Micro Focus WebInspect

Multiple Reports

Web Application Assessment Report

Scan Name:	Site: http://zero.webappsecurity.com/		
Policy:	Standard	**Crawl Sessions:**	16
Scan Date:	12/30/2019 3:47:31 PM	**Vulnerabilities:**	24
Scan Version:	19.2.0.184	**Scan Duration:**	3 minutes : 14 seconds
Scan Type:	Site	**Client:**	FF

Server: http://zero.webappsecurity.com:80

Page: http://zero.webappsecurity.com:80/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1 ; JSESSIONID=C0A918A3

Page: http://zero.webappsecurity.com:80/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/<script>alert('TRACK');</script>

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1 ; JSESSIONID=C0A918A3

Page: http://zero.webappsecurity.com:80/debug.txt

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1 ; JSESSIONID=09389BDA

Page: http://zero.webappsecurity.com:80/faq.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1 ; JSESSIONID=4C4522A6

Page: http://zero.webappsecurity.com:80/faq.html.bak

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/faq.html?question=2%3c%73%43%72%49%70%54%3e%61%6c%65%72%74%28%31%33%31%31%35%29%3c%2f%73%43%72%49%70%54%3e

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=4905A438

Page: http://zero.webappsecurity.com:80/feedback.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/forgot-password.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/forgotten-password-send.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=4C4522A6

Page: http://zero.webappsecurity.com:80/forgotten-password-send.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=4C4522A6

Page: http://zero.webappsecurity.com:80/forgotten-password-send.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=58416A8B

Page: http://zero.webappsecurity.com:80/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/index.html.old

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/index.old

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=AF20C34F

Page: http://zero.webappsecurity.com:80/login.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/login.html?login_error=true

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=4C4522A6

Page: http://zero.webappsecurity.com:80/online-banking.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/README.txt

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=C1C8DF89

Page: http://zero.webappsecurity.com:80/readme.txt

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=011FC3D6

Page: http://zero.webappsecurity.com:80/search.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/search.html?searchTerm=${7166%2b2327}

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=011FC3D6

Page: http://zero.webappsecurity.com:80/search.html?searchTerm=${cookie%5B%22JSESSIONID%22%5D%2Evalue}

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=011FC3D6

Page: http://zero.webappsecurity.com:80/search.html?searchTerm=12345

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/search.html?searchTerm=12345%3c%73%43%72%3c%53%63%52%69%50%74%3e%49%70%54%3e%61%6c%65%72%74%28%36%34%32%32%35%29%3c%2f%73%43%72%3c%53%63%52%69%50%74%3e%49%70%54%3e

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=011FC3D6

Page: http://zero.webappsecurity.com:80/sendFeedback.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=4C4522A6

Page: http://zero.webappsecurity.com:80/sendFeedback.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=92FC45D6

Page: http://zero.webappsecurity.com:80/sendFeedback.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=58416A8B

Page: http://zero.webappsecurity.com:80/sendFeedback.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=4C4522A6

Page: http://zero.webappsecurity.com:80/server-status

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=92FC45D6

Page: http://zero.webappsecurity.com:80/signin.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/signin.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=4C4522A6

Page: http://zero.webappsecurity.com:80/signin.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=4C4522A6

Page: http://zero.webappsecurity.com:80/signin.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=4C4522A6

Scan Name:	Site: http://zero.webappsecurity.com/		
Policy:	Standard	**Crawl Sessions:**	239
Scan Date:	12/30/2019 3:57:58 PM	**Vulnerabilities:**	103
Scan Version:	19.2.0.184	**Scan Duration:**	9 minutes : 58 seconds
Scan Type:	Site	**Client:**	FF

Server: http://zero.webappsecurity.com:80

Page: http://zero.webappsecurity.com:80/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=BD06CAFC

Page: http://zero.webappsecurity.com:80/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/<script>alert('TRACK');</script>

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=967D9987

Page: http://zero.webappsecurity.com:80/account/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/admin/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=CC8ABF64

Page: http://zero.webappsecurity.com:80/admin/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/admin/currencies.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/admin/currencies-add.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/admin/currencies-add.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/admin/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=CC8ABF64

Page: http://zero.webappsecurity.com:80/admin/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/admin/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/admin/users.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/admin/WS_FTP.LOG

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;

JSESSIONID=1B07F0B6

Page: http://zero.webappsecurity.com:80/backup/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/bank/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=F6FED876

Page: http://zero.webappsecurity.com:80/bank/account-activity.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/bank/account-summary.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/bank/money-map.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/bank/money-map.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/bank/money-map.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;

JSESSIONID=F6FED876

Page: http://zero.webappsecurity.com:80/bank/online-statements.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/bank/pay-bills.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/bank/transfer-funds.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/cgi-bin/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/db/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/debug.txt

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=847776D7

Page: http://zero.webappsecurity.com:80/docs/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/aio.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/api/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=1B07F0B6

Page: http://zero.webappsecurity.com:80/docs/api/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/api/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/api/index.html?org/apache/catalina/websocket/package-summary.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/api/index.html?org/apache/catalina/websocket/WebSocketServlet.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/appdev/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=967D9987

Page: http://zero.webappsecurity.com:80/docs/appdev/build.xml.txt

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

;
JSESSIONID=100C4F4E

Page: http://zero.webappsecurity.com:80/docs/appdev/deployment.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/appdev/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/appdev/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/appdev/installation.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/appdev/introduction.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/appdev/processes.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/appdev/sample/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/appdev/sample/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/appdev/sample/sample.war

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=100C4F4E

Page: http://zero.webappsecurity.com:80/docs/appdev/sample/sample.war

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=100C4F4E

Page: http://zero.webappsecurity.com:80/docs/appdev/source.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/appdev/web.xml.txt

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=100C4F4E

Page: http://zero.webappsecurity.com:80/docs/apr.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/architecture/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=22052993

Page: http://zero.webappsecurity.com:80/docs/architecture/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/architecture/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/architecture/overview.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/architecture/requestProcess.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/architecture/startup.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/architecture/startup/serverStartup.txt

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=100C4F4E

Page: http://zero.webappsecurity.com:80/docs/balancer-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/building.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/BUILDING.txt

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/cgi-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/changelog.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/class-loader-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/cluster-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/comments.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=BD06CAFC

Page: http://zero.webappsecurity.com:80/docs/config/ajp.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/automatic-deployment.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=100C4F4E

Page: http://zero.webappsecurity.com:80/docs/config/cluster.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/cluster-channel.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/cluster-deployer.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/cluster-interceptor.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/cluster-listener.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/cluster-manager.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/cluster-membership.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/cluster-receiver.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/cluster-sender.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/cluster-valve.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/context.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/engine.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/executor.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/filter.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/globalresources.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/host.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/http.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/jar-scanner.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/listeners.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/loader.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/manager.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/realm.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/resources.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/server.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/service.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/sessionidgenerator.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/systemprops.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/valve.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/config/valve.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/connectors.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/default-servlet.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/deployer-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/deployer-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/developers.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/elapi/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1 ; JSESSIONID=F6FED876

Page: http://zero.webappsecurity.com:80/docs/elapi/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/elapi/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/extras.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/funcspecs/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1 ; JSESSIONID=F6FED876

Page: http://zero.webappsecurity.com:80/docs/funcspecs/fs-admin-apps.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/funcspecs/fs-admin-objects.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/funcspecs/fs-admin-opers.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/funcspecs/fs-default.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/funcspecs/fs-jdbc-realm.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/funcspecs/fs-jndi-realm.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/funcspecs/fs-memory-realm.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/funcspecs/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/funcspecs/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/funcspecs/mbean-names.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/html-manager-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/introduction.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/jasper-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/jdbc-pool.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/jndi-datasource-examples-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/jndi-resources-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/jspapi/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=F6FED876

Page: http://zero.webappsecurity.com:80/docs/jspapi/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/jspapi/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/logging.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/manager-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/maven-jars.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/mbeans-descriptor-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/monitoring.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/proxy-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/realm-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/RELEASE-NOTES.txt

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/RUNNING.txt

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/security-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/security-manager-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/servletapi/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=2CE54099

Page: http://zero.webappsecurity.com:80/docs/servletapi/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/servletapi/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/setup.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/ssi-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/ssl-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/tribes/developers.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/tribes/faq.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/tribes/interceptors.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/tribes/introduction.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/tribes/introduction.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/tribes/membership.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/tribes/setup.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/tribes/status.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/tribes/transport.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/virtual-hosting-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/websocketapi/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=22052993

Page: http://zero.webappsecurity.com:80/docs/websocketapi/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/websocketapi/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/web-socket-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/windows-auth-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/docs/windows-service-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/error_log/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=F6FED876

Page: http://zero.webappsecurity.com:80/errors/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/errors/errors.log

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/errors/errors.log

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/faq.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/faq.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/faq.html.bak

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=F6FED876

Page: http://zero.webappsecurity.com:80/faq.html?question=1

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/faq.html?question=1%3c%73%43%72%49%70%54%3e%61%6c%65%72%74%28%38%35%37%36%35%29%3c%2f%73%43%72%49%70%54%3e

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=CC8ABF64

Page: http://zero.webappsecurity.com:80/faq.html?question=2

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/feedback.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/forgot-password.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/forgotten-password-send.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/forgotten-password-send.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=CC8ABF64

Page: http://zero.webappsecurity.com:80/forgotten-password-send.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/htbin/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/include/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/include/common.inc

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=62AED4FD

Page: http://zero.webappsecurity.com:80/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/index.html.old

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/index.old

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=BD06CAFC

Page: http://zero.webappsecurity.com:80/login.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/login.html?login_error=true

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/manager

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=100C4F4E

Page: http://zero.webappsecurity.com:80/manager/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=F6FED876

Page: http://zero.webappsecurity.com:80/online-banking.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/README.txt

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=62AED4FD

Page: http://zero.webappsecurity.com:80/readme.txt

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=967D9987

Page: http://zero.webappsecurity.com:80/resources/css/bootstrap.min.css

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/resources/css/font-awesome.css

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/resources/css/jquery-ui-1.8.16.custom.css

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/resources/css/main.css

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/resources/js/bootstrap.min.js

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/resources/js/jquery-1.6.4.min.js

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/resources/js/jquery-1.7.2.min.js

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/resources/js/jquery-1.8.2.min.js

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/resources/js/jquery-ui.min.js

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/resources/js/placeholders.min.js

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/scripts/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/search.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/search.html?searchTerm=${6228%2b1320}

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;

JSESSIONID=22052993

Page: http://zero.webappsecurity.com:80/search.html?searchTerm=${cookie%5B%22JSESSIONID%22%5D%2Evalue}

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=22052993

Page: http://zero.webappsecurity.com:80/search.html?searchTerm=12345

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/search.html?searchTerm=12345%3c%73%43%72%3c%53%63%52%69%50%74%3e%49%70%54%3e%61%6c%65%72%74%28%34%37%32%37%34%29%3c%2f%73%43%72%3c%53%63%52%69%50%74%3e%49%70%54%3e

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=1B07F0B6

Page: http://zero.webappsecurity.com:80/sendFeedback.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=2CE54099

