: Off-Domain Reference/Link',\
        logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',\
        id:931130,\
        severity:'CRITICAL',\
        tag:'application-multi',\
        tag:'language-multi',\
        tag:'platform-multi',\
        tag:'attack-rfi',\
        tag:'OWASP_CRS/WEB_ATTACK/RFI',\
        tag:'paranoia-level/2'"
                SecRule TX:1 "!@beginsWith %{request_headers.host}" \
                        "setvar:'tx.msg=%{rule.msg}',\
                        setvar:tx.rfi_score=+%{tx.critical_anomaly_score},\
                        setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},\
                        setvar:tx.%{rule.id}-OWASP_CRS/WEB_ATTACK/RFI-%{matched_var_name}=%{tx.1}"
```
]

Idee

Formulierung des Core Rule Sets in einer Sprache die nicht mehr an die ModSecurity Semantik gebunden ist.

Darüber wurde schon oft diskutiert, die Zeit dafür scheint reif zu sein.

Ziel ist es, das Regelwerk für andere Infrastruktur zugänglich zu machen.

Towards a better language for WAF Core Rule Set

CRS community summit 2018

Christian Treutler, Mirko Dziadzka
{christian,mirko}@avinetworks.com

Agenda

- Who we are
- Why are we doing this?
- Why not ModSecurity[*]?
- Ideas and Concepts
- Current status and next steps
- Questions and Discussion

* – This always means: ModSecurity – the language, not ModSecurity - the WAF Implementation

Who we are

- Mirko
  - Developed WAF like components for banking in 2000
  - Worked for WAF vendor(s) as lead engineer from 2005-2017
  - Co-author of OWASP Paper: Use of Web Application Firewalls
- Christian
  - Worked as Engineer + Product Manager for WAF vendor(s) since 2007
- Since 2017 working for Avi Networks WAF team
  - WAF is based on ModSecurity 3.x with some heavy changes
  - WAF is using OWASP CRS for base protection

Why are we doing this?

- ModSecurity* is not ideal for a WAF for various reasons
- The Core Rule Set is a valuable resource, both for commercial and for other open-source WAF‘s[2]
- The Core Rule Set should be in a format which can be consumed by other WAF implementations[2]

[2] for example lua-resty-waf, or a middleware input validation layer in your Django Web Stack

Why not ModSecurity*?

- Assumptions:
  - WAF should be **configured** in a **declarative** style and not programmed.
  - In special cases, you need the power of a scripting language.
- ModSecurity* is too complex to be considered configuration or to automatically convert it to other execution models.
- It is not flexible enough to solve more complex problems. Lua support solves this.
- If you want to read my full rant about it, read [1] or talk to me in the next 2 days.

Why not ModSecurity*?

- Syntax – should never show this to a user
- Types (or the absence of types)
  - List of strings is really missed
  - No clear distinction between Number and String leads to subtle errors
- Chain rules
  - Mostly used to
    - a) extract part of the Requests into temp variables
    - b) implement a logical AND for conditions
- Regex is PCRE based
  - CRS is using PCRE extensions which are not available in other implementations, for example python-re or Google re2 lib

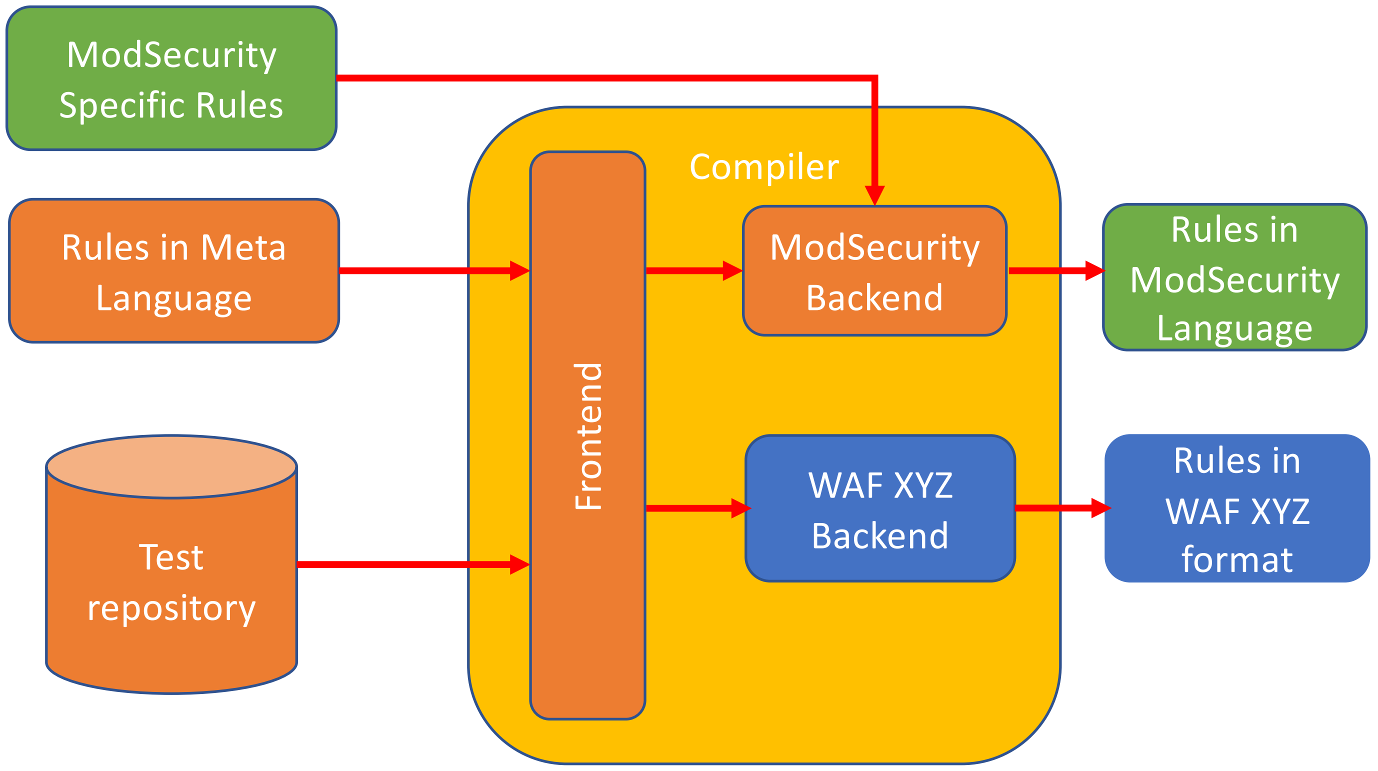
Ideas and Concepts

- Current CRS contains rules which implement WAF functionality but are not part of what we would consider a WAF CRS
  - The concrete way of how the features below are implemented
    - IP reputation, DOS protection
    - anomaly detection, rule corellation
    - Sampling, logging
- We believe that the other “interesting rules“ can be expressed in an easier and more declarative language.
- This is not a language to configure a specific WAF directly

Ideas and Concepts - Summary

- Have a declarative language which can describe (a subset of) current Core Rule Set
- Have compiler to automatically convert rules from this language to different WAF‘s native languages, like ModSecurity.
    - The goal is to create exactly the same rules CRS has today. This may need some backend specific hints in the rules for the compiler.

Ideas and Concepts - Summary



Ideas and Concepts – Language Spec

- Building Blocks are pluggable.
  - Constants (maybe request dependent). Do we need variables?
    - For the sake of discussion: A variable which is only set once is a constant.
  - Conditions
  - Actions
  - Rules
  - Control flow (explicit)
- Avoid state and state modification (get rid of setvar as much as possible)
- We do not want to discuss syntax right now, so we use YAML for all the examples.

Ideas and Concepts - Constants

