OWASP ModSecurity Core Rule Set (CRS) Project

November 8 - 11 2010

Ryan Barnett
OWASP CRS Project Leader
Senior Security Researcher

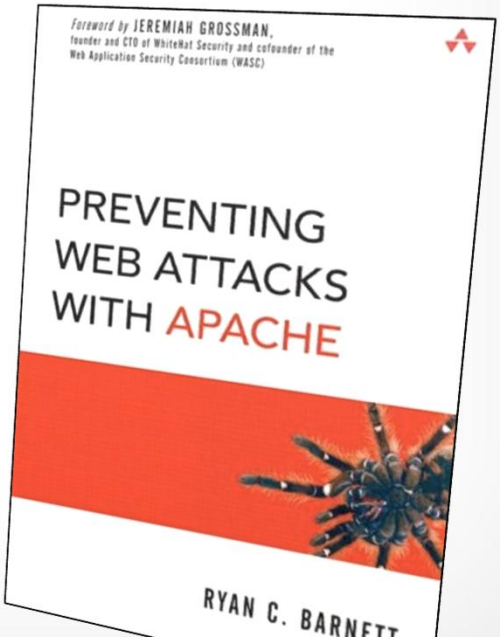
Trustwave®
SpiderLabs®

Trustwave

- **SpiderLabs Research Team**
  - Web application firewall research/development
- **ModSecurity Community Manager**
  - Interface with the community on public mail-list
  - Steer the internal development of ModSecurity

Author

- “Preventing Web Attacks with Apache”



- **Open Web Application Security Project (OWASP)**
  - Project Leader, ModSecurity Core Rule Set
  - Project Contributor, OWASP Top 10
  - Project Contributor, AppSensor
- **Web Application Security Consortium (WASC)**
  - Project Leader, Web Hacking Incident Database
  - Project Leader, Distributed Web Honeypots
  - Project Contributor, Web Application Firewall Evaluation Criteria
  - Project Contributor, Threat Classification
- **The SANS Institute**
  - Courseware Developer/Instructor
  - Project Contributor, CWE/SANS Top 25 Worst Programming Er

Session Outline

- Обзор ModSecurity
- Обзор Core Rule Set (CRS)
- Основные категории определения уязвимостей
- Дальнейшее совершенствование CRS
- CRS Demonstration Page
- Дальнейшие направления развития

Обзор ModSecurity

Open Source Web Application Firewall

- Это open source web application firewall (WAF) module для Apache web servers
  - www.modsecurity.org
- Разделение Rule и Audit Engines
  - Возможности создания логов полного запроса/ответа HTTP
- Глубокий анализ HTTP и HTML
  - Надежный Parsing (form encoding, multipart, XML)
- Язык Rules основан на событиях
  - Возможности предотвращения обхода (функции нормализации)
- Дополнительные возможности
  - Transactional and Persistent Collections

Синтаксис языка ModSecurity's Rules

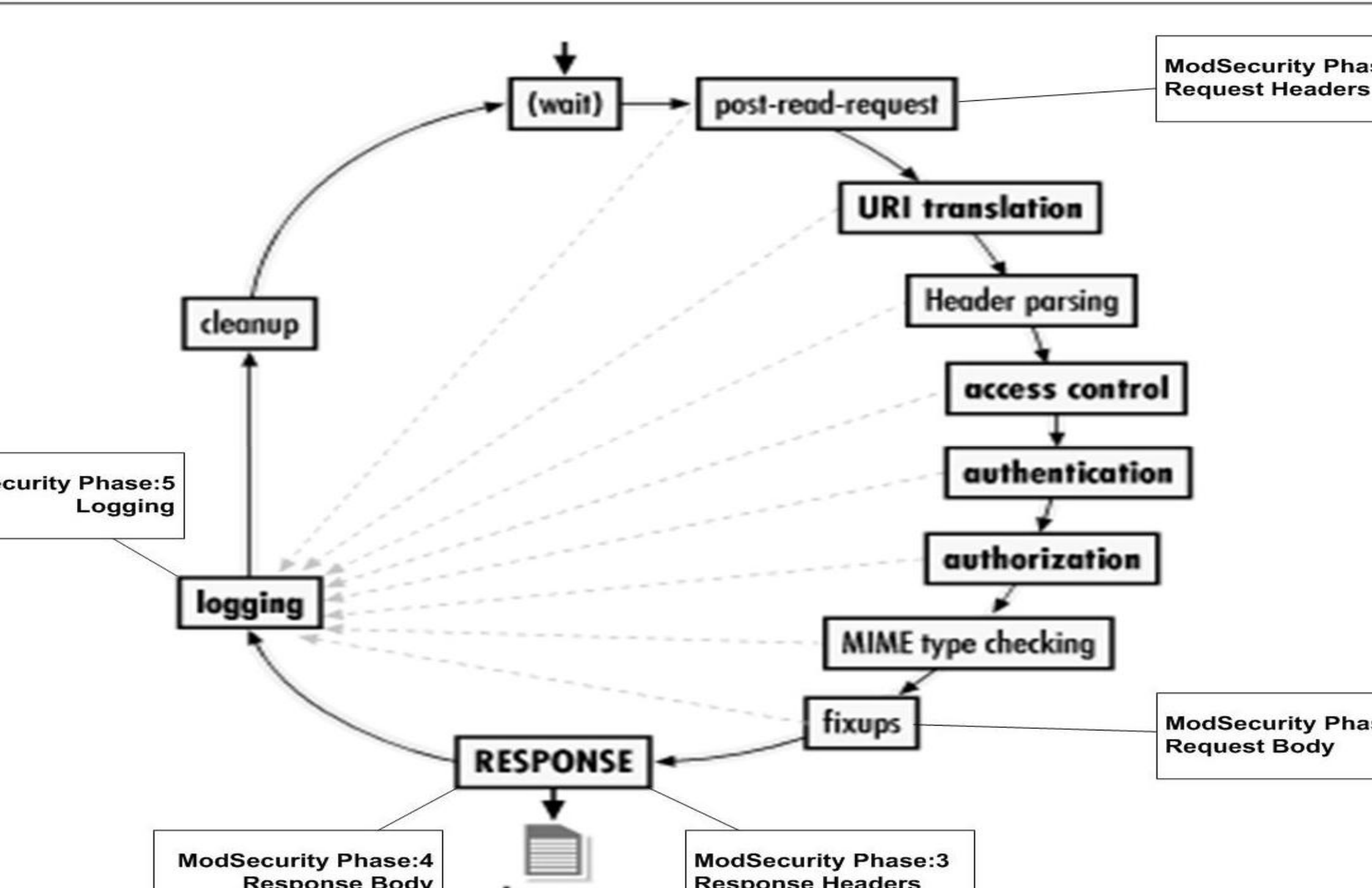
Говорит ModSecurity ***как обрабатывать данные*** (например @rx, @pm or @gt).

```
SecRule TARGETS OPERATOR [ACTIONS]
```

