Trustwave®
SpiderLabs®

ModSecurity Training

AppSec Latam 2011

Sobre mim..

- **Membro Trustwave SpiderLabs Research**
- **Um dos criadores do Suricata IDS/IPS**
- **Mantenedor Apache ModSecurity**
- **Membro IEEE (pesquisas publicadas – China, Inglaterra, EUA, Brasil, Portugal)**

Agenda

- **O que é ModSecurity ?**
  - Pré-requisitos
  - Arquiteturas
  - Exercício 1
- **Configurando ModSecurity**
  - Principais diretivas de configuração
  - Core Rule Set
  - Exercício 2
  - Exercício 3
- **Customizando regras**
  - Sintaxe
  - Fases
  - Variáveis
  - Operadores
  - Ações
  - Funções de transformação (tfn)
  - Usabilidade das tfns, ações e variáveis
  - Exercício 4
  - Exercício 5
  - Exercício 6
  - Exercício 7
  - Exercício 8
- **Logging**
  - Diretivas de logging
  - Entendendo as log parts
  - Exercício 9
- **Lua scripting**
  - Diretivas Lua
  - Lua Hooks
  - Exemplo
- **Performance**
- **Recursos e documentação**

O que é ModSecurity ?

- **O ModSecurity é um Web Application Firewall, open source e concebido como um módulo do Apache HTTPD (2.x);**
- **Possui uma extensiva lista de opções de configuração e sintaxe de regra bastante poderosa;**
- **Utilizado por mais de 10.000 usuários ao redor do mundo;**
- **Suporte / Regras (Oferecidos pela Trustwave)**

O que é ModSecurity ?

- **GeoLocation rules;**
- **IP reputation rules;**
- **Advanced rules with Lua integration;**
- **Slow HTTP DoS mitigation engine;**
- **Robust HTTP protocol parsers;**
- **XML Parsing;**
- **On-the-Fly data modification**
- **Unicode Mappings**
- **SpiderLabs Research faz atualização constante das regras.**

O que é ModSecurity ? (próximas releases)

- **D/Denial of Service learning engine**
- **Learning positive security model**
- **Link / Cookie crypto capabilities**

Pré-requisitos

- **Plataformas**
  - Linux
  - Solaris
  - FreeBSD, OpenBSD, NetBSD
  - Windows (ApacheLounge)
- **Necessário Apache v2 + mod_unique_id**
- **Bibliotecas necessárias:**
  - LibCurl
  - LibXML2
  - LibPcre
  - LibLua
  - Apr/Apu

Arquiteturas

- **Embedded: Instalação realizada no próprio servidor web da aplicação.**
  - Prós: Não necessita modificar sua arquitetura de rede e não adiciona um ponto de falha na rede.
  - Contras: O ModSecurity irá compartilhar recursos do web server. Não é possível proteger múltiplos web server com uma única instalação. Díficil gerenciar mudanças do software e biblioteca.

- **Proxy Reverso: Instalação realizada em um servidor com apache v2 localizado logicamente a frente dos servidores web. É necessário fazer instalação do mod_proxy.**
  - Prós: Não irá compartilhar recursos (cpu, memória, i/o, etc) e poderá proteger múltiplos web servers. Atualizações do software e bibliotecas se tornam processos mais fáceis de se executar.
  - Contras: Necessidade de mudar sua arquitetura de rede e poderá ser um ponto de falha.

Exercício 1

- **Fazer o download da última versão estável no site www.modsecurity.org e salvar no diretório /home/appsec/**
- **Descompactar o pacote**
  - cd /home/appsec/ && tar zxvf modsecurity-apache_2.6.x.tar.gz
- **Compilação**
  - cd /home/appsec/modsecurity-apache_2.6.x/ && ./configure && make && make install
- **Copiar o módulo para o diretório /usr/lib/apache2/modules**
  - cp /home/appsec/modsecurity-apache_2.6.x/apache2/.libs/ mod_security2.so /usr/lib/apache2/modules/
- **Configurar o apache para carregar os módulos unique_id e mod_security**
  - vi /etc/apache2/httpd.conf
  - LoadModule unique_id_module /usr/lib/apache2/modules/mod_unique_id.so
  - LoadModule security2_module /usr/lib/apache2/modules/mod_security2.so