```yaml
- define:
    name: max_body_size
    type: int
    value: 32k

- define:
    name: restricted_extensions
    type: [string]
    value:
        - "asa"
        - "asax"
        - ...
        - "xsd"
        - "xsx"
    transformation:
        - ".%{$1}"

- define:
    - name: unix_shell_data
    - type: [string]
    - load: "unix-shell.data"
```

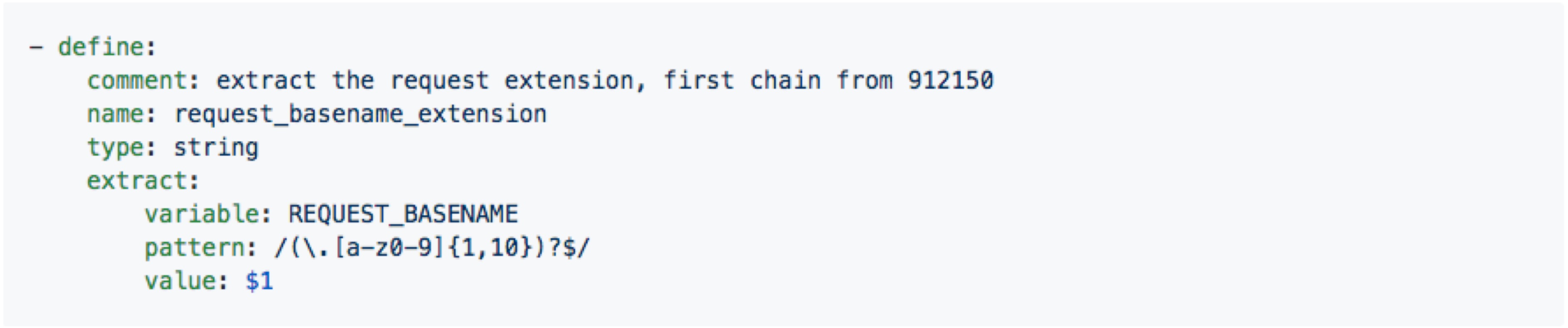
Ideas and Concepts - Constants

```
- define:
    name: max_body_size
    type: int
    value: 32k

- define:
    name: restricted_extensions
    type: [string]
    value:
        - "asa"
        - "asax"
        - ...
        - "xsd"
        - "xsx"
    transformation:
        - ".%{$1}"

- define:
    - name: unix_shell_data
    - type: [string]
    - load: "unix-shell.data"
```