Говорит ModSecurity ***где просматривать*** (например ARGS, ARGS_NAMES or COOKIES)

Говорит ModSecurity ***что делать***, если правило соответствуе пакету (например den exec or setvar)

запроса ModSecurity's Apache



бзор базового набора прав
WASP ModSecurity (Core Ru
Set - CRS)

Информация о проекте

category | discussion | view source | history

Category:OWASP ModSecurity Core Rule Set Project

Home | Download | Bug Tracker | Demo | Installation | Documentation | Presentations and Whitepapers | Related Projects | Latest News and Mail List | Contributors, Users and Adopters | Project About

Overview

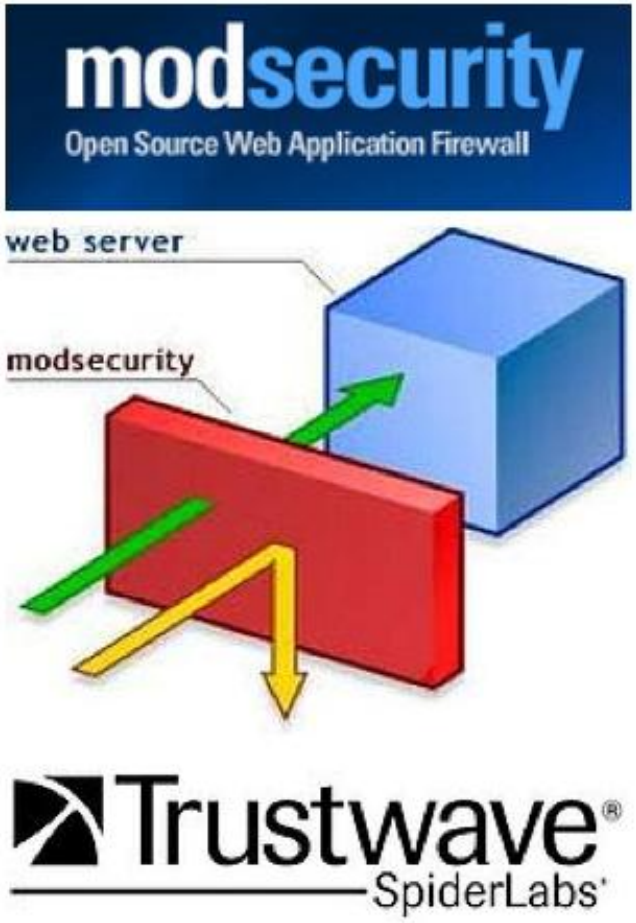
ModSecurity is an Apache web server module that provides a web application firewall engine. The ModSecurity Rules Language engine is extrememly flexible and robust and has been referred to as the "Swiss Army Knife of web application firewalls." While this is certainly true, it doesn't do much implicitly on its own and requires rules to tell it what to do. In order to enable users to take full advantage of ModSecurity out of the box, we have developed the **Core Rule Set (CRS)** which provides critical protections against attacks across most every web architecture.

Unlike intrusion detection and prevention systems, which rely on signatures specific to known vulnerabilities, the CRS is based on generic rules which focus on attack payload identification in order to provide protection from zero day and unknown vulnerabilities often found in web applications, which are in most cases custom coded.

Detection Categories

In order to provide generic web applications protection, the Core Rules use the following techniques:

- Protocol compliance:
  - HTTP request validation - This first line of protection ensures that all abnormal HTTP requests are detected. This line of defense eliminates a large number of automated and non targeted attacks as well as protects the web server itself.