Page: http://zero.webappsecurity.com:80/sendFeedback.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/sendFeedback.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1
;
JSESSIONID=399FB37E

Page:	http://zero.webappsecurity.com:80/server-status
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1 ; JSESSIONID=22052993

Page:	http://zero.webappsecurity.com:80/signin.html
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1 ; JSESSIONID=399FB37E

Page:	http://zero.webappsecurity.com:80/signin.html
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1 ; JSESSIONID=399FB37E

Page:	http://zero.webappsecurity.com:80/signin.html
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1 ; JSESSIONID=399FB37E

Page:	http://zero.webappsecurity.com:80/signin.html
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1 ; JSESSIONID=399FB37E

Page:	http://zero.webappsecurity.com:80/signin.html
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1 ; JSESSIONID=399FB37E

Page:	http://zero.webappsecurity.com:80/signin.html
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1 ; JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/signin.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=399FB37E

Page: http://zero.webappsecurity.com:80/stats/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/testing/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Page: http://zero.webappsecurity.com:80/user/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Scan Info

Scan Name	# of Servers	Policy	Date	Duration	Vulns
(A) Site: http://zero.webappsecurity.com/	1	Standard	12/30/19	3 mins	24
(B) Site: http://zero.webappsecurity.com/	1	Standard	12/30/19	9 mins	103

Vulnerability Comparison

Vulnerabilities only in 'Site: http://zero.webappsecurity.com/' (A)

Medium **(11294)** [description]

URL: http://zero.webappsecurity.com:80/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Medium **(11622)** [description]

URL: http://zero.webappsecurity.com:80/sendFeedback.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=58416A8B

Low **(11279)** [description]

URL: http://zero.webappsecurity.com:80/sendFeedback.html

Cookie:
CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=4C4522A6

Low **(11309)** [description]

URL: http://zero.webappsecurity.com:80/index.html

Cookie:
CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(11603)** [description]

URL: http://zero.webappsecurity.com:80/index.html

Cookie:
CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Informational **(11606)** [description]

URL: http://zero.webappsecurity.com:80/index.html

Cookie:
CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Vulnerabilities only in 'Site: http://zero.webappsecurity.com/' (B)

Critical **(742)** [description]

URL: http://zero.webappsecurity.com:80/account/

Cookie:
CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Critical **(10834)** [description]

URL: http://zero.webappsecurity.com:80/admin/users.html

Cookie:
CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Medium **(746)** [description]

URL: http://zero.webappsecurity.com:80/errors/

Cookie:
CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Medium **(764)** [description]

URL: http://zero.webappsecurity.com:80/admin/WS_FTP.LOG

Cookie:
CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=1B07F0B6

Medium **(10365)** [description]

URL: http://zero.webappsecurity.com:80/include/common.inc

Cookie:
CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=62AED4FD

Medium **(11294)** [description]

URL: http://zero.webappsecurity.com:80/

Cookie:
CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Medium **(11622)** [description]

URL: http://zero.webappsecurity.com:80/bank/money-map.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=F6FED876

Low	**(810)**	[description]

URL: http://zero.webappsecurity.com:80/docs/manager-howto.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/realm-howto.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/security-howto.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/virtual-hosting-howto.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/class-loader-howto.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/config/context.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/config/listeners.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/ssl-howto.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/setup.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/security-manager-howto.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/cluster-howto.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/building.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/monitoring.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/logging.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/appdev/processes.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/jasper-howto.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/config/resources.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/changelog.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/windows-auth-howto.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(2291)** [description]

URL: http://zero.webappsecurity.com:80/docs/ssi-howto.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(3508)** [description]

URL: http://zero.webappsecurity.com:80/docs/config/filter.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/monitoring.html
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/errors/errors.log
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/admin/WS_FTP.LOG
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=1B07F0B6

Low **(10210)** [description]

URL: http://zero.webappsecurity.com:80/manager/
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=F6FED876

URL: http://zero.webappsecurity.com:80/admin/
Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(10211)** [description]

URL: http://zero.webappsecurity.com:80/backup/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(10212)** [description]

URL: http://zero.webappsecurity.com:80/htbin/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/cgi-bin/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/scripts/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(10214)** [description]

URL: http://zero.webappsecurity.com:80/include/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/errors/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(10216)** [description]

URL: http://zero.webappsecurity.com:80/db/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(10217)** [description]

URL: http://zero.webappsecurity.com:80/testing/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(10218)** [description]

URL: http://zero.webappsecurity.com:80/docs/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(10220)** [description]

URL: http://zero.webappsecurity.com:80/bank/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=F6FED876

Low **(10229)** [description]

URL: http://zero.webappsecurity.com:80/stats/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/error_log/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1;

Low **(10233)** [description]

URL: http://zero.webappsecurity.com:80/user/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(10735)** [description]

URL: http://zero.webappsecurity.com:80/docs/html-manager-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/building.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/config/host.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/windows-service-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/admin/WS_FTP.LOG

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=1B07F0B6

URL: http://zero.webappsecurity.com:80/docs/windows-auth-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(10810)** [description]

URL: http://zero.webappsecurity.com:80/admin/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=CC8ABF64

Low **(10842)** [description]

URL: http://zero.webappsecurity.com:80/docs/realm-howto.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/config/listeners.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(10932)** [description]

URL: http://zero.webappsecurity.com:80/account/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(11279)** [description]

URL: http://zero.webappsecurity.com:80/docs/servletapi/

Cookie:

URL:	http://zero.webappsecurity.com:80/docs/api/index.html
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL:	http://zero.webappsecurity.com:80/docs/websocketapi/
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=22052993

URL:	http://zero.webappsecurity.com:80/docs/funcspecs/index.html
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL:	http://zero.webappsecurity.com:80/docs/appdev/
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=967D9987

URL:	http://zero.webappsecurity.com:80/docs/jspapi/index.html
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL:	http://zero.webappsecurity.com:80/docs/api/
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=1B07F0B6

URL:	http://zero.webappsecurity.com:80/docs/tribes/introduction.html
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL:	http://zero.webappsecurity.com:80/errors/errors.log
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL:	http://zero.webappsecurity.com:80/docs/deployer-howto.html
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL:	http://zero.webappsecurity.com:80/docs/elapi/index.html
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL:	http://zero.webappsecurity.com:80/docs/funcspecs/
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=F6FED876

URL:	http://zero.webappsecurity.com:80/docs/appdev/sample/
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL:	http://zero.webappsecurity.com:80/bank/money-map.html
Cookie:	CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=399FB37E

URL: http://zero.webappsecurity.com:80/docs/config/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=BD06CAFC

URL: http://zero.webappsecurity.com:80/docs/appdev/sample/sample.war

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=100C4F4E

URL: http://zero.webappsecurity.com:80/docs/websocketapi/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/architecture/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=22052993

URL: http://zero.webappsecurity.com:80/docs/appdev/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/config/valve.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/servletapi/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/admin/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/architecture/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/docs/elapi/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=F6FED876

URL: http://zero.webappsecurity.com:80/docs/jspapi/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=F6FED876

URL: http://zero.webappsecurity.com:80/faq.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=399FB37E

Low **(11309)** [description]

URL: http://zero.webappsecurity.com:80/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Low **(11603)** [description]

URL: http://zero.webappsecurity.com:80/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Informational **(3872)** [description]

URL: http://zero.webappsecurity.com:80/admin/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1; JSESSIONID=CC8ABF64

Informational **(11606)** [description]

URL: http://zero.webappsecurity.com:80/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Best Practice **(5597)** [description]

URL: http://zero.webappsecurity.com:80/admin/currencies.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/admin/index.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/admin/currencies-add.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/admin/users.html

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

URL: http://zero.webappsecurity.com:80/admin/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Best Practice **(11359)** [description]

URL: http://zero.webappsecurity.com:80/docs/appdev/sample/

Cookie: CustomCookie=WebInspect153248ZXB974323F2DC54432902E2A17CE79B820Y7DD1

Coverage Comparison

Pages only encountered in 'Site: http://zero.webappsecurity.com/' (A)

- http://zero.webappsecurity.com:80/signin.html

  Postdata: user_login=foo&user_password=foo&user_remember_me=on&submit=Sign%20in&user_token=fbaef29e-78ce-4fac-9a6a-c91eb0121889

- http://zero.webappsecurity.com:80/sendFeedback.html
- http://zero.webappsecurity.com:80/sendFeedback.html
- http://zero.webappsecurity.com:80/signin.html

  Postdata: user_login=foo&user_password=foo&user_remember_me=on&submit=Sign%20in&user_token=a721261b-2409-4819-8d02-1ecea4d44e96

Pages only encountered in 'Site: http://zero.webappsecurity.com/' (B)

- http://zero.webappsecurity.com:80/stats/
- http://zero.webappsecurity.com:80/docs/servletapi/
- http://zero.webappsecurity.com:80/docs/config/cluster-channel.html
- http://zero.webappsecurity.com:80/docs/html-manager-howto.html
- http://zero.webappsecurity.com:80/htbin/
- http://zero.webappsecurity.com:80/admin/currencies.html
- http://zero.webappsecurity.com:80/admin/index.html
- http://zero.webappsecurity.com:80/docs/api/index.html
- http://zero.webappsecurity.com:80/docs/funcspecs/fs-admin-opers.html
- http://zero.webappsecurity.com:80/docs/manager-howto.html
- http://zero.webappsecurity.com:80/admin/currencies-add.html
- http://zero.webappsecurity.com:80/docs/architecture/startup/serverStartup.txt
- http://zero.webappsecurity.com:80/docs/config/cluster-manager.html
- http://zero.webappsecurity.com:80/docs/websocketapi/
- http://zero.webappsecurity.com:80/docs/appdev/index.html
- http://zero.webappsecurity.com:80/docs/default-servlet.html
- http://zero.webappsecurity.com:80/docs/architecture/startup.html
- http://zero.webappsecurity.com:80/docs/api/index.html
- http://zero.webappsecurity.com:80/docs/extras.html
- http://zero.webappsecurity.com:80/signin.html

  Postdata: user_login=foo&user_password=foo&user_remember_me=on&submit=Sign%20in&user_token=61a30dd2-79b8-4239-a723-0468e78433fe
- http://zero.webappsecurity.com:80/docs/funcspecs/index.html
- http://zero.webappsecurity.com:80/docs/tribes/developers.html
- http://zero.webappsecurity.com:80/docs/config/jar-scanner.html
- http://zero.webappsecurity.com:80/docs/realm-howto.html
- http://zero.webappsecurity.com:80/docs/security-howto.html
- http://zero.webappsecurity.com:80/docs/appdev/
- http://zero.webappsecurity.com:80/docs/config/valve.html
- http://zero.webappsecurity.com:80/docs/config/cluster-deployer.html
- http://zero.webappsecurity.com:80/signin.html

  Postdata: user_login=foo&user_password=foo&submit=Sign%20in&user_token=e563bb3d-f1d4-43ee-b049-b93d86ede475
- http://zero.webappsecurity.com:80/docs/config/http.html
- http://zero.webappsecurity.com:80/docs/api/index.html?org/apache/catalina/websocket/WebSocketServlet.html
- http://zero.webappsecurity.com:80/sendFeedback.html
- http://zero.webappsecurity.com:80/docs/tribes/status.html
- http://zero.webappsecurity.com:80/docs/config/sessionidgenerator.html
- http://zero.webappsecurity.com:80/signin.html