Configurando o ModSecurity

- **O ModSecurity possui uma ampla variedade de diretivas de configuração, tornando-o ajustável a vários tipos ambientes.**

- **As principais diretivas para o funcionamento adequado do ModSecurity serão discutidas. Estas estão presentes no arquivo modsecurity.conf-recommended para instalação default.**

Principais diretivas de configuração

- **SecRuleEngine [On|Off|DetectionOnly]**
  - Define o modo de operação do ModSecurity.
- **SecRequestBodyAccess [On|Off]**
  - Se habilitado o ModSecurity passa a processar os request bodies
- **SecRequestBodyLimit [n bytes]**
  - Define o tamanho máximo que será permitido para um request body.
- **SecRequestBodyNoFilesLimit [n bytes]**
  - Define o tamanho máximo que será permitido para um request body, sem levar em consideração qualquer tipo de arquivo sendo enviado.
- **SecRequestBodyInMemoryLimit [n bytes]**
  - Define o tamanho máximo que será permitido para um request body ser armazenado em memória, o excedente será armazenado em arquivos temporários.
- **SecRequestBodyLimitAction [Reject|ProcessPartial]**
  - Define a ação a ser tomada quando o tamanho do request body for excedido.
- **SecPcreMatchLimit [inteiro]**
  - Define um limite para a chamada de match() da API Pcre.
- **SecPcreMatchLimitRecursion [inteiro]**
  - Define um limite para a chamada de match() recursivamente.

Principais diretivas de configuração

- **SecResponseBodyAccess [On|Off]**
  - Se habilitado o ModSecurity passa a processar os response bodies
- **SecResponseBodyMimeType [mimetype minetype ...]**
  - Define quais os tipos de response body serão processados pelo modsecurity.
- **SecResponseBodyLimit [n bytes]**
  - Define o tamanho máximo que será permitido para um response body.
- **SecResponseBodyLimitAction [Reject|ProcessPartial]**
  - Define a ação a ser tomada quando o tamanho do response body for excedido.
- **SecTmpDir [diretório]**
  - Define em qual diretório serão armazenados os arquivos temporários.
- **SecDataDir [diretório]**
  - Define em qual diretório ficarão armazenados os arquivos de dados do modsecurity.
- **SecUploadDir [diretório]**
  - Define em qual diretório serão salvos os uploads.
- **SecUploadKeepFiles [On|Off]**
  - Se habilitado o modsecurity não apagará os arquivos após a transação http ser finalizada.
- **SecUploadFileMode [octal]**
  - Define as permissões dos arquivos de upload em disco.

Principais diretivas de configuração

- **SecDebugLog [diretório/arquivo]**
  - Define em qual diretório e arquivo ficarão as informações de debug
- **SecDebugLogLevel [inteiro]**
  - Define o nível de detalhe das informações de debug.
- **SecAuditEngine [On|Off|RelevanteOnly]**
  - Define o modo de operação da engine de audit do modsecurity.
- **SecAuditLogRelevantStatus [regex]**
  - Define quais response status são interessantes para arquivos de audit.
- **SecAuditLogParts [ABCDEFGHIJKZ]**
  - Define quais informações serão registradas nos arquivos de log.
- **SecAuditLogType [serial|concurrent]**
  - Define o modo de gravação dos arquivos de audit log.
- **SecAuditLog [diretório/arquivo]**
  - Define em qual diretório e arquivo ficarão as informações de audit.
- **SecAuditLogStorageDir [diretório]**
  - Define onde serão armazenados os arquivos de audit log em modo concurrent.