It's like core Rule Set (CRS):

щий, plug-n-play набор WAF-правил

зволяет выбирать режим выполнения

Стандартный vs. Определение аномалий

тегории обнаружения:

Корректность протокола

Вредоносная идентификация клиента

Сигнатуры общих атак

Сигнатуры известных уязвимостей

Доступ Trojan/Backdoor

Утечка данных вовне

Утилиты и скрипты антивируса и DoS

```
./base_rules:
modsecurity_40_generic_attacks.data
modsecurity_41_sql_injection_attacks.
modsecurity_46_et_sql_injection.data
modsecurity_46_et_web_rules.data
modsecurity_50_outbound.data
modsecurity_crs_20_protocol_violation
modsecurity_crs_21_protocol_anomalies
modsecurity_crs_23_request_limits.con
modsecurity_crs_30_http_policy.conf
modsecurity_crs_35_bad_robots.conf
modsecurity_crs_40_generic_attacks.co
modsecurity_crs_41_phpids_converter.c
modsecurity_crs_41_phpids_filters.con
modsecurity_crs_41_sql_injection_att
modsecurity_crs_41_xss_attacks.conf
modsecurity_crs_45_trojans.conf
modsecurity_crs_46_et_sql_injection.c
modsecurity_crs_46_et_web_rules.conf
modsecurity_crs_47_common_exceptions.
modsecurity_crs_48_local_exceptions.c
modsecurity_crs_49_enforcement.conf
modsecurity_crs_50_outbound.conf
modsecurity_crs_60_correlation.conf

./optional_rules:
modsecurity_crs_20_protocol_violation
modsecurity_crs_21_protocol_anomalies
modsecurity_crs_40_generic_attacks.co
modsecurity_crs_42_comment_spam.conf
modsecurity_crs_42_tight_security.con
modsecurity_crs_55_marketing.conf
```

Начальная конфигурация

После распаковки следует отредактировать основной конфигурационный файл

- `modsecurity_crs_10_config.conf`

Настроить следующие элементы

- Способ обнаружения – стандартный vs. Определение аномалий
- Уровни строгости определения аномалий
- Необходимость блокировки (Enable/Disable)
- Уровни порогов блокировок
- Paranoid режим – агрессивная инспекция
- Установки политики HTTP
- Выбор, где хранить логи событий (Apache error_log и/и

Концепция «самодостаточных» правил

S/IPS режим с “самодостаточными” правилами

одобно самому HTTP – правила не поддерживают
стояния

- **Нет разделяемой информации между правилами**
  - Если правило срабатывает, оно будет выполнять disruptive/logging действие
- гкость освоения новыми пользователями
- оптимально с точки зрения управления правила
  учная обработка ложных позитивностей
  ключений)
- оптимально с точки зрения безопасности
  - Не все сайты имеют одинаковые риски безопасности

Концепция совместно выполняющихся прави

Усовершенствованный режим инспектирования/обнаружения

Задержка блокировки

- Правила устанавливают транзакционные переменные (tx) для хранения временных мета-данных о соответствии правилу
- Правила также приведут к возрастанию обнаружений атак, связанных с аномальным поведением

Правила, усиленные аномальными оценками, принимают решение, запрещать ли транзакции при завершении входящего запроса.

- `modsecurity_crs_49_inbound_blocking.cor`

отладочных логов

```
Target value: "" OR 1=1#"
AdResolved macro %{rule.msg} to: SQL Injection AttackSet variable "tx.msg" to "SQL Injection Attack"
Setting variable: tx.sql_injection_score=+%{tx.critical_anomaly_score}
Recorded original collection variable: tx.sql_injection_score = "0"
Resolved macro %{tx.critical_anomaly_score} to: 5
Relative change: sql_injection_score=0+5
Set variable "tx.sql_injection_score" to "5".
Setting variable: tx.anomaly_score=+%{tx.critical_anomaly_score}
Original collection variable: tx.anomaly_score = "3"
Resolved macro %{tx.critical_anomaly_score} to: 5
Relative change: anomaly_score=3+5
Set variable "tx.anomaly_score" to "8".
Setting variable: tx.%{rule.id}-WEB_ATTACK/SQL_INJECTION-%{matched_var_name}=%{tx.0}
Resolved macro %{rule.id} to: 950901
Resolved macro %{matched_var_name} to: ARGS:prod_id
Resolved macro %{tx.0} to: 1=1
Set variable "tx.950901-WEB_ATTACK/SQL_INJECTION-ARGS:prod_id" to "1=1".
Warning. Pattern match "\b(\d+) ?(?:=|<>|<=>|<|>|!=) ?\1\b|[\'"\`\´\’\‘](\d+)[\'"\`\´\’\‘]
?(?:=|<>|<=>|<|>|!=) ?[\'"\`\´\’\‘]\2\b|[\'"\`\´\‘](\w+)[\'"\`\´\’\‘] ?(?:=|<>|<=>|<|>|!=)
?[\'"\`\´\’\‘]\3\b|([\'"\;\`\´\’\‘]*)?\s+(and|or)\s+([\s\'"\` ..." at ARGS:prod_id. [file
"/usr/local/apache/conf/modsec_current/base_rules/modsecurity_crs_41_sql_injection_attacks.conf"]
[line "425"] [id "950901"] [rev "2.0.9"] [msg "SQL Injection Attack"] [data "1=1"] [severity
"CRITICAL"]
ded regex subexpression to TX.0: 1=1Added regex subexpression to TX.1: 1
Operator completed in 19 usec.Ctl: Set auditLogParts to ABIFHZE.Setting variable: tx.msg=%{rule.msg}
```