  Postdata: user_login=foo&user_password=foo&submit=Sign%20in&user_token=

61a30dd2-79b8-4239-a723-0468e78433fe

- http://zero.webappsecurity.com:80/docs/jspapi/index.html
- http://zero.webappsecurity.com:80/docs/jspapi/index.html
- http://zero.webappsecurity.com:80/bank/transfer-funds.html
- http://zero.webappsecurity.com:80/docs/funcspecs/fs-memory-realm.html
- http://zero.webappsecurity.com:80/docs/api/
- http://zero.webappsecurity.com:80/docs/appdev/web.xml.txt
- http://zero.webappsecurity.com:80/docs/virtual-hosting-howto.html
- http://zero.webappsecurity.com:80/error_log/
- http://zero.webappsecurity.com:80/docs/class-loader-howto.html
- http://zero.webappsecurity.com:80/docs/jndi-datasource-examples-howto.html
- http://zero.webappsecurity.com:80/docs/tribes/introduction.html
- http://zero.webappsecurity.com:80/resources/css/bootstrap.min.css
- http://zero.webappsecurity.com:80/docs/config/systemprops.html
- http://zero.webappsecurity.com:80/docs/config/filter.html
- http://zero.webappsecurity.com:80/docs/deployer-howto.html
- http://zero.webappsecurity.com:80/docs/tribes/interceptors.html
- http://zero.webappsecurity.com:80/signin.html

  Postdata: user_login=foo&user_password=foo&user_remember_me=on&submit=Sign%20in&user_token=e563bb3d-f1d4-43ee-b049-b93d86ede475

- http://zero.webappsecurity.com:80/errors/errors.log
- http://zero.webappsecurity.com:80/docs/config/context.html
- http://zero.webappsecurity.com:80/docs/appdev/source.html
- http://zero.webappsecurity.com:80/cgi-bin/
- http://zero.webappsecurity.com:80/docs/appdev/installation.html
- http://zero.webappsecurity.com:80/resources/js/placeholders.min.js
- http://zero.webappsecurity.com:80/docs/appdev/deployment.html
- http://zero.webappsecurity.com:80/account/
- http://zero.webappsecurity.com:80/docs/deployer-howto.html
- http://zero.webappsecurity.com:80/manager/
- http://zero.webappsecurity.com:80/bank/account-activity.html
- http://zero.webappsecurity.com:80/docs/index.html
- http://zero.webappsecurity.com:80/docs/balancer-howto.html
- http://zero.webappsecurity.com:80/docs/web-socket-howto.html
- http://zero.webappsecurity.com:80/docs/websocketapi/index.html
- http://zero.webappsecurity.com:80/docs/appdev/build.xml.txt
- http://zero.webappsecurity.com:80/docs/funcspecs/fs-jdbc-realm.html
- http://zero.webappsecurity.com:80/docs/config/cluster-interceptor.html
- http://zero.webappsecurity.com:80/resources/css/jquery-ui-1.8.16.custom.css
- http://zero.webappsecurity.com:80/docs/architecture/index.html
- http://zero.webappsecurity.com:80/bank/

- http://zero.webappsecurity.com:80/docs/funcspecs/fs-jndi-realm.html
- http://zero.webappsecurity.com:80/docs/config/listeners.html
- http://zero.webappsecurity.com:80/user/
- http://zero.webappsecurity.com:80/docs/config/manager.html
- http://zero.webappsecurity.com:80/docs/ssl-howto.html
- http://zero.webappsecurity.com:80/docs/introduction.html
- http://zero.webappsecurity.com:80/docs/aio.html
- http://zero.webappsecurity.com:80/bank/pay-bills.html
- http://zero.webappsecurity.com:80/docs/config/globalresources.html
- http://zero.webappsecurity.com:80/admin/index.html
- http://zero.webappsecurity.com:80/manager
- http://zero.webappsecurity.com:80/docs/config/cluster-sender.html
- http://zero.webappsecurity.com:80/docs/elapi/index.html
- http://zero.webappsecurity.com:80/docs/funcspecs/
- http://zero.webappsecurity.com:80/docs/setup.html
- http://zero.webappsecurity.com:80/bank/money-map.html
- http://zero.webappsecurity.com:80/resources/js/jquery-ui.min.js
- http://zero.webappsecurity.com:80/resources/js/jquery-1.8.2.min.js
- http://zero.webappsecurity.com:80/docs/config/index.html
- http://zero.webappsecurity.com:80/docs/security-manager-howto.html
- http://zero.webappsecurity.com:80/docs/config/server.html
- http://zero.webappsecurity.com:80/docs/config/cluster-listener.html
- http://zero.webappsecurity.com:80/admin/users.html
- http://zero.webappsecurity.com:80/docs/elapi/index.html
- http://zero.webappsecurity.com:80/bank/account-summary.html
- http://zero.webappsecurity.com:80/docs/appdev/sample/
- http://zero.webappsecurity.com:80/docs/config/cluster-valve.html
- http://zero.webappsecurity.com:80/docs/mbeans-descriptor-howto.html
- http://zero.webappsecurity.com:80/testing/
- http://zero.webappsecurity.com:80/docs/config/service.html
- http://zero.webappsecurity.com:80/docs/config/automatic-deployment.html
- http://zero.webappsecurity.com:80/bank/money-map.html
- http://zero.webappsecurity.com:80/scripts/
- http://zero.webappsecurity.com:80/bank/money-map.html
- http://zero.webappsecurity.com:80/docs/cluster-howto.html
- http://zero.webappsecurity.com:80/docs/building.html
- http://zero.webappsecurity.com:80/docs/tribes/membership.html
- http://zero.webappsecurity.com:80/docs/BUILDING.txt
- http://zero.webappsecurity.com:80/resources/js/jquery-1.6.4.min.js
- http://zero.webappsecurity.com:80/docs/servletapi/index.html

- http://zero.webappsecurity.com:80/docs/config/host.html
- http://zero.webappsecurity.com:80/docs/windows-service-howto.html
- http://zero.webappsecurity.com:80/docs/funcspecs/index.html
- http://zero.webappsecurity.com:80/docs/monitoring.html
- http://zero.webappsecurity.com:80/docs/tribes/transport.html
- http://zero.webappsecurity.com:80/docs/config/cluster.html
- http://zero.webappsecurity.com:80/docs/config/engine.html
- http://zero.webappsecurity.com:80/docs/funcspecs/fs-admin-apps.html
- http://zero.webappsecurity.com:80/docs/developers.html
- http://zero.webappsecurity.com:80/docs/config/
- http://zero.webappsecurity.com:80/docs/tribes/faq.html
- http://zero.webappsecurity.com:80/docs/appdev/sample/sample.war
- http://zero.webappsecurity.com:80/docs/architecture/overview.html
- http://zero.webappsecurity.com:80/backup/
- http://zero.webappsecurity.com:80/docs/logging.html
- http://zero.webappsecurity.com:80/admin/
- http://zero.webappsecurity.com:80/docs/appdev/sample/
- http://zero.webappsecurity.com:80/docs/appdev/introduction.html
- http://zero.webappsecurity.com:80/include/
- http://zero.webappsecurity.com:80/docs/jdbc-pool.html
- http://zero.webappsecurity.com:80/db/
- http://zero.webappsecurity.com:80/faq.html?question=1
- http://zero.webappsecurity.com:80/docs/api/index.html?org/apache/catalina/websocket/package-summary.html
- http://zero.webappsecurity.com:80/errors/
- http://zero.webappsecurity.com:80/docs/tribes/setup.html
- http://zero.webappsecurity.com:80/docs/websocketapi/index.html
- http://zero.webappsecurity.com:80/docs/architecture/
- http://zero.webappsecurity.com:80/docs/config/cluster-receiver.html
- http://zero.webappsecurity.com:80/docs/appdev/processes.html
- http://zero.webappsecurity.com:80/resources/js/bootstrap.min.js
- http://zero.webappsecurity.com:80/errors/errors.log
- http://zero.webappsecurity.com:80/docs/jasper-howto.html
- http://zero.webappsecurity.com:80/resources/js/jquery-1.7.2.min.js
- http://zero.webappsecurity.com:80/resources/css/font-awesome.css
- http://zero.webappsecurity.com:80/docs/appdev/sample/sample.war
- http://zero.webappsecurity.com:80/docs/proxy-howto.html
- http://zero.webappsecurity.com:80/docs/ssi-howto.html
- http://zero.webappsecurity.com:80/docs/tribes/introduction.html
- http://zero.webappsecurity.com:80/docs/appdev/index.html
- http://zero.webappsecurity.com:80/docs/config/valve.html

- http://zero.webappsecurity.com:80/docs/RUNNING.txt
- http://zero.webappsecurity.com:80/signin.html

  Postdata: user_login=foo&user_password=foo&submit=Sign%20in
- http://zero.webappsecurity.com:80/docs/funcspecs/mbean-names.html
- http://zero.webappsecurity.com:80/docs/config/executor.html
- http://zero.webappsecurity.com:80/docs/config/loader.html
- http://zero.webappsecurity.com:80/docs/config/resources.html
- http://zero.webappsecurity.com:80/docs/funcspecs/fs-default.html
- http://zero.webappsecurity.com:80/admin/
- http://zero.webappsecurity.com:80/bank/online-statements.html
- http://zero.webappsecurity.com:80/admin/WS_FTP.LOG
- http://zero.webappsecurity.com:80/docs/comments.html
- http://zero.webappsecurity.com:80/docs/config/realm.html
- http://zero.webappsecurity.com:80/faq.html?question=2
- http://zero.webappsecurity.com:80/docs/cgi-howto.html
- http://zero.webappsecurity.com:80/docs/funcspecs/fs-admin-objects.html
- http://zero.webappsecurity.com:80/docs/config/ajp.html
- http://zero.webappsecurity.com:80/include/common.inc
- http://zero.webappsecurity.com:80/docs/servletapi/index.html
- http://zero.webappsecurity.com:80/admin/index.html
- http://zero.webappsecurity.com:80/docs/architecture/requestProcess.html
- http://zero.webappsecurity.com:80/docs/RELEASE-NOTES.txt
- http://zero.webappsecurity.com:80/docs/jndi-resources-howto.html
- http://zero.webappsecurity.com:80/docs/apr.html
- http://zero.webappsecurity.com:80/docs/
- http://zero.webappsecurity.com:80/docs/maven-jars.html
- http://zero.webappsecurity.com:80/docs/changelog.html
- http://zero.webappsecurity.com:80/docs/architecture/index.html
- http://zero.webappsecurity.com:80/docs/connectors.html
- http://zero.webappsecurity.com:80/resources/css/main.css
- http://zero.webappsecurity.com:80/docs/elapi/
- http://zero.webappsecurity.com:80/docs/jspapi/
- http://zero.webappsecurity.com:80/faq.html
- http://zero.webappsecurity.com:80/admin/currencies-add.html

  Postdata: id=12345&country=12345&name=Jason
- http://zero.webappsecurity.com:80/docs/windows-auth-howto.html
- http://zero.webappsecurity.com:80/docs/config/cluster-membership.html

Response Codes Comparison

URL	Scan A	Scan B

None

File Not Found Comparison

FNF only in 'Site: http://zero.webappsecurity.com/' (A)	File Not Found Reason
None	

FNF only in 'Site: http://zero.webappsecurity.com/' (B)	File Not Found Reason
● http://zero.webappsecurity.com:80/bank/money-map.html	8

Appendix (Check Descriptions)

Critical

Summary

Critical database server error message vulnerabilities were identified in the web application, indicating that an unhandled exception was generated in your web application code. Unhandled exceptions are circumstances in which the application has received user input that it did not expect and does not know how to handle. When successfully exploited, an attacker can gain unauthorized access to the database by using the information recovered from seemingly innocuous error messages to pinpoint flaws in the web application and to discover additional avenues of attack. Recommendations include designing and adding consistent error-handling mechanisms that are capable of handling any user input to your web application, providing meaningful detail to end-users, and preventing error messages that might provide information useful to an attacker from being displayed.