Principais diretivas de configuração

- **SecArgumentSeparator [char]**
  - Define qual será o separador para argumentos (usualmente &).
- **SecCookieFormat [inteiro]**
  - Define qual será a versão de formato do cookie (usualmente 0).
- **SecContentInjection [On|Off]**
  - Se habilitado o ModSecurity poderá utilizar operadores e ações para injetar código na transação. Ex: rsub, append, prepend.
- **SecStreamOutBodyInspection [On|Off]**
  - Se habilitado o ModSecurity passa a copiar o conteúdo do response body em uma variável realocável para utilização com o operador rsub.
- **SecStreamInBodyInspection [On|Off]**
  - Se habilitado o ModSecurity passa a copiar o conteúdo do request body em uma variável realocável para utilização com o operador rsub.
- **SecWriteStateLimit [inteiro]**
  - Proteção contra Slow Body DoS
- **SecReadStateLimit [inteiro]**
  - Proteção contra Slow Header DoS
- **SecGeoLookupDb [diretório/arquivo]**
  - Define onde está o arquivo que contém os dados de geo mapeamento para utilização do operador geolookup
- **SecGsbLookupDb [diretório/arquivo]**
  - Define onde está o arquivo que contém os dados do Google Safe Browsing para utilização do operador gsblookup

Core Rule Set (CRS)

- **O Core Rule Set é uma coleção de regras mantidas por Ryan Barnett da TrustWave. Possui regras para os principais ataques e problemas de segurança em aplicações web:**
  - SQL Injection
  - Cross Site-Scripting
  - Protocol violations
  - Scanners
  - Bad robots
  - Trojans e Malwares
  - Brute Force
  - D/Dos

- **Download:** https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project#tab=Download

- **Os arquivos .data são carregados por regras presentes nos arquivoc de extensão .conf.**

Exercício 2

- **Fazer o download da última versão do crs no site e salva-lo em /home/appsec/:**
  https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project#tab=Download

- **Descompactar o pacote**
  - cd /home/appsec/ && tar zxvf modsecurity-crs_2.2.2.tar.gz

- **Criar o diretório modsecurity em /etc/apache/ e copiar as regras**
  - mkdir –p /etc/apache2/modsecurity && cp –R /home/appsec/modsecurity-crs_2.2.1/* /etc/apache/modsecurity

- **Criar o arquivo de configuração modsecurity.conf em /etc/apache2/modsecurity**
  - cp /etc/apache2/modsecurity/modsecurity_crs_10_config.conf.example /etc/apache2/modsecurity/ modsecurity_crs_10_config.conf
  - Ajustar arquivo de configuração modsecurity.conf-recommended

- **Configurar o modsecurity para carregar os arquivos**
  - vi /etc/apache2/httpd.conf

```
<IfModule mod_security2.c>
Include modsecurity/*.conf
Include modsecurity/base_rules/*.conf
</IfModule>
```

- **Inicializar o apache e verificar se carregou corretamente o modsecurity**
  - apache2ctl start
  - vi /var/log/apache2/error.log

Exercício 3

- **Realizar alguns ataques no website local.**
  - Nikto –host http://IP

- **Verificar logs (debug, audit, error).**

Customizando regras

- **O ModSecurity possui uma linguagem de regra bastante robusta e flexível.**

- **Há possibilidade de criação de código Lua e integração com ModSecurity em casos onde seja necessário um tratamento diferenciado das transações HTTP.**

Sintaxe

- **SecRule VARIÁVEL[IS] "@OPERADOR" "[AÇÕES, TFNS]"**
  - Ex: SecRule ARGS|REQUEST_BODY "@rx union\+select" "phase: 2, drop, t:hexDecode"
- **SecMarker STRING**
  - Ex:

    SecRule ARGS|REQUEST_BODY "@rx union\+select" "phase:2, drop, t:hexDecode, skipafter:SQL_ATTACK"

    SecRule ARGS "@rx \<script\>" "phase:2, drop"

    SecMarker SQL_ATTACK

    SecRule ARGS "@rx table_name" "phase:2, drop"

Fases

- **Fase 1 -> REQUEST HEADER**
- **Fase 2 -> REQUEST HEADER + REQUEST BODY**
- **Fase 3 -> REQUEST HEADER + REQUEST BODY + RESPONSE HEADER**
- **Fase 4: -> REQUEST HEADER + REQUEST BODY + RESPONSE HEADER + RESPONSE BODY**
- **Fase 5: -> ACCESS LOG INFORMATIONS**