Изучение аномального поведения

```
Alert and Block based on Anomaly Scores

cRule TX:ANOMALY_SCORE "@gt 0" \
  "chain,phase:2,t:none,nolog,auditlog,deny,msg:'Inbound Anomaly
ore Exceeded (Total Score: %{TX.ANOMALY_SCORE},
Li=%{TX.SQL_INJECTION_SCORE}, XSS=%{TX.XSS_SCORE}):
tx.msg}',setvar:tx.inbound_tx_msg=%{tx.msg},setvar:tx.inbound_an
y_score=%{tx.anomaly_score}"
      SecRule TX:ANOMALY_SCORE "@ge
tx.inbound_anomaly_score_level}" chain
              SecRule TX:ANOMALY_SCORE_BLOCKING "@streq on"
```

ример SQL Injection

регация индикаторов для определения атаки

трогие индикаторы

Ключевые слова, такие как: xp_cmdshell, varchar,

Последовательности, такие как: *union .... select, select ... 1*

Amount: *script, cookie* и *document* появляются в котором поле ввода

лабые индикаторы – мета-символы

-, ;, ', ...

RS используют только слабые сигнатуры

Пример условного правила

```
cMarker BEGIN_SQL_INJECTION_WEAK


cRule &TX:/SQL_INJECTION/ "@eq 0"
hase:2,t:none,nolog,pass,skipAfter:END_SQL_INJECTION_WEAK"


cRule TX:/SQL_INJECTION/
b(?:rel(?:(?:nam|typ)e|kind)|a(?:ttn(?:ame|um)|scii)|c(?:o(?:nver|un)t
s(?:hutdown|elect)|to_(?:numbe|cha)r|u(?:pdate|nion)|d(?:elete|rop)|gr
*\bby|having|insert|length|where)\b" \

    "phase:2,chain,capture,t:none,ctl:auditLogParts=+E,block,nolog,audi
sg:'SQL Injection
tack',id:'950001',tag:'WEB_ATTACK/SQL_INJECTION',logdata:'%{TX.0}',sev
2'"
    SecRule MATCHED_VAR "(?:[\\\(\)\%#]|--)" \
    "t:none,setvar:'tx.msg=%{rule.msg}',setvar:tx.sqli_score=+1,setvar
maly_score=+20,setvar:tx.%{rule.id}-WEB_ATTACK/SQL_INJECTION-
matched_var_name}=%{matched_var}"


cMarker END_SQL_INJECTION_WEAK
```

Стандартные vs. Коррелирующие события

Стандартный режим

- Правила записывают данные о событии как в Apache error_log, так и в ModSecurity Audit log

Коррелирующий режим

- Базовые правила анализируют ссылки и не записывают их непосредственно в Apache error_log
- Коррелирующие правила на фазе создания логов анализируют входящие /исходящие события и создают специальные события
- `modsecurity_crs_60_correlation.conf`

Входящая / исходящая корреляция

Сопоставление входных данных и выходных для улучшенного принятия решений

- Была ли входящая атака?
- Был ли HTTP Status Code Error (4xx/5xx уровень)?
- Была ли утечка прикладной информации?

Возможности корреляции лучше анализируют ответ

- Ошибка в приложении без входной атаки -> Contact O
- Входная атака + выходная ошибка -> Contact Security

Коррелирующие события

- 0: Emergency – генерируется при корреляции (входная атак выходная утечка информации)
- 1: Alert – генерируется при корреляции (входная атака + выходная ошибка прикладного уровня)

Не коррелирующие события

- 2: Critical – самый высокий уровень важности, возможно, бе корреляции. Обычно это генерируется правилами веб-атаки level files)
- 3: Error – обычно генерируется правилами выходной утечки информации (50 level files)
- 4: Warning – генерируется правилами при обнаружении вредоносного клиента (35 level files)
- 5: Notice – генерируется политикой протокола
- 6: Info – генерируется при использовании клиентом поисков

Сообщения о коррелирующих событиях

```
essage: Pattern match "\;\W*?\bdrop\b" at TX:pm_sqli_data_REQUEST_URI.
file "/opt/wasc-
oneypot/etc/rules/base_rules/modsecurity_crs_41_sql_injection_attacks.c
[line "262"] [id "959001"] [msg "SQL Injection Attack"] [data "; drop"
severity "CRITICAL"] [tag "WEB_ATTACK/SQL_INJECTION"]

essage: Operator GE matched 0 at TX:anomaly_score. [file "/opt/wasc-
oneypot/etc/rules/base_rules/modsecurity_crs_49_enforcement.conf"] [lin
0"] [msg "Anomaly Score Exceeded (score 55): SQL Injection Attack
etected"]
essage: Pattern match "\bsupplied argument is not a valid MySQL\b" at
ESPONSE_BODY. [file "/opt/wasc-
oneypot/etc/rules/base_rules/modsecurity_crs_50_outbound.conf"] [line
59"] [id "971156"] [msg "SQL Information Leakage"] [severity "ERROR"]
LEAKAGE/ERRORS"]
essage: Warning. Operator GE matched 1 at TX. [file "/opt/wasc-
oneypot/etc/rules/base_rules/modsecurity_crs_60_correlation.conf"] [lin
4"] [msg "Correlated Successful Attack Identified: Inbound Attack (SQL
njection Attack Detected) + Outbound Data Leakage (SQL Information Leak
(Transactional Anomaly Score: 85)"] [severity "EMERGENCY"]
```