Description

The most common cause of an unhandled exception is a failure to properly sanitize client-supplied data that is used in SQL statements. They can also be caused by a bug in the web application's database communication code, a misconfiguration of database connection settings, an unavailable database, or any other reason that would cause the application's database driver to be unable to establish a working session with the server. The problem is not that web applications generate errors. All web applications in their normal course of operation will at some point receive an unhandled exception. The problem lies not in that these errors were received, but rather in how they are handled. Any error handling solution needs to be well-designed, and uniform in how it handles errors. For instance, assume an attacker is attempting to access a specific file. If the request returns an error File not Found, the attacker can be relatively sure the file does not exist. However, if the error returns "Permission Denied," the attacker has a fairly good idea that the specific file does exist. This can be helpful to an attacker in many ways, from determining the operating system to discovering the underlying architecture and design of the application.

The error message may also contain the location of the file that contains the offending function. This may disclose the webroot's absolute path as well as give the attacker the location of application "include" files or database configuration information. A fundamental necessity for a successful attack upon your web application is reconnaissance. Database server error messages can provide information that can then be utilized when the attacker is formulating his next method of attack. It may even disclose the portion of code that failed.

Be aware that this check is part of unknown application testing which seeks to uncover new vulnerabilities in both custom and commercial software. Because of this, there are no specific patches or remediation information for this issue. Please note that this vulnerability may be a false positive if the page it is flagged on is technical documentation relating to a database server.

Implication

The severity of this vulnerability depends on the reason that the error message was generated. In most cases, it will be the result of the web application attempting to use an invalid client-supplied argument in a SQL statement, which means that SQL injection will be possible. If so, an attacker will at least be able to read the contents of the entire database arbitrarily. Depending on the database server and the SQL statement, deleting, updating and adding records and executing arbitrary commands may also be possible. If a software bug or bug is responsible for triggering the error, the potential impact will vary, depending on the circumstances. The location of the application that caused the error can be useful in facilitating other kinds of attacks. If the file is a hidden or include file, the attacker may be able to gain more information about the mechanics of the web application, possibly even the source code. Application source code is likely to contain usernames, passwords, database connection strings and aids the attacker greatly in discovering new vulnerabilities.

Execution

The ways in which an attacker can exploit the conditions that caused the error depend on its cause. In the case of SQL injection, the techniques that are used will vary from database server to database server, and even query to query. An in-depth guide to SQL Injection attacks is available at http://download.hpsmartupdate.com/asclabs/sql_injection.pdf, or in the SQL Injection vulnerability information, accessible via the Policy Manager. Primarily, the information gleaned from database server error messages is what will allow an attacker to conduct a successful attack after he combines his various findings.

Summary

A critical vulnerability has been detected within your web application due to the presence of one or more Social Security Numbers. If this information is carried over to a production server, it can cause major security problems. Recommendations include not storing this information on your web application.

Implication

Social Security Numbers are a highly sought out prize for attackers, and an item to which a large percentage of time would be dedicated in an effort to find. At a minimum, this can lead to theft of the victim's identity.

High

Medium

Summary

A serious Directory Listing vulnerability was discovered within your web application. Risks associated with an attacker discovering a Directory Listing, which is a complete index of all of the resources located in that directory, result from the fact that files that should remain hidden, such as data files, backed-up source code, or applications in development, may then be visible. The specific risks depend upon the specific files that are listed and accessible. Recommendations include restricting access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that could expose private files and provide information that could be utilized by an attacker when formulating or conducting an attack.

Implication

Risks associated with an attacker discovering a Directory Listing on your application server depend upon what type of directory is discovered, and what types of files are contained within it. The primary threat from an accessible Directory Listing is that hidden files such as data files, source code, or applications under development will then be visible to a potential attacker. In addition to accessing files containing sensitive information, other risks include an attacker utilizing the information discovered in that directory to perform other types of attacks.

Execution

http://zero.webappsecurity.com:80/errors/

Summary

A serious WS_FTP vulnerability was identified within your web application. WS_FTP is a popular FTP client for Windows which is utilized by system administrators and developers to upload and download files from web servers, with each transfer notated in a log file by default. Risks associated with an attacker discovering a WS-FTP log file result from the fact that files that should remain hidden, such as administrative or maintenance applications, web application configuration files, or application data files, may then be visible. Recommendations include removing the WS_FTP log file from the application server and configuring WS_FTP so that it does not create log files.

Implication

When WS_FTP is used to transfer files, a log file called 'ws_ftp.log' is created on the server. This log file contains records of every file that is accessed by WS_FTP, which could possibly contain very valuable information to an attacker because it may list files that are otherwise "hidden." This often includes administrative or maintenance applications, web application configuration files, applications-in-development, backed-up application source code and possible application data files.

Primarily, WS_FTP log files are valuable to attackers because they display all files in a directory, not just ones that are intended to be used. How easy is it for an attacker to take advantage of an insecure web application via the discovery of a WS_FTP log file on your web application server? Often, this is as simple as typing in the name of the file garnered directly from the WS_FTP log files. In essence, gaining access to a WS_TP log file greatly reduces the amount of effort a potential attacker must employ to gain knowledge of your web application.

A fundamental necessity for a successful attack upon your web application is reconnaissance. An attacker will employ a variety of methods, including malicious scanning agents and Google searches, to find out as much information about your web application as possible. That information can then be utilized when the attacker is formulating his next method of attack. An attacker who finds a WS_FTP log files has had a large portion of his reconnaissance conducted for him.

Execution

Click the following link to examine the contents of the WS_FTP log file discovered on your web application server.

http://zero.webappsecurity.com:80/admin/WS_FTP.LOG

Summary

An application include file was found. This results in a possible information disclosure vulnerability, exposing internal application workings to an attacker who can then potentially leverage that information to exploit the application. Recommendations include storing include files in a location other than the webroot.

Implication

An attacker could view web application source code. Web application source code often contains database usernames, passwords and connection strings and locations of sensitive files. It also reveals the detailed mechanics and design of the web application's logic, which can be used to develop other attacks.

Execution

Open a web browser and navigate to http://zero.webappsecurity.com:80/include/common.inc.

Summary

A Cross-Frame Scripting (XFS) vulnerability can allow an attacker to load the vulnerable application inside an HTML iframe tag on a

malicious page. The attacker could use this weakness to devise a Clickjacking attack to conduct phishing, frame sniffing, social engineering or Cross-Site Request Forgery attacks.

Clickjacking
The goal of a Clickjacking attack is to deceive the victim (user) into interacting with UI elements of the attacker's choice on the target web site without their knowledge and then executing privileged functionality on the victim's behalf. To achieve this goal, the attacker must exploit the XFS vulnerability to load the attack target inside an iframe tag, hide it using Cascading Style Sheets (CSS) and overlay the phishing content on the malicious page. By placing the UI elements on the phishing page so they overlap with those on the page targeted in the attack, the attacker can ensure that the victim must interact with the UI elements on the target page not visible to the victim.

WebInspect has detected a response containing one or more forms that accept user input but is missing XFS protection.

This response is not protected by a valid X-Frame-Options header and a valid Content-Security-Policy(CSP) frame-ancestors directive header.Furthermore,
An effective frame-busting technique was not observed while loading this page inside a frame.

Implication

A Cross-Frame Scripting weakness could allow an attacker to embed the vulnerable application inside an iframe. Exploitation of this weakness could result in:

Hijacking of user events such as keystrokes
Theft of sensitive information
Execution of privileged functionality through combination with Cross-Site Request Forgery attacks

Execution

Create a test page containing an HTML iframe tag whose src attribute is set to http://zero.webappsecurity.com:80/. Successful framing of the target page indicates that the application is susceptibile to XFS.

Note that WebInspect will report only one instance of this check across each host within the scope of the scan. The other visible pages on the site may, however, be vulnerable to XFS as well and therefore should be protected against it with an appropriate fix.

Summary

WebInspect has detected the application to be vulnerable to an HTTP Request Smuggling attack.

HTTP Request Smuggling vulnerabilities arise due to the discrepancy in parsing of non-compliant HTTP headers by the front-end and back-end servers. By supplying a request that is interpreted as being of different lengths by different servers, an attacker can poison the back-end TCP/TLS socket and prepend arbitrary data to the next request or smuggle additional requests to the back-end server without the front-end server being aware of it.

There are numerous ways in which a malicious user can accomplish an HTTP Request Smuggling attack. In this instance, an incoming HTTP request that contains both Content-Length and Transfer-Encoding headers is interpreted differently by the front-end server and the back-end server. One honors the Content-Length header and the other the Transfer-Encoding header to determine the length of the request. This can render the application vulnerable to smuggling attacks.

The mean response time for the reported HTTP request/response is seconds.
The attack timed out after seconds.

WebInspect was unable to verify the vulnerability in this instance. However, we recommend that you inspect your network and configuration settings of all chained elements in the network to ensure that the application is not vulnerable to the HTTP Request Smuggling vulnerability.

Implication

Malicious users can use the HTTP Request Smuggling vulnerability to bypass front-end security rules, access internal systems, and poison web caches. In addition, it can also be exploited to steal user information and launch cross-site scripting attacks.

Low

Summary

A minor vulnerability has been discovered within your web application due to the the presence of a fully qualified path name to the root of your system. This most often occurs in context of an error being produced by the web application. Fully qualified server path names allow an attacker to know the file system structure of the web server, which is a baseline for many other types of attacks to be successful. Recommendations include adopting a consistent error handling scheme and mechanism that prevents fully qualified path names from being displayed.

Execution

To verify the issue, click the 'HTTP Response' button on the properties view and review the highlighted areas to determine the Unix path found.

Summary

System Environment variables log files contain information about the nature of your web application, and would allow an attacker to gain insightful information about the web system setup. Recommendations include removing this file from the affected system.

Implication

A fundamental part of any successful attack is reconnaissance and information gathering. The primary danger from exploitation of this vulnerability is that an attacker will be able to utilize the information in launching a more serious attack. It is very simple to check for its existence, and a file most definitely on the short list of things for which a potential attacker would look.