Variáveis

- **REQUEST_HEADER**
  - Contém dados http request header.
- **REQUEST_BODY**
  - Contém dados http request body.
- **RESPONSE_HEADER**
  - Contém dados http response header.
- **RESPONSE_BODY**
  - Contém dados http response body.
- **ARGS**
  - Coleção de argumentos. É possível acessar apenas um único argumento ex: ARGS:foo.
- **ARGS_NAMES**
  - Similar a ARGS, mas contém apens os nomes do argumentos.
- **ARGS_GET**
  - Similar a ARGS, mas contém apenas valores dos argumentos enviandos via GET.
- **ARGS_GET_NAMES**
  - Similar a ARGS, mas contém apenas os nomes dos argumentos enviandos via GET.
- **ARGS_POST**
  - Similar a ARGS, mas contém apenas valores dos argumentos enviandos via POST.
- **ARGS_POST_NAMES**
  - Similar a ARGS, mas contém apenas os nomes dos argumentos enviandos via POST.

Variáveis

- **QUERY_STRING**
  - Contém a query string.
- **STREAM_INPUT_BODY**
  - Similar a REQUEST_BODY. Utilizado para operadores que transformam dados da transação ex: @rsub.
- **STREAM_OUTPUT_BODY**
  - Similar a RESPONSE_BODY. Utilizado para operadores que transformam dados da transação ex: @rsub.
- **REMOTE_HOST**
  - Contém o hostname remoto.
- **REMOTE_ADDR**
  - Contém o endereço ip do host remoto.
- **TX**
  - Coleção de variáveis ex: tx.0, tx.1...tx.9 geralmente utilziadas para armazenar dados durante uma determinada transação pelos operadores.

Operadores

- **beginsWith <string>**
  - Detecta se os dados em análise começam com "string"
- **contains <string>**
  - Detecta se "string" está contida nos dados em análise
- **eq, gt, ge, lt, le <inteiro>**
  - Operadores igual, maior-que, maior-igual, menor-que, menor-igual
- **geoLookup**
  - Realiza geo lookup
- **rbl <endereço da black list>**
  - Realiza inspeção em uma black list.
- **gsbLookup <regex para extrair url>**
  - Realiza operação de lookup na base google safe browsing
- **rx <regex>**
  - Silimar a contains, mas permite utilização de uma regex
- **rsub <s/regex/string/[id]**
  - Realiza operação de substuição de dados. Utilizado apenas com as variáveis STREAM_*

Operadores

- **pm <string, string2, string3...>**
  - Similar a contains, mas pode-se utilizar vários strings
- **pmf <arquivo>**
  - Similar a pm, entretando as strings não carregadas de um arquivo
- **streq <string>**
  - Similar a contains, mas a string e a totalidade dos dados em análise devem ser exatamente iguais
- **verifyCC <regex para CC>**
  - Verifica a presença de números de cartões de crédito
- **verifyCPF <regex para CPF>**
  - Verifica a presença de números de CPF.

Ações

- **allow**
  - Pára o processamento das regras em caso de match e permite a transação.
- **append**
  - Adiciona um texto no fim do response body.
- **auditlog**
  - Indica que a regra deverá ser registrada no audit log
- **noauditlog**
  - Indica que a regra não deverá ser registrada no audit log.
- **capture**
  - Alguns operadores quando utilizado com essa ação salvam os resultados da operação de string/regex match em variáveis tx.
- **chain**
  - Indica que a regra pertence a uma cadeia de regras. A última regra da cadeia não deve conter essa ação.
- **ctl**
  - Permite modificar algumas diretivas em tempo de transação:

    auditEngine
    auditLogParts
    debugLogLevel
    ruleEngine
    ruleRemoveById
    ruleUpdateTargetById

Ações

- **deny**
  - Pára o processamento das regras e intercepta a transação.
- **id**
  - Número de identificação da regra.
- **log**
  - Indica que a regra deverá ser registrada em audit e error log.
- **nolog**
  - Indica que a regra não deverá ser registrada em audit e error log.
- **msg**
  - Define a mensagem que será salva nos arquivos de log para a regra ou chain.
- **pass**
  - Continua a processas das regras mesmo em caso de match.
- **prepend**
  - Adiciona um texto no início do response body.
- **redirect**
  - Redireciona o usuário para uma outra url qualquer.
- **skip**
  - Modifica o fluxo de execução do arquivo de regras para depois de N regras.
- **skipafter**
  - Modifica o fluxo de execução do arquivo de regras para depois de uma SecMarker.
- **sanitizeMatchedBytes, sanitizeArgs, sanitizeRequestHeaders, sanitizeMatched**
  - Permite sanitizar os dados que estão sendo registrados em log.