язвимости протокола, такие как расщепленный
твет, Request Smuggling, преждевременное
авершение URL

- Длина содержимого только для не GET/HEAD методов
- Не ASCII символы или кодировки в заголовках
- Корректное использование заголовков (например, длина содержимого является числом)
- Доступ через прокси
- `modsecurity_crs_20_protocol_violations.conf`

апросы атакующего распознаются автоматически

- Пропуск заголовков таких, как Host, Accept, User-Agent
- Хост является IP-адресом (стандартный метод распространения червей)
- `modsecurity_crs_21_protocol_anomalies.conf`

Обычно политика зависит от приложения

- Некоторые ограничения могут применяться в целом
- Для определенных окружения может быть создан белый список

Ограничения на размер

- Размер запроса, размер загрузки
- # параметров, длина параметров
- `modsecurity_crs_23_request_limits.conf`

Элементы, которые могут быть как разрешены, та ограничены

- Методы – позволять или ограничивать WebDAV, блокировать «плохие» методы, такие как CONNECT, TRA или DEBUG
- Расширения файлов – backup файлы, файлы БД, ini-файл
- Content-Types (и некоторые другие заголовки)

клиенты

- помогает против атак на определенные цели, но
могает против общей вредоносной интернет-
тивности
- Загрузка большого количества чуши & шума
- Эффективно против спама
- Уменьшает количество событий

ределение вредоносных роботов

- Уникальные атрибуты запроса: User-Agent header, URL, He
- Черный список IP-адресов
- Определение, основанное на скорости (rate)
- Определение сканеров безопасности
- Блокировка может сбить с толку ПО тестирования безопасности (WAFW00f)

уровня

Определение атак прикладного уровня, таких, которые описаны в OWASP top 10

- SQL injection и blind SQL injection
- Cross site scripting (XSS)
- OS command injection и удаленное выполнение кманд
- Удаленное включение файла

`modsecurity_crs_40_generic_attacks.conf`

`modsecurity_crs_41_sql_injection_attacks.`

`modsecurity_crs_41_xss_attacks.conf`

Сигнатуры известных уязвимостей

SpiderLabs имеет авторизацию от ET для преобразования их правил дя Snort и включения их CRS

- http://www.emergingthreats.net/

Конвертирование следующих файлов правил

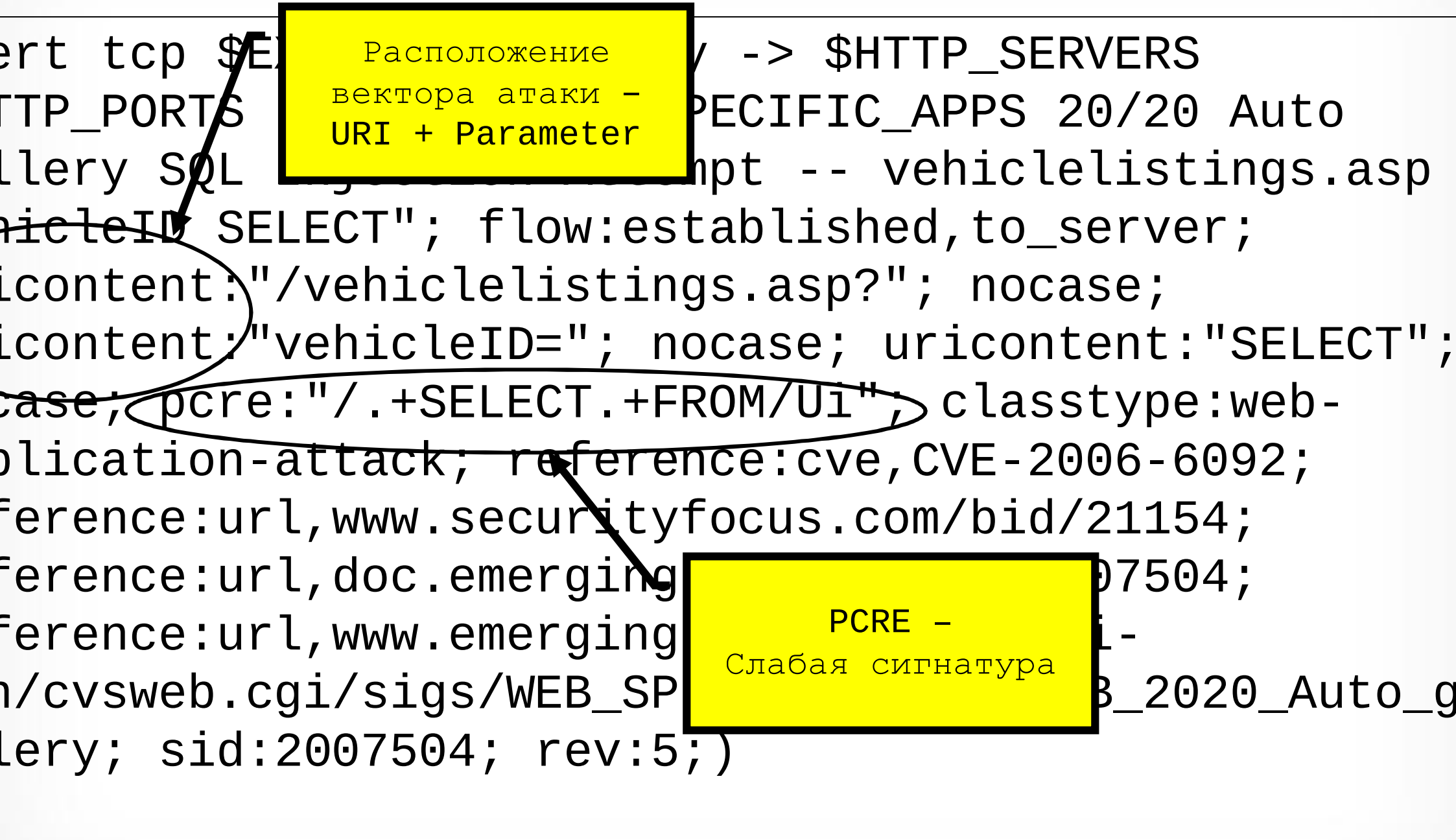
- emerging-web_server.rules
- emerging-web_specific_apps.rules