```
SecRule &TX:restricted_extensions "@eq 0" \
   "id:901164, phase:1, pass, nolog,\
   setvar:'tx.restricted_extensions=.asa/ .asax/ .ascx/ .axd/ .backup/
.bak/ .bat/ .cdx/ .cer/ .cfg/ .cmd/ .com/ .config/ .conf/ .cs/ .csproj/ .csr/
.dat/ .db/ .dbf/ .dll/ .dos/ .htr/ .htw/ .ida/ .idc/ .idq/ .inc/ .ini/ .key/ .licx/
.lnk/ .log/ .mdb/ .old/ .pass/ .pdb/ .pol/ .printer/ .pwd/ .resources/
.resx/ .sql/ .sys/ .vb/ .vbs/ .vbproj/ .vsdisco/ .webinfo/ .xsd/ .xsx/'"
```

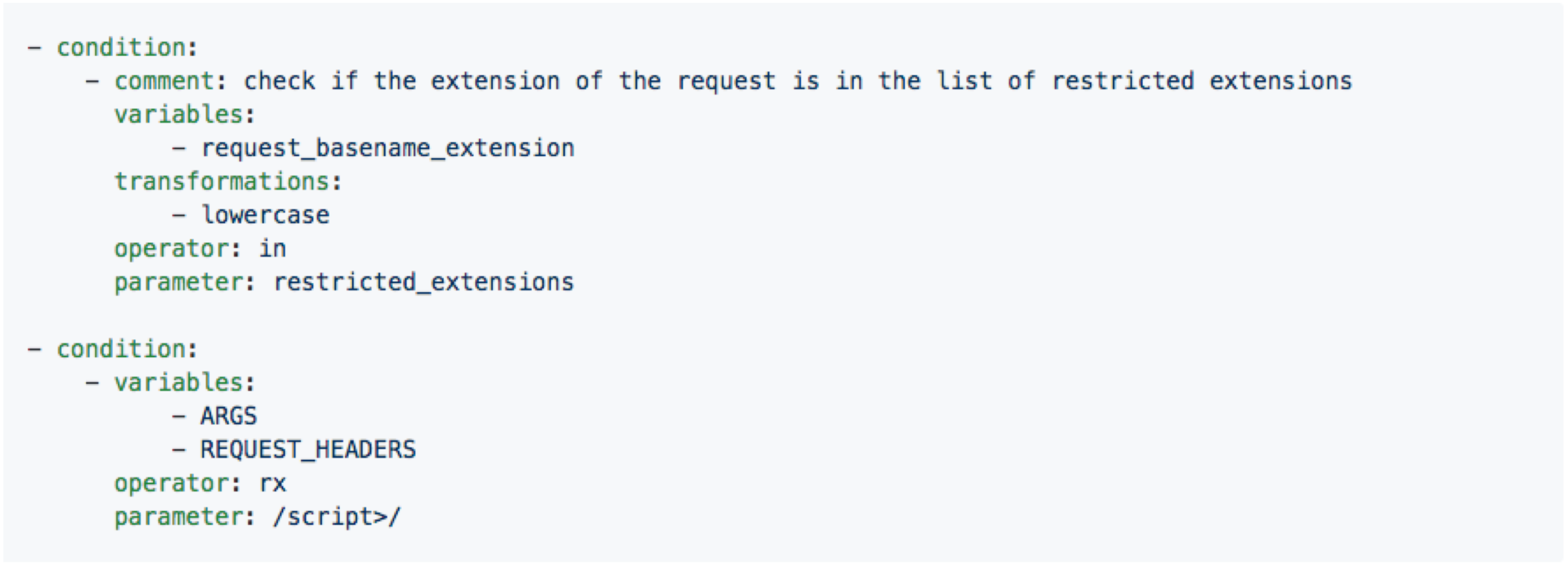
Ideas and Concepts – Extract Data

- Chain rules are often used to extract data from the request
- This should be explicit