Funções de transformação (tfns)

- **base64Decode, base64Encode, base64DecodeEx**
  - Funções para encode e decode de dados em base64
- **hexDecode, hexEncode**
  - Funções para encode e decode de dados em hexadecimal
- **jsDecode**
  - Função para decodificação de dados javascript em formato \uHHHH
- **Lowercase**
  - Função utilizada para normalizar todos os dados para minúsculo.
- **Md5**
  - Realiza operação de hash md5 para os dados da variável em questão
- **Sha1**
  - Realiza operação de hash sha1 para os dados da variável em questão
- **urlDecodeUni**
  - Função para decodificar dados em unicode.
- **removeNulls**
  - Função para remoção de null bytes.
- **replaceComments**
  - Substitui caracteres de comentário como /* */ por espaço (0x20).

Usabilidade das tfns, ações e variáveis

- **Deve-se indicar os argumentos para aquelas tfns, ações e variáveis que os aceitam utilizando o carácter ":"**
  - Id:1122
  - t:jsDecode
  - msg:'meu texto aqui'
  - ARGS:teste

- **Pode utilizar mais de uma tfn, variável e/ou mais de uma ação na mesma SecRule**
  - SecRule RESPONSE_BODY|RESPONSE_HEADER "@pmf arquivo.txt" "id: 1122,drop,log,t:jsDecode,t:hexDecode,t:htmlEntityDecode,msg:'ataque'"

- **Alguns operadores são capazes de expadir variáveis**
  - SecRule RESPONSE_BODY "@rx %{tx.1}" "id: 1123,drop,log,t:jsDecode,t:hexDecode,t:htmlEntityDecode,msg:'ataque'"

- **Os operadores aceitam os caracteres de modificação ! e &**
  - SecRule RESPONSE_BODY "!@ipmatch 192.168.0.1" "id:111,drop"
  - SecRule &ARGS "@gt 20" ïd:22,drop"

Exemplos

SecRule REQUEST_BODY "^username=(\w{25,})" phase:2,capture,t:none,chain,deny,msg:”Usuario admin logando de maquina suspeita”

SecRule TX:0 "(?:(?:a(dmin|dministrator)))" “chain”

SecRule REMOTE_ADDR “!@ipmatch 192.168.0.1”

SecRule ARGS "@verifyCC \d{13,16}" "phase:2, capture,deny,msg:'Potential credit card number in request',sanitiseMatchedBytes”

Exercício 4

- **Escrever uma regra que detecte a string: “table_name” nos argumentos.**

  Retornando para o usuário uma message box: <script>alert (‘ I have been watching you for a long time! ‘) </script>

A:

SecContentInjection On

SecRule ARGS "@rx table_name" "phase:2,id: 111,setvar:tx.SQLI=1”

SecRule TX:SQLI "@eq 1" "phase:4,prepend:'<script>alert (\'SQL\')</script>

Exercício 5

- **Escrever uma regra que decodifique a string:**

  selectLOAD_FILE0x633A5C626F6F742E696E69

  enviado via GET em um determinado parametro e detecte o que está codificado.

A: SecRule ARGS ”@rx boot" "phase:1,deny,id: 1111,t:sqlHexDecode"

Exercício 6

- **Escrever uma regra que detect a string “union select” do argumento “sql” a partir do input “u/*ni/*on sel/*ect”**

A: SecRule ARGS:sql "@rx union" "phase:1,pass,id: 111,t:removeCommentsChar"

Exercício 7

- **Escrever uma Chain que detecte a presença da string "Works" dentro de RESPONSE_BODY e o substitua pela string "DADO_REMOVIDO" permitindo que a conexão/ transação http aconteça normalmente.**

A: SecContentInjection On

SecStreamOutBodyInspection On

SecResponseBodyAccess On

SecRule RESPONSE_BODY "@rx Works" "pass,id:111,phase: 4,chain,capture"