Summary

A string matching an internal/reserved IPv4 or IPv6 address range was discovered. This may disclose information about the IP addressing scheme of the internal network and can be valuable to attackers.Internal IPv4/IPv6 ranges are:
10.x.x.x
172.16.x.x through 172.31.x.x
192.168.x.x
fd00::x
If not a part of techical documentation, recommendations include removing the string from the production server.

Summary

Administrative directories were discovered within your web application during a Directory Enumeration scan. Risks associated with an attacker discovering an administrative directory on your application server typically include the potential for the attacker to use the administrative applications to affect the operations of the web site. Recommendations include restricting access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that provide information that could be utilized by an attacker when formulating or conducting an attack.

Implication

The primary danger from an attacker finding a publicly available directory on your web application server depends on what type of directory it is, and what files it contains. Administrative directories typically contain applications capable of changing the configuration of the running software; an attacker who gains access to an administrative application can drastically affect the operation of the web site.

Summary

Directory Enumeration vulnerabilities were discovered within your web application. Risks associated with an attacker discovering a directory on your application server depend upon what type of directory is discovered, and what types of files are contained within it. The primary threat, other than accessing files containing sensitive information, is that an attacker can utilize the information discovered in that directory to perform other types of attacks. Recommendations include restricting access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that provide information that could be utilized by an attacker when formulating or conducting an attack.

Implication

The primary danger from an attacker finding a publicly available directory on your web application server depends on what type of directory it is, and what files it contains.

Summary

Directory Enumeration vulnerabilities were discovered within your web application. Risks associated with an attacker discovering a directory on your application server depend upon what type of directory is discovered, and what types of files are contained within it. The primary threat, other than accessing files containing sensitive information, is that an attacker can utilize the information discovered in that directory to perform other types of attacks. Recommendations include restricting access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that provide information that could be utilized by an attacker when formulating or conducting an attack.

Summary

Directory Enumeration vulnerabilities were discovered within your web application. Risks associated with an attacker discovering a directory on your application server depend upon what type of directory is discovered, and what types of files are contained within it. The primary threat, other than accessing files containing sensitive information, is that an attacker can utilize the information

discovered in that directory to perform other types of attacks. Recommendations include restricting access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that provide information that could be utilized by an attacker when formulating or conducting an attack.

Summary

Data-related directories were discovered within your web application during a Directory Enumeration. Risks associated with an attacker discovering a directory on your application server depend upon what type of directory is discovered, and what types of files are contained within it. The primary threat, other than accessing files containing sensitive information, is that an attacker can utilize the information discovered in that directory to perform other types of attacks. Recommendations include restricting access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that provide information that could be utilized by an attacker when formulating or conducting an attack.

Summary

Development-related directories were discovered within your web application during a Directory Enumeration scan. Risks associated with an attacker discovering a directory on your application server depend upon what type of directory is discovered, and what types of files are contained within it. The primary threat, other than accessing files containing sensitive information, is that an attacker can utilize the information discovered in that directory to perform other types of attacks. Recommendations include removing any source code directories and repositories from the production server, disabling the use of remote repositories, and ensuring that the latest patches and version updates have been performed on the version control system being used Additionally, restrict access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that provide information that could be utilized by an attacker when formulating or conducting an attack.

Implication

An attacker may use the internal information obtained from the source code files to craft a precise attack against the web application. Such attacks can include, but are not limited to, SQL injection, remote file system access, malware injection and database manipulation.

Execution

Browse to http://zero.webappsecurity.com:80/testing/ and inspect the content. Response should return with HTTP status code 200 and should not match target site’s file not found response.

Summary

Directory Enumeration vulnerabilities were discovered within your web application. Risks associated with an attacker discovering a directory on your application server depend upon what type of directory is discovered, and what types of files are contained within it. The primary threat, other than accessing files containing sensitive information, is that an attacker can utilize the information discovered in that directory to perform other types of attacks. Recommendations include restricting access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that provide information that could be utilized by an attacker when formulating or conducting an attack.

Summary

E-Commerce and/or financial-related directories were discovered within your web application during a Directory Enumeration scan. Risks associated with an attacker discovering a directory on your application server depend upon what type of directory is discovered, and what types of files are contained within it. The primary threat, other than accessing files containing sensitive information, is that an attacker can utilize the information discovered in that directory to perform other types of attacks. Recommendations include restricting access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that provide information that could be utilized by an attacker when formulating or conducting an attack.

Summary

Logfile and/or statistic directories were discovered within your web application during a Directory Enumeration scan. Risks associated with an attacker discovering a directory on your application server depend upon what type of directory is discovered, and what types of files are contained within it. The primary threat, other than accessing files containing sensitive information, is that an attacker can utilize the information discovered in that directory to perform other types of attacks. Recommendations include restricting access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that provide information that could be utilized by an attacker when formulating or conducting an attack.

Summary

Directories that may be restricted to only certain users were discovered within your web application during a Directory Enumeration scan. Risks associated with an attacker discovering a directory on your application server depend upon what type of directory is discovered, and what types of files are contained within it. The primary threat, other than accessing files containing sensitive information, is that an attacker can utilize the information discovered in that directory to perform other types of attacks. Recommendations include restricting access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that provide information that could be utilized by an attacker when formulating or conducting an attack.

Summary

A minor vulnerability has been detected within your web application due to the discovery of a fully qualified path name to the root of your system. This most often occurs in context of an error being produced by the web application. Fully qualified server path names allow an attacker to know the file system structure of the web server, which is a baseline for many other types of attacks to be successful. Recommendations include adopting a consistent error handling scheme and mechanism that prevents fully qualified path names from being displayed.

Summary

A directory named 'admin' was discovered within your web application. Risks associated with an attacker discovering an administrative directory on your application server typically include the potential for the attacker to use the administrative applications to affect the operations of the web site. Recommendations include restricting access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that provide information that could be utilized by an attacker when formulating or conducting an attack.

Implication

Administrative directories typically contain applications capable of changing the configuration of the running software; an attacker who gains access to an administrative application can drastically affect the operation of the web site.

Summary

A possible LDAP query was discovered. This could reveal variable names, path information, and other things of value to a potential attacker. Recommendations include not hard-coding LDAP query strings in your application code.

Implication

An attacker who discovers an LDAP query string could orchestrate more damaging attacks such as LDAP Injection which could be utilized to retrieve information from the LDAP server.

Summary

A server error response was detected. The server could be experiencing errors due to a misbehaving application, a misconfiguration, or a malicious value sent during the auditing process. While error responses in and of themselves are not dangerous, per se, the error responses give attackers insight into how the application handles error conditions. Errors that can be remotely triggered by an attacker can also potentially lead to a denial of service attack or other more severe vulnerability. Recommendations include designing and adding consistent error handling mechanisms which are capable of handling any user input to your web application, providing meaningful detail to end-users, and preventing error messages that might provide information useful to an attacker from being displayed.

Implication

The server has issued a 500 error response. While the body content of the error page may not expose any information about the technical error, the fact that an error occurred is confirmed by the 500 status code. Knowing whether certain inputs trigger a server error can aid or inform an attacker of potential vulnerabilities.

Summary

A resource on the target website has been found to be shared across websites using CORS with an open access control policy.

Cross-Origin Resource Sharing, commonly referred to as CORS, is a technology that allows a domain to define a policy for its resources to be accessed by a web page hosted on a different domain using cross domain XML HTTP Requests (XHR). Historically, the browser restricts cross domain XHR requests to abide by the same origin policy. At its basic form, the same origin policy sets the script execution scope to the resources available on the current domain and prohibits any communication to domains outside this scope. While CORS is supported on all major browsers, it also requires that the domain correctly defines the CORS policy in order to have its resources shared with another domain. These restrictions are managed by access policies typically included in specialized response headers, such as:

- Access-Control-Allow-Origin

- Access-Control-Allow-Headers
- Access-Control-Allow-Methods

A domain includes a list of domains that are allowed to make cross domain requests to shared resources in Access-Control-Allow-Origin header. This header can have either list of domains or a wildcard character ("*") to allow all access. Having a wildcard is considered overly permissive policy.

Implication

An overly permissive CORS policy can allow a malicious application to communicate with the victim application in an inappropriate way, leading to spoofing, data theft, relay and other attacks. It can open possibilities for entire domain compromise. For example, let's say a Resource is located on a private intranet and a universal access policy is created with the intent that only other intranet domains can reach it. Subsequently, an internal employee browses to an Internet resource that includes a malicious embedded JavaScript that enumerates the private resource and enables external accessibility; effectively exposing it to the Internet. If the resource discloses any sensitive information, this attack can quickly escalate into an unintentional breach of sensitive information.

Summary

Almost all browsers are designed to use a mime sniffing technique to guess the content type of the HTTP response instead of adhering to the Content-Type specified by the application in specific cases or ignoring the content when no mime type is specified. Inconsistencies introduced by the mime sniffing techniques could allow attackers to conduct Cross-Site Scripting attacks or steal sensitive user data. WebInspect has determined that the application fails to instruct the browser to strictly enforce the Content-Type specification supplied in the response.

Web server misconfiguration can cause an application to send HTTP responses with the missing Content-Type header or specify a mime type that does not match up accurately with the response content. When a browser receives such a response, it attempts to programmatically determine the mime type based on the content returned in the response. The mime type derived by the browser, however, might not accurately match the one intended by the application developer. Such inconsistencies have historically allowed attackers to conduct Cross-Site Scripting or data theft using Cascading Style Sheets (CSS) by letting them bypass server-side filters using mime type checking and yet have the malicious payload with misleading mime type specification executed on the client-side due to the browser mime sniffing policies.

Microsoft Internet Explorer (IE) introduced the X-Content-Type-Options: nosniff specification that application developers can include in all responses to ensure that mime sniffing does not occur on the client-side. This protection mechanism is limited to Microsoft Internet Explorer versions 9 and above.

Implication

By failing to dictate the suitable browser interpretation of the response content, application developers can expose their users to Cross-Site Scripting or information stealing attacks.

Execution

. Build a test page that includes a reference to an external JavaScript or CSS resource

. Configure the server to return the external resource with an incorrect mime type specification

. Visit the test page using an old version of Microsoft's Internet Explorer (version IE 8) browser

. Interpretation of the external content as JavaScript or CSS by the browser despite the misleading mime type specification indicates a potential for compromise.

Summary

X-XSS-Protection HTTP response header enables developers and security architects to manage browser protection against reflected cross-site scripting. The mechanism is also known as the XSS Auditor in Chrome and the XSS filter in Internet Explorer. In modern browsers, the Content-Security-Policy header can provide better protection against XSS and setting X-XSS-Protection might be redundant. However, this header can reduce the risk of reflected XSS attacks in earlier browsers that do not support CSP.

This header can be set to one of three possible values: `0`, `1`, or `1; mode=block` . A value of `0` disables the protection. A value of `1` is the default behaviour in modern browsers that enables the protection in filter or replacement mode. For example, IE replaces JavaScript keywords such as <script> with <scr#pt> to render injected string ineffective. The value of `1; mode=block` instructs browsers to block the response from rendering in the browser. Reports of multiple exploits that leverage false positives from

default behaviour that filters or replaces JavaScript injection string within the response r eturned from server. Therefore, the current recommendation is to set the header in block mode.