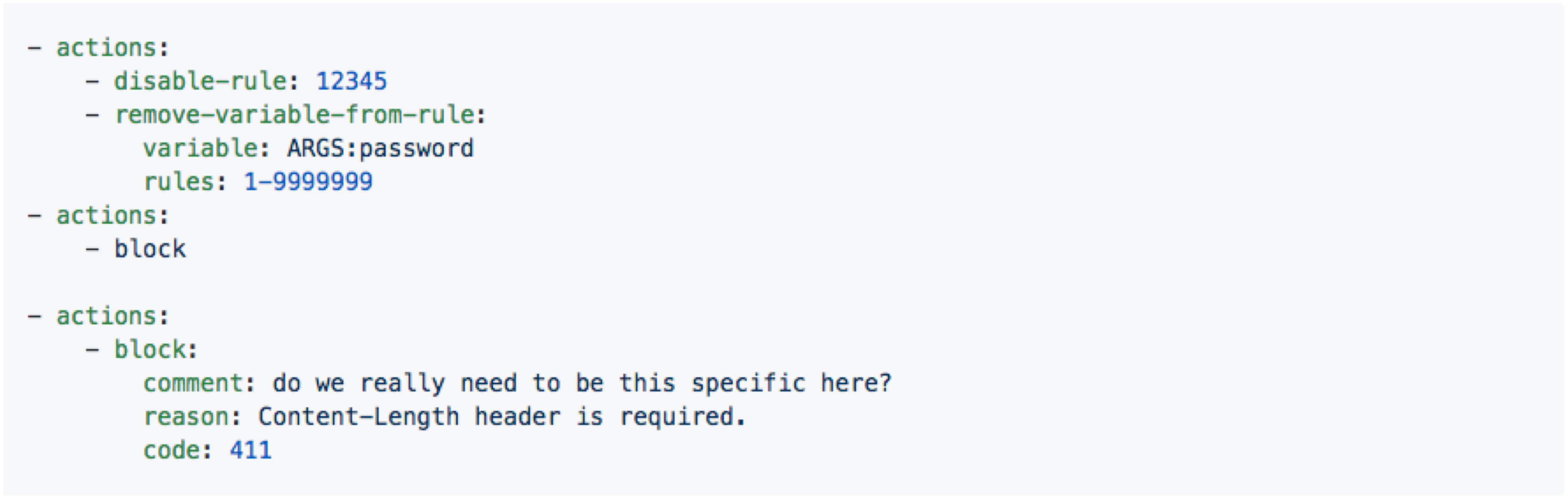
```
- define:
    comment: extract the request extension, first chain from 912150
    name: request_basename_extension
    type: string
    extract:
        variable: REQUEST_BASENAME
        pattern: /(\.[a-z0-9]{1,10})?$/
        value: $1
```

Ideas and Concepts – Conditions



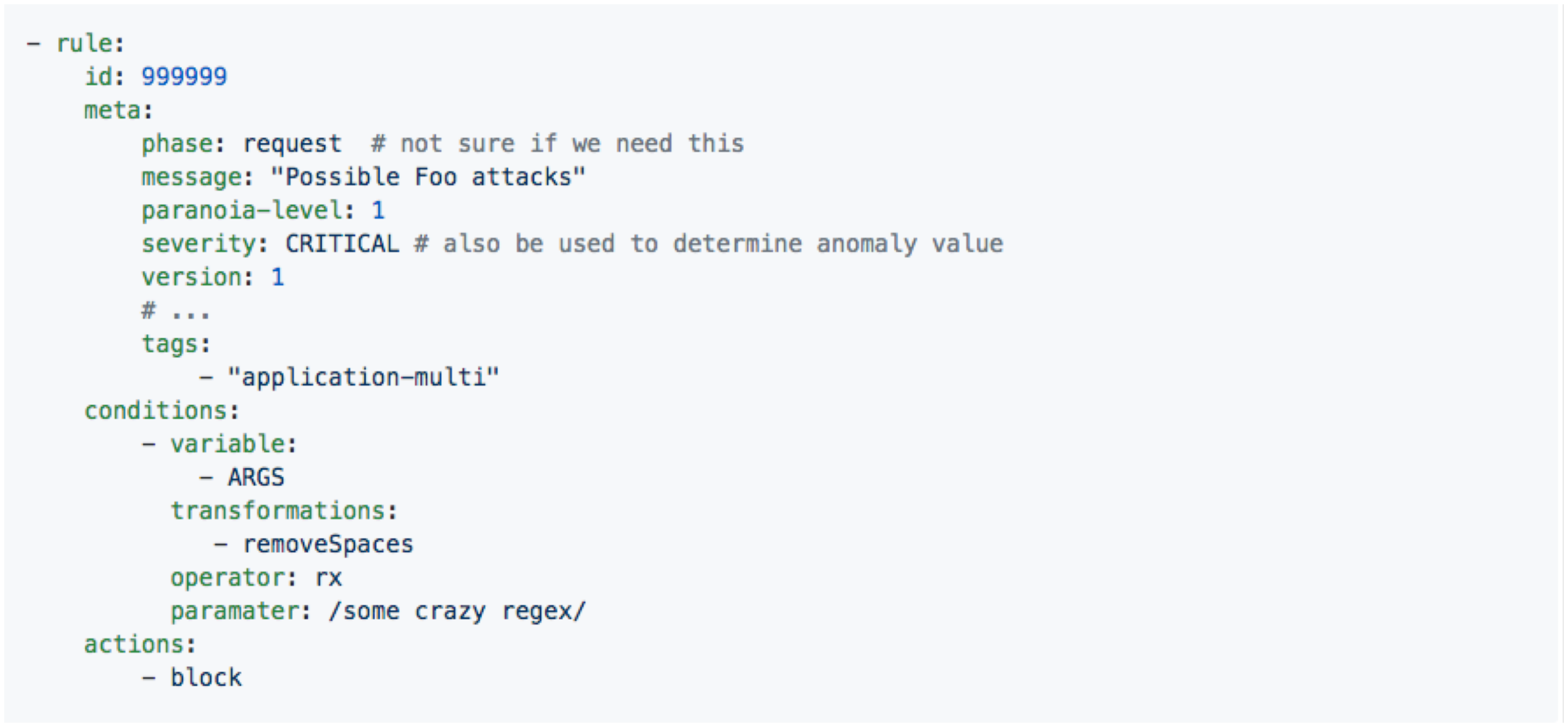
```
- condition:
    - comment: check if the extension of the request is in the list of restricted extensions
      variables:
          - request_basename_extension
      transformations:
          - lowercase
      operator: in
      parameter: restricted_extensions