Идентификация атак на известные уязвимости

- Поднятие уровня угрозы
- Если выполнено корректно, то уменьшение ложных срабатываний

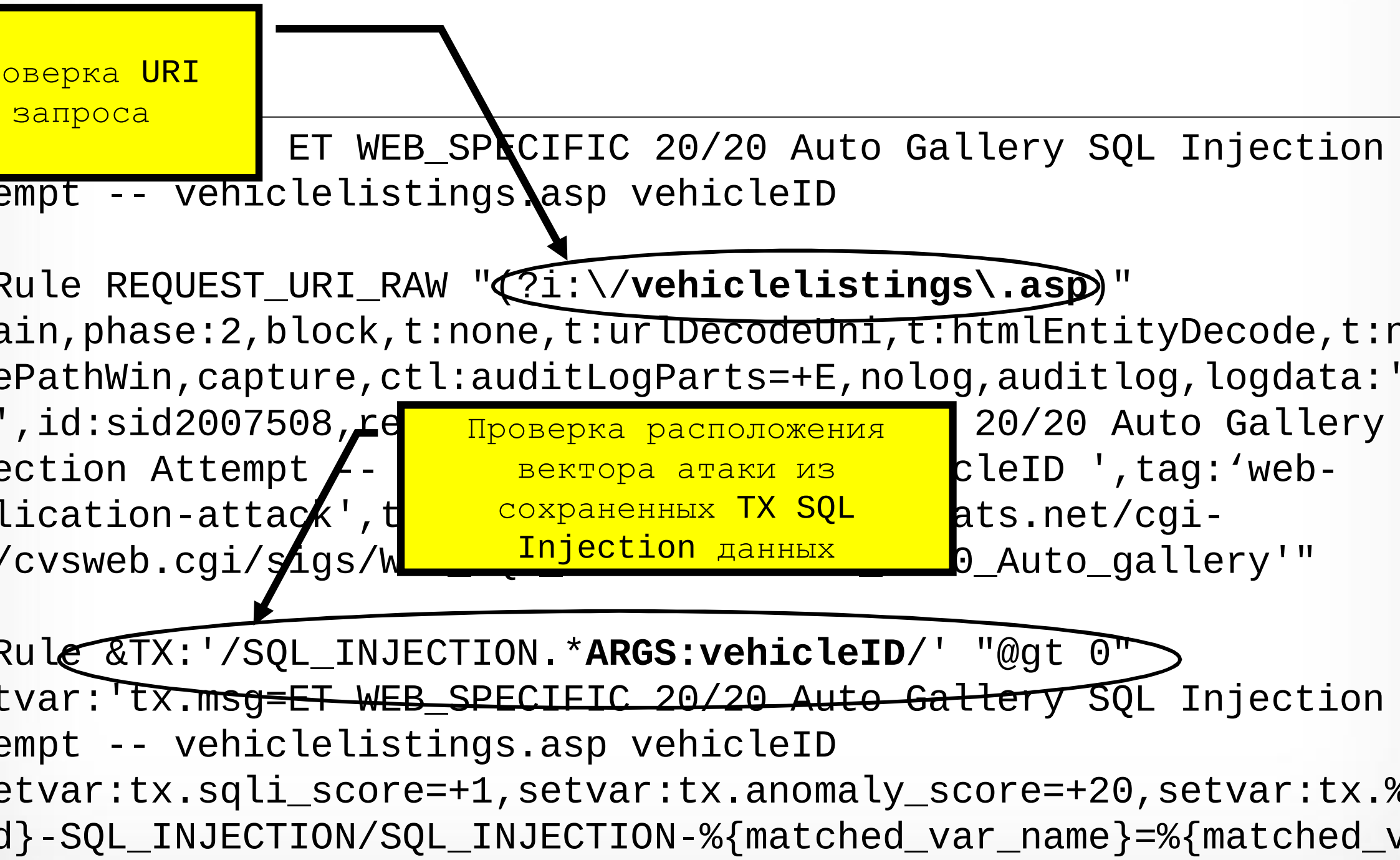
CRS комбинирует ***что*** определять в содержимом ата

Пример правила обнаружения угроз




```
ert tcp $E                    y -> $HTTP_SERVERS
TTP_PORTS                     PECIFIC_APPS 20/20 Auto
llery SQL                     mpt -- vehiclelistings.asp
hicleID SELECT"; flow:established,to_server;
icontent:"/vehiclelistings.asp?"; nocase;
icontent:"vehicleID="; nocase; uricontent:"SELECT";
case; pcre:"/.+SELECT.+FROM/Ui"; classtype:web-
plication-attack; reference:cve,CVE-2006-6092;
ference:url,www.securityfocus.com/bid/21154;
ference:url,doc.emerging                      07504;
ference:url,www.emerging                      i-
n/cvsweb.cgi/sigs/WEB_SP                      3_2020_Auto_g
lery; sid:2007504; rev:5;)
```