Implication

Attackers may leverage zero day reflective XSS against a site. If the header is not set in block mode, an attacker can use browser-specific filter bypass bugs to succeed in launching a reflected XSS against the site.

Execution

Click the response tab for the highlighted request. The response header X-XSS-Protection is either missing or set to `1`.

By default, WebInspect flags only one instance of this vulnerability per host because it is typical to set this header at the host level in a server configuration.

Perform the following steps to flag all instances of this issue:

- Create a new policy with the selection of checks that you want to include in a rescan. We recommend using the Blank or Passive policy as a base.
- Select this check and unselect the check input, "FlagAtHost",from standard description window.
- Save the policy.
- Rescan with this new custom policy.

Informational

Summary

The Ws_ftp.log parser engine will parse any Ws_ftp.log files found within the scan for links to add to the crawler engine.

Summary

Content Security Policy (CSP) is an HTTP response security header that developers and security architects can leverage to whitelist domains from which the site is allowed to load resources. This header provides an in-depth security protection from critical vulnerabilities such as cross-site scripting and clickjacking. Additionally, CSP restricts execution of inline JavaScript, dynamic JavaScript code evaluation from strings, and framing of the site from external domains. While CSP is not a replacement for input validation, it can help to significantly reduce the risk of XSS from unknown weaknesses. The CSP frame-ancestors directive is equivalent to X-Frame-Options and restricts the domain that are allowed to frame the site's content.

Implication

Security architects and developers can leverage CSP to significantly reduce the risk of XSS and clickjacking attacks. CSP headers can restrict leakage of information to external domains by restricting which domains the site is allowed to load contents from when rendered in browser .

Execution

Access link http://zero.webappsecurity.com:80/ through a proxy and notice the missing CSP header in the response. By default, WebInspect flags only one instance of this vulnerability per host because it is typical to set this header at the host level in a server configuration.

Perform the following steps to flag all instances of this issue:

- Create a new policy with the selection of checks that you want to include in a rescan. We recommend using the Blank or Passive policy as a base.
- Select this check and uncheck the "FlagAtHost" check input from standard description.
- Save the policy.
- Rescan with this new custom policy.

Summary

Most recent browsers have features that will save form field content entered by users and then automatically complete form entry the next time the fields are encountered. This feature is enabled by default and could leak sensitive information since it is stored on the hard drive of the user. The risk of this issue is greatly increased if users are accessing the application from a shared environment. Recommendations include setting autocomplete to "off" on all your forms.

Summary

The Content-Type HTTP response header or the HTML meta tag provides a mechanism for the server to specify an appropriate character encoding for the response content to be rendered in the web browser. Proper specification of the character encoding through the charset parameter in the Content-Type field reduces the likelihood of misinterpretation of the characters in the response content and ensure reliable rendering of the web page.Failure to ensure enforcement of the desired character encoding could result in client-side attacks like Cross-Site Scripting.

Implication

In the absence of the character set specification, a user-agent might default to a non-standard character set, or could derive an incorrect character set based on certain characters in the response content. In some cases, both these approaches can cause the response to be incorrectly rendered. This may enable other attacks such as Cross-site Scripting.

Execution

Verify the character set specification on every HTTP response. Character sets can be specified in the HTTP header or in an HTML meta tag. In the case of an XML response, the character set can be specified along with the XML Declaration.

Scan Name:	Site: http://zero.webappsecurity.com/		
Policy:	Standard	**Crawl Sessions:**	16
Scan Date:	12/30/2019 3:47:31 PM	**Vulnerabilities:**	24
Scan Version:	19.2.0.184	**Scan Duration:**	3 minutes : 14 seconds
Scan Type:	Site	**Client:**	FF

Server: http://zero.webappsecurity.com:80

Critical Issues

Cross-Site Scripting: Reflected (5649)

View Description

CWE: 79,80,82,83,87,116,692,811

Kingdom: Input Validation and Representation

Page: http://zero.webappsecurity.com:80/faq.html?question=2%3c%73%43%72%49%70%54%3e%61%6c%65%72%74%28%31%33%31%31%35%29%3c%2f%73%43%72%49%70%54%3e

Parameter: question

Page: http://zero.webappsecurity.com:80/search.html?searchTerm=12345%3c%73%43%72%3c%53%63%52%69%50%74%3e%49%70%54%3e%61%6c%65%72%74%28%36%34%32%32%35%29%3c%2f%73%43%72%3c%53%63%52%69%50%74%3e%49%70%54%3e

Parameter: searchTerm

Cross-Site Scripting: Reflected (5650)

View Description

CWE: 79,80,82,83,87,116,692,811

Kingdom: Input Validation and Representation

Page: http://zero.webappsecurity.com:80/sendFeedback.html

Parameter: name

High Issues

Web Server Misconfiguration: Unprotected File (157)

View Description

CWE: 200

Kingdom: Environment

Page: http://zero.webappsecurity.com:80/server-status

Web Server Misconfiguration: Unprotected File (708)

View Description

CWE: 200

Kingdom: Environment

Page: http://zero.webappsecurity.com:80/faq.html.bak

Web Server Misconfiguration: Unprotected File (709) View Description

CWE: 200

Kingdom: Environment

Page: http://zero.webappsecurity.com:80/index.html.old

Web Server Misconfiguration: Unprotected File (1368) View Description

CWE: 284,200

Kingdom: Environment

Page: http://zero.webappsecurity.com:80/debug.txt

Web Server Misconfiguration: Unprotected File (2083) View Description

CWE: 200

Kingdom: Environment

Page: http://zero.webappsecurity.com:80/index.old

Insecure Transport (4722) View Description

CWE: 287

Kingdom: Security Features

Page: http://zero.webappsecurity.com:80/forgot-password.html

Page: http://zero.webappsecurity.com:80/login.html

Often Misused: Login (10595) View Description

CWE: 287,311

Kingdom: API Abuse

Page: http://zero.webappsecurity.com:80/login.html

Page: http://zero.webappsecurity.com:80/forgot-password.html

Cross-Frame Scripting (11293) View Description

CWE: 352

Kingdom: Security Features

Page: http://zero.webappsecurity.com:80/login.html

Expression Language Injection (11310) View Description

CWE: 78

Kingdom: Input Validation and Representation

Page: http://zero.webappsecurity.com:80/search.html?searchTerm=${7166%2b2327}

Expression Language Injection (11319) View Description

CWE: 78

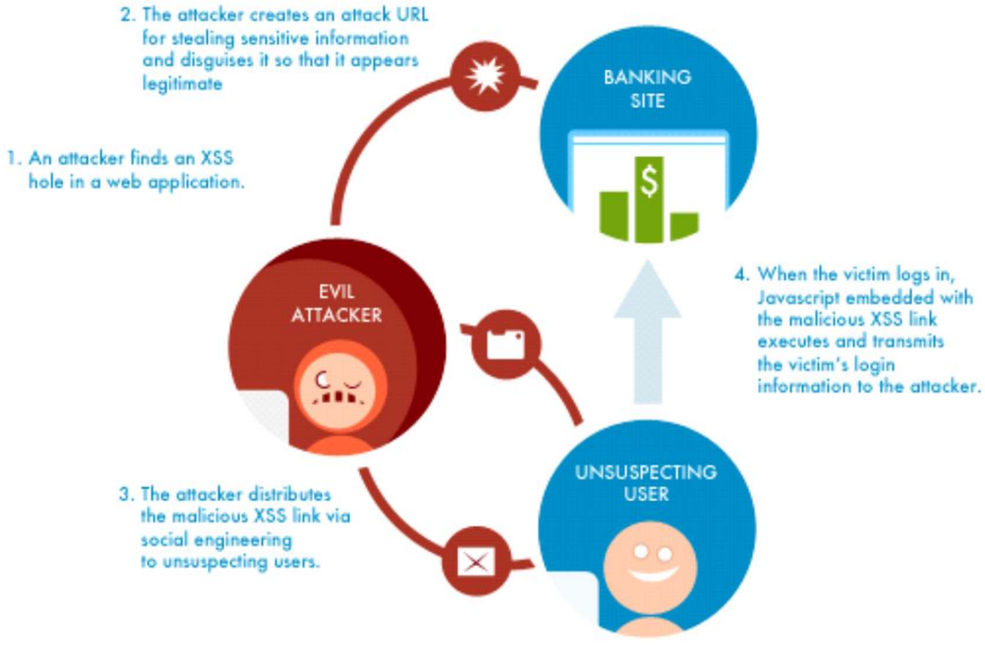
Kingdom: Input Validation and Representation

Page: http://zero.webappsecurity.com:80/search.html?searchTerm=${cookie%5B%22JSESSIONID%22%5D%2Evalue}

Appendix (Check Descriptions)

Cross-Site Scripting: Reflected (5649)

Summary



Cross-Site Scripting vulnerability found in Get parameter question. The following attack uses plain encoding:

<sCrIpT>alert(13115)</sCrIpT>

A Cross-Site Scripting (XSS) vulnerability was detected in the web application. Cross-Site Scripting occurs when dynamically generated web pages display user input, such as login information, that is not properly validated, allowing an attacker to embed malicious scripts into the generated page and then execute the script on the machine of any user that views the site. In this instance, the web application was vulnerable to an automatic payload, meaning the user simply has to visit a page to make the malicious scripts execute. If successful, Cross-Site Scripting vulnerabilities can be exploited to manipulate or steal cookies, create requests that can be mistaken for those of a valid user, compromise confidential information, or execute malicious code on end user systems. Recommendations include implementing secure programming techniques that ensure proper filtration of user-supplied data, and encoding all user supplied data to prevent inserted scripts being sent to end users in a format that can be executed.

Execution

View the attack string included with the request to check what to search for in the response. For instance, if "(javascript:alert ('XSS')" is submitted as an attack (or another scripting language), it will also appear as part of the response. This indicates that the web application is taking values from the HTTP request parameters and using them in the HTTP response without first removing potentially malicious data.The response can be viewed in "Web Browser" view in the Vulnerability pane to see the injected popup events in action. Events requiring user interaction (e.g. onmouseover or onclick events) can be triggered by performing the corresponding action (e.g. clicking the injected link). Injection with numeric string in src, or href, attributes indicates that the site is vulnerable to script include or content exfiltration. These can be verified by repeating the request in a browser and intercepting originating network traffic in a web proxy.

Implication

Cross-Site Scripting(XSS) happens when user input from a web client is immediately included via server-side scripts in a dynamically generated web page. Reflected XSS is specifically considered critical when malicious payload can be embedded in a URL (e.g. in query strings of GET requests). An attacker can trick a victim, via phishing attack, to click on a link with vulnerable input which has been altered to include attack code and then sent to the legitimate server. The injected code is

then reflected back to the user's browser which executes it.