- condition:
    - variables:
          - ARGS
          - REQUEST_HEADERS
      operator: rx
      parameter: /script>/
```

Ideas and Concepts – Actions



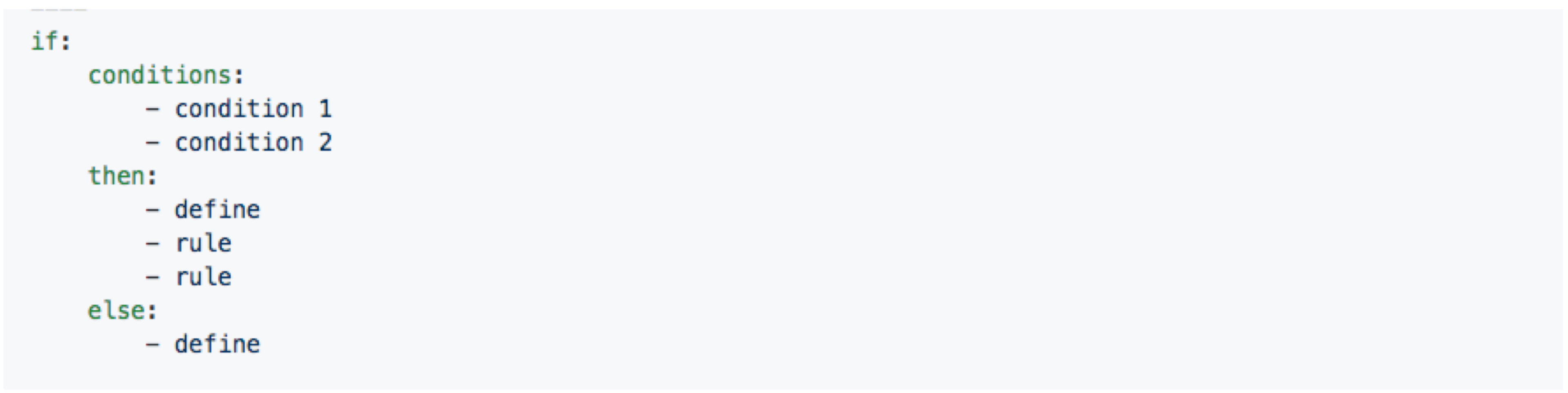
```yaml
- actions:
    - disable-rule: 12345
    - remove-variable-from-rule:
        variable: ARGS:password
        rules: 1-9999999
- actions:
    - block