SecRule STREAM_OUTPUT_BODY "@rsub s/%{tx.0}/ DADO_REMOVIDO/"

Exercício 8

- **Escrever regra que faça o ModSecurity não inspecionar o diretório /admin**

A: SecRule REQUEST_FILENAME "@beginsWith /admin" "phase:1,t:none,pass,nolog,ctl:ruleEngine=Off"

Logging

- **O ModSecurity possui features para logging poderosas, tanto para debug quanto para audit.**
  - Debug: Informações de debug do processamento das regras
    - Possui níveis de detalhes 1 a 9
  - Audit: Possui informações detalhadas da transação http, contendo informações sobre o protocolo, regras que deram match...
    - Possui níveis de detalhes (ABCDEFGHIJKZ)

Diretivas de logging

- **SecAuditEngine <On|Off|RelevantOnly>**
  - Define o modo de funcionamento da engine de logging
- **SecAuditLogRelevantStatus <regex>**
  - Quando a engine for habilitada em RevelantOnly pode-se definir os códigos http relevantes para logar.
- **SecAuditLogType <serial|concurrent>**
  - Define o modo de geração dos arquivos audit log.
- **SecAuditLogParts ABCDEFHJKZ**
  - Define as partes do audit que deverão ser preenchidas.
- **SecAuditLogStorageDir <diretório>**
  - Utilizado para definir onde serão gravados os arquivos no modo concurrent.
- **SecAuditLog <diretório/arquivo>**
  - Define o nome e localização do arquivo de audit log.
- **SecDebugLog <diretório/arquivo>**
  - Define o nome e localização do arquivo de debug log
- **SecDebugLogLevel 9**
  - Define o nível de detalhamento do debug log.

Entendendo as Log Parts

- **Part A:**
  - Cabeçalho do log (timestamp, src ip, dst ip, src port, dst port, transaction id)
- **Part B:**
  - Request Header
- **Part C**
  - Request Body
- **Part D**
  - Intended Response Header.
- **Part E**
  - Intended Response Body. (Apenas quando a transação foi interceptada)
- **Part F**
  - Response Headers
- **Part G**
  - Response Body
- **Part H**
  - Audit Log Trailer (Actions, Apache Errors, Sanitised data, Phase time…)
- **Part I**
  - Reduced Multipart Request Body
- **Part J**
  - Multipart Files Information
- **Part K**
  - Matched Rules
- **Part Z**
  - Audit log Footer

Audit Log

--87c77e78-A--
[24/Aug/2011:08:27:30 --0700] TlUYYsCoAGQAAH9RDrUAAAGQ 192.168.0.110 52991 192.168.0.101 80
--87c77e78-B--
GET /index.html?teste=testselectLOAD_FILE0x633A5C626F6F742E696E69%27+ HTTP/1.1
Host: 192.168.0.101
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US; rv:1.9.2.6) Gecko/20100625 Firefox/3.6.6
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive

--87c77e78-F--
HTTP/1.1 403 Forbidden
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 255
Keep-Alive: timeout=300
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

Audit Log

--87c77e78-E--
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
You don't have permission to access /index.html
on this server.
<hr>
<address>Apache/2.2.14 (Ubuntu) mod_fcgid/2.3.4 Server at 192.168.0.101 Port 80</address>
</body></html>

--87c77e78-H--
Message: Access denied with code 403 (phase 1). Pattern match "test" at ARGS:teste. [file "/etc/apache2/modsecurity/
modsecurity_crs_15_customrules.conf"] [line "134"] [id "1114"]
Action: Intercepted (phase 1)
Stopwatch: 1314199650684721 1556 (- - -)
Stopwatch2: 1314199650684721 1556; combined=240, p1=212, p2=0, p3=0, p4=0, p5=27, sr=0, sw=1, l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.6.1 (http://www.modsecurity.org/).
Server: Apache/2.2.14 (Ubuntu)

--87c77e78-K--
SecRule "ARGS" "@rx test" "phase:1,auditlog,deny,id:1114,t:hexDecode,log"