угроз


оверка URI
запроса
ET WEB_SPECIFIC 20/20 Auto Gallery SQL Injection
empt -- vehiclelistings.asp vehicleID
Rule REQUEST_URI_RAW "(?i:\/vehiclelistings\.asp)"
ain,phase:2,block,t:none,t:urlDecodeUni,t:htmlEntityDecode,t:r
ePathWin,capture,ctl:auditLogParts=+E,nolog,auditlog,logdata:'
',id:sid2007508,re
20/20 Auto Gallery
ection Attempt --
cleID ',tag:‘web-
lication-attack',t
ats.net/cgi-
/cvsweb.cgi/sigs/W
0_Auto_gallery'"
Проверка расположения вектора атаки из сохраненных TX SQL Injection данных
Rule &TX:'/SQL_INJECTION.*ARGS:vehicleID/' "@gt 0"
tvar:'tx.msg=ET WEB_SPECIFIC 20/20 Auto Gallery SQL Injection
empt -- vehiclelistings.asp vehicleID
etvar:tx.sqli_score=+1,setvar:tx.anomaly_score=+20,setvar:tx.%
d}-SQL_INJECTION/SQL_INJECTION-%{matched_var_name}=%{matched_v

Основная проблема для окружений, выполняющи
хостинг

- Разрешены загрузки
- Некоторые сайты могут быть юезопасны, в то время как другие нет

Определение загрузок

- Проверка загрузок файлов, содержащих вирусы (например, WORD docs)
  - `util/modsec-clamscan.pl`
- Проверка загрузок http-страниц с backdoor

Определение доступа

- Известные сигнатуры (x_key заголовок)
- Generic file management output (gid, uid, drwx, c:\)

Каналы обнаружения: утечка информа

Мониторинг исходящих прикладных данных

- Коды статуса ответа HTTP Error
- Утечка информации через SQL
- Дампы стека
- Утечка исходного кода

Последняя линяя обороны, если всё остальное вышло из строя

Обеспечить обратную связь с разработчиками приложения

Важно для приобретения опыта клиентами

Осложняет жизнь хакерам (если используется болкировка)

`modsecurity_crs_50_outbound.conf`

Усовершенствования CRS:
v2.0.9

Самые последние улучшения

- Lua портировал of PHPIDS Converter.php код
  - Улучшенные функции нормализации
  - Более точное использование PHPIDS-фильтров
  - Раздельное обнаружение содержимого атаки
- Экспериментальные правила
  - Обнаружение содержимого общих атак
- Демонстрационная страница CRS
- Тегирование заголовка запроса

Lua **портировал** PHPIDS

http://phpids.net/

~70 правил регулярных выражений для определения содержимого общих атак

- XSS
- SQL Injection
- RFI

Фильтры тяжело тестировать и часто изменять

Trustwave SpiderLabs работает с PHPIDS для портирования кода в Lua для использования с ModSecurity's API

- https://svn.php-ids.org/svn/trunk/lib/IDS/Converter.php
- https://svn.php-ids.org/svn/trunk/lib/IDS/default_filter.xml

Пример функций нормализации

- --[[Make sure the value to normalize and monitor doesn't contain Regex DoS
- --[[Check for comments and erases them if available]]
- --[[Strip newlines]]
- --[[Checks for common charcode pattern and decodes them]]
- --[[Eliminate JS regex modifiers]]
- --[[Converts from hex/dec entities]]
- --[[Normalize Quotes]]
- --[[Converts SQLHEX to plain text]]
- --[[Converts basic SQL keywords and obfuscations]]
- --[[Detects nullbytes and controls chars via ord()]]
- --[[This method matches and translates base64 strings and fragments]]
- --[[Strip XML patterns]]
- --[[This method converts JS unicode code points to regular characters]]
- --[[Converts relevant UTF-7 tags to UTF-8]]
- --[[Converts basic concatenations]]
- --[[This method collects and decodes proprietary encoding types]]

Пример фильтра PHPIDS

```
lter>
  <id>1</id>
  <rule><![CDATA[(?:"[^"]*[^-
>)|(?:[^\w\s]\s*\/>)|(?:>")]]></rule>
  <description>finds html breaking injecti
cluding whitespace attacks</description>
  <tags>
      <tag>xss</tag>
      <tag>csrf</tag>
  </tags>
  <impact>4</impact>
filter>
```