- actions:
    - block:
        comment: do we really need to be this specific here?
        reason: Content-Length header is required.
        code: 411
```

Ideas and Concepts – Rules



```yaml
- rule:
    id: 999999
    meta:
        phase: request  # not sure if we need this
        message: "Possible Foo attacks"
        paranoia-level: 1
        severity: CRITICAL # also be used to determine anomaly value
        version: 1
        # ...
        tags:
            - "application-multi"
    conditions:
        - variable:
            - ARGS
          transformations:
             - removeSpaces
          operator: rx
          paramater: /some crazy regex/
    actions:
        - block
```

Ideas and Concepts – Control Flow



```
if:
    conditions:
        - condition 1
        - condition 2
    then:
        - define
        - rule
        - rule
    else:
        - define
```

Current status and next steps

- Proposal for language semantics, needs validation and iteration
- Python lib which can convert between ModSecurity* rule format, an internal object representation of these rules and an equivalent JSON format.
- TODO:
    - semi-automatically translating ModSecurity rules from the ModSecurity* representation to the new meta language, apply this to a subset of current CRS
    - translator from this meta-language to ModSecurity*
    - Implement an engine in Python to execute these new language directly as proof of concept and for test integrations

Open Questions

- Will it work?
- What about a positive security model?
- What about test integration into FTW
- How make the regex more readable?
    - Having a “readable version“ and a automatically generated „optimized“ regex may help.
    - Also, integrating test strings to the regexes which should or should not match could help as better documentation

Questions and Discussion

- [1] https://github.com/avinetworks/owasp-crs-technical-discussion
- We would welcome any feedback and contributions
- We would love to talk to you about this or other ideas in the next couple of days