The implications of successful Cross-Site Scripting attacks are:

- Account hijacking - An attacker can hijack the user's session before the session cookie expires and take actions with the privileges of the user who accessed the URL, such as issuing database queries and viewing the results.
- Malicious script execution - Users can unknowingly execute JavaScript, VBScript, ActiveX, HTML, or even Flash content that has been inserted into a dynamically generated page by an attacker.
- Worm propagation - With Ajax applications, XSS can propagate somewhat like a virus. The XSS payload can autonomously inject itself into pages, and easily re-inject the same host with more XSS, all of which can be done with no hard refresh. Thus, XSS can send multiple requests using complex HTTP methods to propagate itself invisibly to the user.
- Information theft - Via redirection and fake sites, attackers can connect users to a malicious server of the attacker's choice and capture any information entered by the user.
- Denial of Service - Often by utilizing malformed display requests on sites that contain a Cross-Site Scripting vulnerability, attackers can cause a denial of service condition to occur by causing the host site to query itself repeatedly .
- Browser Redirection - On certain types of sites that use frames, a user can be made to think that he is in fact on the original site when he has been redirected to a malicious one, since the URL in the browser's address bar will remains the same. This is because the entire page isn't being redirected, just the frame in which the JavaScript is being executed is redirected.
- Manipulation of user settings - Attackers can change user settings for nefarious purposes.
- Bypass Content-Security-Policy protection - Attackers can inject a malformed tag formation, known as dangling tag injection, which in some cases allows injected script to reuse valid nonce on the page and bypass script source restriction. Additionally dangling tag injection can be used to steal sensitive information embedded in HTML response if browser is able to make a request to the injected link.
- Base tag injection: Attacker can cause relative links on a page to load from a different domain by modifying the base URL for the page via base tag injection.
- Link prefetch injection: While unable to execute script, attackers can use link tag with rel=prefetch that will make browsers pre-fetch the specified link even though it is never rendered and rejected subsequently due to web application enforced cross-site policy (e.g. CSP protections).
- Edge side includes (ESI) Injection - ESI is a markup language used in various HTTP devices, such as reverse proxies and load balancers, that are positioned between client and server. An attacker can inject ESI markup to perform critical attacks such as cross-site scripting and HTTPOnly cookie protection bypass.

Fix

For Development:

Cross-Site Scripting attacks can be avoided by carefully validating all input, and properly encoding all output. When validating user input, verify that it matches the strictest definition of valid input possible. For example, if a certain parameter is supposed to be a number, attempt to convert it to a numeric data type in your programming language.

PHP: intval("0".$_GET['q']);

ASP.NET: int.TryParse(Request.QueryString["q"], out val);

The same applies to date and time values, or anything that can be converted to a stricter type before being used. When accepting other types of text input, make sure the value matches either a list of acceptable values (white-listing), or a strict regular expression. If at any point the value appears invalid, do not accept it. Also, do not attempt to return the value to the user in an error message.

Most server side scripting languages provide built in methods to convert the value of the input variable into correct, non-interpretable HTML. These should be used to sanitize all input before it is displayed to the client.

PHP: string htmlspecialchars (string string [, int quote_style])

ASP.NET: Server.HTMLEncode (strHTML String)

When reflecting values into JavaScript or another format, make sure to use a type of encoding that is appropriate. Encoding data for HTML is not sufficient when it is reflected inside of a script or style sheet. For example, when reflecting data in a JavaScript string, make sure to encode all non-alphanumeric characters using hex (\xHH) encoding.

If you have JavaScript on your page that accesses unsafe information (like location.href) and writes it to the page (either with document.write, or by modifying a DOM element), make sure you encode data for HTML before writing it to the page. JavaScript does not have a built-in function to do this, but many frameworks do. If you are lacking an available function, something like the following will handle most cases:

s = s.replace(/&/g,'&').replace(/"/i,'"').replace(/</i,'<').replace(/>/i,'>').replace(/'/i,''')

Ensure that you are always using the right approach at the right time. Validating user input should be done as soon as it is received. Encoding data for display should be done immediately before displaying it.

For Security Operations:

Server-side encoding, where all dynamic content is first sent through an encoding function where Scripting tags will be replaced with codes in the selected character set, can help to prevent Cross-Site Scripting attacks.

Many web application platforms and frameworks have some built-in support for preventing Cross-Site Scripting. Make sure that any built-in protection is enabled for your platform. In some cases, a misconfiguration could allow Cross-Site Scripting. In ASP.NET, if a page's EnableViewStateMac property is set to False, the ASP.NET view state can be used as a vector for Cross-Site Scripting.

An IDS or IPS can also be used to detect or filter out XSS attacks. Below are a few regular expressions that will help detect Cross-Site Scripting.

Regex for a simple XSS attack:
/((\%3C) <)((\%2F) \/)*[a-z0-9\%]+((\%3E) >)/ix

The above regular expression would be added into a new Snort rule as follows:

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"NII Cross-Site Scripting attempt"; flow:to_server,established; pcre:"/((\%3C) <)((\%2F) \/)*[a-z0-9\%]+((\%3E) >)/i"; classtype:Web-application-attack; sid:9000; rev:5;)

Paranoid regex for XSS attacks:
/((\%3C) <)[^\n]+((\%3E) >)/I

This signature simply looks for the opening HTML tag, and its hex equivalent, followed by one or more characters other than the new line, and then followed by the closing tag or its hex equivalent. This may end up giving a few false positives depending upon how your web application and web server are structured, but it is guaranteed to catch anything that even remotely resembles a Cross-Site Scripting attack.

For QA:

Fixes for Cross-Site Scripting defects will ultimately require code based fixes. Read the the following links for more information about manually testing your application for Cross-Site Scripting.

Reference

OWASP Cross-Site Scripting Information
https://www.owasp.org/index.php/XSS

CERT
http://www.cert.org/advisories/CA-2000-02.html

Apache
http://httpd.apache.org/info/css-security/apache_specific.html

SecurityFocus.com
http://www.securityfocus.com/infocus/1768

Classifications

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
http://cwe.mitre.org/data/definitions/79.html

CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)
http://cwe.mitre.org/data/definitions/80.html

CWE-82: Improper Neutralization of Script in Attributes of IMG Tags in a Web Page
http://cwe.mitre.org/data/definitions/82.html

CWE-83: Improper Neutralization of Script in Attributes in a Web Page
http://cwe.mitre.org/data/definitions/83.html

CWE-87: Improper Neutralization of Alternate XSS Syntax
http://cwe.mitre.org/data/definitions/87.html

CWE-116: Improper Encoding or Escaping of Output
http://cwe.mitre.org/data/definitions/116.html

CWE-692: Incomplete Blacklist to Cross-Site Scripting

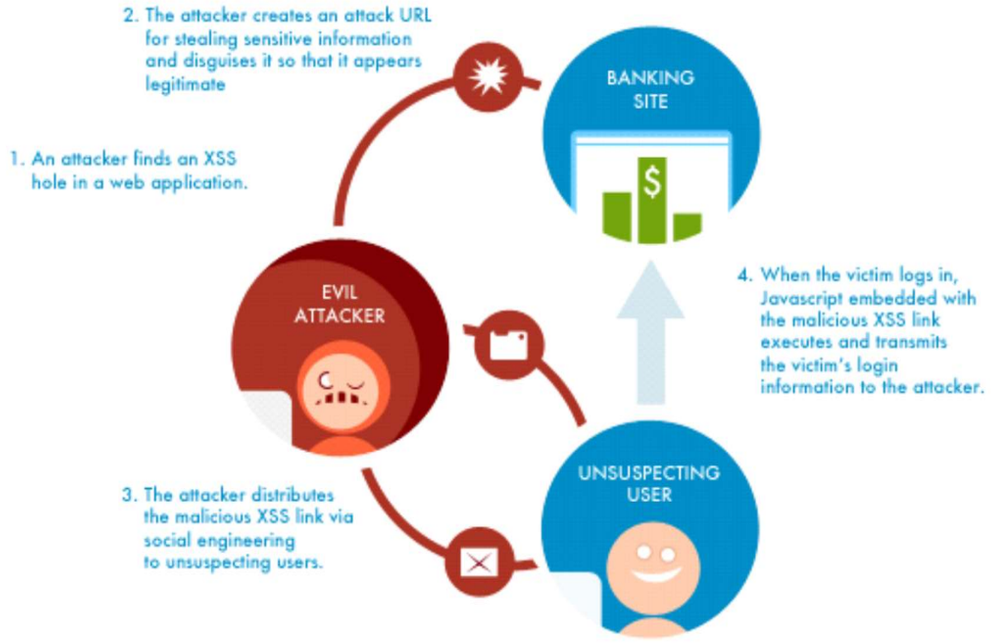
http://cwe.mitre.org/data/definitions/692.html

CWE-811: OWASP Top Ten 2010 Category A2 - Cross-Site Scripting (XSS)
http://cwe.mitre.org/data/definitions/811.html

Kingdom: Input Validation and Representation
https://vulncat.fortify.com/

Cross-Site Scripting: Reflected (5650)

Summary



Cross-Site Scripting vulnerability found in Post parameter name. Triggering this vulnerability requires user action. The following attack uses plain encoding:

<a HrEf=JaVaScRiPt:alert(52745)>

Cross-Site Scripting occurs when dynamically generated web pages display user input, such as login information, that is not properly validated, allowing an attacker to embed malicious scripts into the generated page and then execute the script on the machine of any user that views the site. User interaction vulnerabilities such as this one require the user to trigger the execution of the malicious scripts via an action such as clicking a link or moving the mouse pointer over text. If successful, Cross-Site Scripting vulnerabilities can be exploited to manipulate or steal cookies, create requests that can be mistaken for those of a valid user, compromise confidential information, or execute malicious code on end user systems. Recommendations

include implementing secure programming techniques that ensure proper filtration of user-supplied data, and encoding all user supplied data to prevent inserted scripts being sent to end users in a format that can be executed.

Execution

View the attack string included with the request to check what to search for in the response. For instance, if "(javascript:alert ('XSS')" is submitted as an attack, it will also appear as part of the response. This indicates that the web application is taking values from the HTTP request parameters and using them in the HTTP response without first removing potentially malicious data.

Implication

XSS can generally be subdivided into two categories: stored and reflected attacks. The main difference between the two is in how the payload arrives at the server. Stored attacks are just that...in some form stored on the target server, such as in a database, or via a submission to a bulletin board or visitor log. The victim will retrieve and execute the attack code in his browser when a request is made for the stored information. Reflected attacks, on the other hand, come from somewhere else. This happens when user input from a web client is immediately included via server-side scripts in a dynamically generated web page. Via some social engineering, an attacker can trick a victim, such as through a malicious link or "rigged" form, to submit information which will be altered to include attack code and then sent to the legitimate server. The injected code is then reflected back to the user's browser which executes it because it came from a trusted server. The implication of each kind of attack is the same.