--87c77e78-Z--

Debug Log

1- [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][4] Initialising transaction (txid TlOxGcCoAGQAAGmyDt4AAAGQ).
2- [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][5] Adding request argument (QUERY_STRING): name "test", value "selectLOAD_FILE0x633A5C626F6F742E696E69"
3 - [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][4] Transaction context created (dcfg 22df6940).
4 - [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][4] First phase starting (dcfg 22df6940).
5 - [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][4] Starting phase REQUEST_HEADERS.
6 - [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][9] This phase consists of 1 rule(s).
7 - [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][4] Recipe: Invoking rule 22e2ee08; [file "/etc/apache2/modsecurity/modsecurity_crs_15_customrules.conf"] [line "134"] [id "1112"].
8 - [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][5] Rule 22e2ee08: SecRule "ARGS" "@rx test" "phase:1,log,auditlog,pass,id:1112,t:hexDecode”
9 - [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][9] T (0) hexDecode: "selectLOAD_FILE0xc:\\boot.ini"
10 - [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][4] Transformation completed in 15 usec.
11 - [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][4] Executing operator "rx" with param "test" against ARGS:test.
12 - [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][9] Target value: "selectLOAD_FILE0xc:\\boot.ini"
13 - [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][4] Operator completed in 7 usec.
14- [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][4] Rule returned 0.
15 - [23/Aug/2011:06:54:33 --0700] [192.168.0.101/sid#22e6ffb0][rid#22ebae50][/index.html][9] No match, not chained -> mode NEXT_RULE.

Exercício 9

- **Com SecDebugLogLevel 0 modificar a regra abaixo para que o debug log seja habilitado apenas para a transação http que der match.**

  SecRule ARGS "@pmf dados.data" "id:111,pass"

A: SecRule ARGS "@pmf dados.data" "id: 111,pass,ctl:debugLogLevel=9"

Scripting Lua - Diretivas

- **SecRuleScript /path/to/script.lua [ACTIONS]**

- **SecRule ARGS:p attack "phase:2,block,exec:/usr/local/apache/conf/exec.lua"**

Scripting Lua - Hooks

- **getvar(“VARIÁVEL_NAME”, {“tfn1, tfn2...”});**
- **getvars(“COLEÇÃO”, {“tfn1, tfn2...”});**
- **setvar(“COLEÇÃO.VARIÁVEL_NAME”, “VALOR”);**
- **Log(level, “MSG”);**

Scripting Lua - Exemplo

```
require(‘m’);
function main()
local d = m.getvars("ARGS”);

    for i = 1, #vars do
        value = (vars[i].value);
        if (string.gmatch(value, “script")) then
            m.setvar(“tx.attack”, "Suspected XSS”);
            return ("Suspected XSS”);
        end
    end
return nil;
end
```

Performance

- **Optar por @pm/@pmf a @rx sempre que possível**
- **Combinar multi-patterns em uma mesma SecRule**
  - SecRule "@pm teste teste2 teste3…"
- **Regex combinadas são rápidas para porções de dados menores (ie. 300 bytes), mas não para porções maiores (ie 15kb).**
- **Reduzir falso-positivo. Redução I/O de disco.**
- **Utilize corretamente as fases. Trabalhar com a porção de dados imediatamente após estar disponível.**
- **Enforce de limites.**
  - Limite dos valores de parâmetros
  - Byte range dos parâmetros
  - Quantidade de parâmetros na query string
  - Limite de tamanho máximo da query string
  - Limite máximo para o request body
- **Variáveis PERF_family podem ser utilizadas para identificar performance issues:**
  - SecRule PERF_COMBINED "@gt 2500" "phase:5,pass,id:111"

Performance

- **Não habilitar debug log em produção**
- **Utilizar concurrent audit log. Elimina overhead dos mutexes.**
- **FreeBSD + HTTP Accept Filter**
- **Apache 2.2**

Recursos e documentação

- **ModSecurity WebSite**
  - www.modsecurity.org
- **User-List e Devel-List**
  - mod-security-users@lists.sourceforge.netModSecurity demo page, mod-security-developers@lists.sourceforge.net.
- **ModSecurity HandBook**
- **ModSecurity Reference Manual**
  - http://sourceforge.net/apps/mediawiki/mod-security/index.php?title=Reference_Manual

Contato: bpinto@trustwave.com

Trustwave®

SpiderLabs®

Obrigado!