Примеры фильтров конвертации

```
Rule TX:'/^(QUERY_|REQUEST_|ARGS:).*_normalized/'
:\<\w*:?\s(?:[^\>]*)t(?!rong))|(?:\<scri)|(<\w+:\w+)"
ase:2,capture,t:none,pass,skip:1,nolog,auditlog,msg:'Detects
uscated script tags and XML wrapped
L',id:'9000033',tag:'WEB_ATTACK/XSS',logdata:'%{TX.0}',severit
,setvar:'tx.msg=%{rule.id}-
ule.msg}',setvar:tx.anomaly_score=+4,setvar:'tx.%{tx.msg}-
_ATTACK/XSS-%{matched_var_name}=%{tx.0}'"

Rule TX:PARANOID_MODE "@eq 1"
ain,phase:2,t:none,logdata:'%{TX.0}',severity:'2',pass,nolog,a
og,msg:'Detects obfuscated script tags and XML wrapped
L',id:'9000033',tag:'WEB_ATTACK/XSS'"
    SecRule ARGS|REQUEST_BODY|REQUEST_URI_RAW
:\<\w*:?\s(?:[^\>]*)t(?!rong))|(?:\<scri)|(<\w+:\w+)"
pture,multiMatch,t:none,t:urlDecodeUni,t:cssDecode,t:jsDecode,
lEntityDecode,t:replaceComments,t:compressWhiteSpace,t:lowerca
tvar:'tx.msg=%{rule.id}-
ule.msg}',setvar:tx.anomaly_score=+4,setvar:'tx.%{tx.msg}-
_ATTACK/XSS-%{matched_var_name}=%{tx.0}'"
```

IDS Centrifuge

Negative security approach to combating XSS and SQL Injection is doomed to fail...

- Unlimited ways to write functionally equivalent code
- Obfuscation methods, however often have certain characteristics

PHPIDS has an interesting approach to identify attack payloads through heuristics called *Centrifuge*

- Analysis of the use of special characters

Ratio between the count of the word characters, spaces punctuation and the non word characters

- If <3.49 = malicious

Normalization and stripping of any word character and spaces including line breaks, tabs and carriage returns

Демо портирования PHPIDS
Lua

а новых экспериментальных /beta общих правила
наружения

Optional_rules/modsecurity_crs_40_experimental.conf

пользование ограничений на символы для обнаружен
омалий

- Анализ значений и типов мета-символов, присутствующих с содержимом
- Тестирование, как будет показано далее, что обнаружение явля корректным, за исключением текстовых полей, допускающих вв свободной форме.
- Возможность приспособить определение аномалий для своего са

вторное использование символов, не имеющих
ображение

- В настоящий момент создаются оповещения (alerts), если найде

Демонстрационная страниц

CRS

Демонстрационная страница CRS

Запрос первым делом проходит через первую страницу CRS и затем запрос пропускается через прокси и передается странице PHPIDS

- http://demo.php-ids.org/

Затем инспектируются входные и выходные значени после чего предоставляются результаты

- CRS определил атаку
- CRS не нашел что-либо вредоносное, а PHPIDS нашел
- Ни CRS, ни PHPIDS не нашли ничего вредоносного

Предоставляется ссылка на отчет ложных срабатываний для JIRA ticketing системы

- https://www.modsecurity.org/tracker/browse/CORERULES

Таким способом определено >6700 атак

Демонстрационная страница CRS

odSecurity Core Rule Set (CRS) <-> PHPIDS Smoketest

st Submitted:

:lick=eval/**/(/ale/.source%2b/rt/.source%2b/(7)/.source);

```
nclick=eval/**/(/ale/.source%2b/rt/.source%2b/
7)/.source);
```

☐ Harmless HTML is allowed

☐ Input is JSON encoded

Send method=POST enctype=application/x-www-form-urlencoded

esults (txn: tltaZX8AAQEAAGwhEhcAAAAS)

S Anomaly Score Exceeded (score 77): Cross-site Scripting (XSS) Attack

Demonstration page Demo

Тегирование заголовка запроса

Демонстрационная страница CRS

В случае распределенной архитектуры можно share данные WAF с защищаемых хостов

Аналогично SMTP SPAM-данные в mime-headers

- `optional_rules/modsecurity_crs_49_header_taggin`
  `onf`

Отображение в точки обнаружения AppSensor – RP2 (подозрительное поведение внешнего пользователя)

Пример тегирования заголовка

```
T /path/to/foo.php?test=1%27%20or%20%272%27=%272%27;-- HTTP/
st: www.example.com
er-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.1.5
cko/20091109 Ubuntu/9.10 (karmic) Firefox/3.5.5
cept:
xt/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0
cept-Language: en-us,en;q=0.5
cept-Encoding: gzip,deflate
cept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
WAF-Events: TX: / 999935-Detects common comment types-
B_ATTACK/INJECTION-ARGS:test, TX:999923-Detects JavaScript
cation/document property access and window access obfuscatio
B_ATTACK/INJECTION-REQUEST_URI_RAW, TX:950001-
B_ATTACK/SQL_INJECTION-ARGS:test
WAF-Score: Total=48; sqli=2; xss=
nnection: Keep-Alive
```

Направления будущего развития

Направления будущего развития

Превентивный XSS with Content Injection

- Javascript Sandboxing with Active Content Signatures
- http://blog.modsecurity.org/2010/09/advanced-topic-of-th week-xss-defense-via-content-injection.html
- http://www.modsecurity.org/demo/demo-deny-noescape.html

Реализация OWASP AppSensor точек обнаружени

- We currently have some mappings for existing events
- Will be using Lua to implement other aggregate/behavioral/trend detection Points

Call for Community Help

We have made great strides with CRS v2.0 but there is still much work to be done

Test out the CRS demo page and report any issues found either to the mail-list or to JIRA

Need Rule Documentation help

Please sign up on our project mail-list if you want to help

- https://lists.owasp.org/mailman/listinfo/owasp-modsecurity-core-rule-set

Questions?

- Email – Ryan.Barnett@owasp.org
- Twitter - @ryancbarnett / @modsecurity