The main problems associated with successful Cross-Site Scripting attacks are:

- Account hijacking - An attacker can hijack the user's session before the session cookie expires and take actions with the privileges of the user who accessed the URL, such as issuing database queries and viewing the results.
- Malicious script execution - Users can unknowingly execute JavaScript, VBScript, ActiveX, HTML, or even Flash content that has been inserted into a dynamically generated page by an attacker.
- Worm propagation - With Ajax applications, XSS can propagate somewhat like a virus. The XSS payload can autonomously inject itself into pages, and easily re-inject the same host with more XSS, all of which can be done with no hard refresh. Thus, XSS can send multiple requests using complex HTTP methods to propagate itself invisibly to the user.
- Information theft - Via redirection and fake sites, attackers can connect users to a malicious server of the attacker's choice and capture any information entered by the user.
- Denial of Service - Often by utilizing malformed display requests on sites that contain a Cross-Site Scripting vulnerability, attackers can cause a denial of service condition to occur by causing the host site to query itself repeatedly .
- Browser Redirection - On certain types of sites that use frames, a user can be made to think that he is in fact on the original site when he has been redirected to a malicious one, since the URL in the browser's address bar will remains the same. This is because the entire page isn't being redirected, just the frame in which the JavaScript is being executed.
- Manipulation of user settings - Attackers can change user settings for nefarious purposes.

Fix

For Development:

Cross-Site Scripting attacks can be avoided by carefully validating all input, and properly encoding all output. When validating user input, verify that it matches the strictest definition of valid input possible. For example, if a certain parameter is supposed to be a number, attempt to convert it to a numeric data type in your programming language.

PHP: intval("0".$_GET['q']);

ASP.NET: int.TryParse(Request.QueryString["q"], out val);

The same applies to date and time values, or anything that can be converted to a stricter type before being used. When accepting other types of text input, make sure the value matches either a list of acceptable values (white-listing), or a strict regular expression. If at any point the value appears invalid, do not accept it. Also, do not attempt to return the value to the user in an error message.

Most server side scripting languages provide built in methods to convert the value of the input variable into correct, non-interpretable HTML. These should be used to sanitize all input before it is displayed to the client.

PHP: string htmlspecialchars (string string [, int quote_style])

ASP.NET: Server.HTMLEncode (strHTML String)

When reflecting values into JavaScript or another format, make sure to use a type of encoding that is appropriate. Encoding data for HTML is not sufficient when it is reflected inside of a script or style sheet. For example, when reflecting data in a JavaScript string, make sure to encode all non-alphanumeric characters using hex (\xHH) encoding.

If you have JavaScript on your page that accesses unsafe information (like location.href) and writes it to the page (either with document.write, or by modifying a DOM element), make sure you encode data for HTML before writing it to the page.

JavaScript does not have a built-in function to do this, but many frameworks do. If you are lacking an available function, something like the following will handle most cases:

s = s.replace(/&/g,'&').replace(/"/i,'"').replace(/</i,'<').replace(/>/i,'>').replace(/'/i,''')

Ensure that you are always using the right approach at the right time. Validating user input should be done as soon as it is received. Encoding data for display should be done immediately before displaying it.

For Security Operations:

Server-side encoding, where all dynamic content is first sent through an encoding function where Scripting tags will be replaced with codes in the selected character set, can help to prevent Cross-Site Scripting attacks.

Many web application platforms and frameworks have some built-in support for preventing Cross-Site Scripting. Make sure that any built-in protection is enabled for your platform. In some cases, a misconfiguration could allow Cross-Site Scripting. In ASP.NET, if a page's EnableViewStateMac property is set to False, the ASP.NET view state can be used as a vector for Cross-Site Scripting.

An IDS or IPS can also be used to detect or filter out XSS attacks. Below are a few regular expressions that will help detect Cross-Site Scripting.

Regex for a simple XSS attack:
/((\%3C) <)((\%2F) \/)*[a-z0-9\%]+((\%3E) >)/ix

The above regular expression would be added into a new Snort rule as follows:

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"NII Cross-Site Scripting attempt"; flow:to_server,established; pcre:"/((\%3C) <)((\%2F) \/)*[a-z0-9\%]+((\%3E) >)/i"; classtype:Web-application-attack; sid:9000; rev:5;)

Paranoid regex for XSS attacks:
/((\%3C) <)[^\n]+((\%3E) >)/I

This signature simply looks for the opening HTML tag, and its hex equivalent, followed by one or more characters other than the new line, and then followed by the closing tag or its hex equivalent. This may end up giving a few false positives depending upon how your web application and web server are structured, but it is guaranteed to catch anything that even remotely resembles a Cross-Site Scripting attack.

For QA:

Fixes for Cross-Site Scripting defects will ultimately require code based fixes.

Reference

OWASP Cross-Site Scripting Information:
https://www.owasp.org/index.php/XSS

Microsoft:
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q252985

Microsoft Anti-Cross Site Scripting Library
https://msdn.microsoft.com/en-us/security/aa973814.aspx

CERT:
http://www.cert.org/advisories/CA-2000-02.html

Apache:
http://httpd.apache.org/info/css-security/apache_specific.html

SecurityFocus.com:
http://www.securityfocus.com/infocus/1768

Classifications

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
http://cwe.mitre.org/data/definitions/79.html

CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)
http://cwe.mitre.org/data/definitions/80.html

CWE-82: Improper Neutralization of Script in Attributes of IMG Tags in a Web Page

http://cwe.mitre.org/data/definitions/82.html

CWE-83: Improper Neutralization of Script in Attributes in a Web Page
http://cwe.mitre.org/data/definitions/83.html

CWE-87: Improper Neutralization of Alternate XSS Syntax
http://cwe.mitre.org/data/definitions/87.html

CWE-116: Improper Encoding or Escaping of Output
http://cwe.mitre.org/data/definitions/116.html

CWE-692: Incomplete Blacklist to Cross-Site Scripting
http://cwe.mitre.org/data/definitions/692.html

CWE-811: OWASP Top Ten 2010 Category A2 - Cross-Site Scripting (XSS)
http://cwe.mitre.org/data/definitions/811.html

Kingdom: Input Validation and Representation
https://vulncat.fortify.com/

Web Server Misconfiguration: Unprotected File (157)

Summary

The Apache server status page (server-status) was found. This page contains detailed information about the current use of your web server, including information about the current hosts and requests being processed. This vulnerability is caused by default or incorrect configuration of the httpd.conf file. If exploited, an attacker could view the sensitive system information in the file. Recommendations include editing the web server configuration file to prevent access to the server-status page.

Execution

To verify the exploit, click the following link:http://zero.webappsecurity.com:80/server-status

Implication

A basic requirement for a successful attack upon your web application is reconnaissance. An attacker will employ a variety of methods, including malicious scanning agents and Google searches, to find out as much information about your web application as possible. The attacker can then use that information to formulate the next method of attack. An attacker who discovers sensitive system information has had a large portion of reconnaissance conducted for him or her.

Fix

For Security Operations:

For security reasons, you should restrict access to the server-status page in your web server configuration. To do this, comment out the following lines in the httpd.conf file:

<Location /server-status>
SetHandler server-status
</Location>

For Development:

Unless you are actively involved with implementing the web application server, there is not a wide range of available solutions to prevent problems that can occur from an attacker discovering sensitive system information about your application. Primarily, this problem will be resolved by the web application server administrator or security operations. However, there are certain actions you can take that will help to secure your web application and make it harder for an attacker to conduct a

successful attack.

- Ensure that files containing sensitive information are not left publicly accessible, or that comments left inside files do not reveal the locations of directories best left confidential.
- Do not reveal information in pathnames that are publicly displayed. Do not include drive letters or directories outside of the web document root in the pathname when a file must call another file on the web server. Use pathnames that are relative to the current directory or the webroot.
- Do not display error messages to the end user that provide information, such as directory names, that could be used in orchestrating an attack.
- Restrict access to important files or directories only to those who actually need it.

For QA:

This assessment performs the rote tasks of determining the directories and contents that are available via your web application. For reasons of security, it is important to test the web application not only from the perspective of a normal user, but also from that of a malicious one. Whenever possible, adopt the mindset of an attacker when testing your web application for security defects. Access your web application from outside your firewall or IDS. Use Google or another search engine to ensure that searches for vulnerable files or directories do not return information regarding your web application. For example, an attacker will use a search engine, and search for directory listings such as 'index of / cgi-bin'. Make sure that your directory structure is not obvious, and that only files that are necessary are capable of being accessed.

Reference

Apache Documentation
Configuration Files
Apache Module mod_status

Classifications

CWE-200: Information Exposure
http://cwe.mitre.org/data/definitions/200.html

Kingdom: Environment
https://vulncat.fortify.com/

Web Server Misconfiguration: Unprotected File (708)

Summary

Webinspect has detected a backup file with the .bak extension on the target server. The severity of the threats posed by the web-accessible backup files depends on the sensitivity of the information stored in original document. Based on that information, the attacker can gain sensitive information about the site architecture, database and network access credential details, encryption keys, and so forth from these files. The attacker can use information obtained to craft precise targeted attacks, which may not otherwise be feasible, against the application.

Execution

Browse to http://zero.webappsecurity.com:80/faq.html.bak and inspect the content. Response should be a return with HTTP status code 200 and should not match target site's file not found response.

Implication

An attacker can use the information obtained from the backup file of a sensitive document to craft a precise targeted attack against the web application. Such attacks can include, but are not limited to, SQL injection, remote file system access to

overwrite or inject malware, and database manipulation.

Fix

- Webroot Security Policy: Implement a security policy that prohibits storage of backup files in webroot.
- Temporary Files: Many tools and editors automatically create temporary files or backup files in the webroot. Be careful when editing files on a production server to avoid inadvertently leaving a backup or temporary copy of the file(s) in the webroot.
- Default Installations: Often, a lot of unnecessary files and folders are installed by default. For instance, IIS installations include demo applications. Be sure to remove any files or folders that are not required for application to work properly.
- Development Backup: Source code back up should not be stored and left available on the webroot.

Further QA can include test cases to look for the presence of backup files in the webroot to ensure none are left in publicly accessible folders of the web application.

Reference

OWASP - Review Old, Backup and Unreferenced Files for Sensitive Information (OTG-CONFIG-004)
CWE - 200 Information Exposure

Classifications

CWE-200: Information Exposure
http://cwe.mitre.org/data/definitions/200.html

Kingdom: Environment
https://vulncat.fortify.com/

Web Server Misconfiguration: Unprotected File (709)

Summary

Webinspect has detected a backup file with the .old extension on the target server. The severity of the threats posed by the web-accessible backup files depends on the sensitivity of the information stored in original document. Based on that information, the attacker can gain sensitive information about the site architecture, database and network access credential details, encryption keys, and so forth from these files. The attacker can use information obtained to craft precise targeted attacks, which may not otherwise be feasible, against the application.

Execution

Browse to http://zero.webappsecurity.com:80/index.html.old and inspect the content. Response should be a return with HTTP status code 200 and should not match target site's file not found response.

Implication

An attacker can use the information obtained from the backup file of a sensitive document to craft a precise targeted attack against the web application. Such attacks can include, but are not limited to, SQL injection, remote file system access to overwrite or inject malware, and database manipulation.

Fix

- Webroot Security Policy: Implement a security policy that prohibits storage of backup files in webroot.
- Temporary Files: Many tools and editors automatically create temporary files or backup files in the webroot. Be careful when editing files on a production server to avoid inadvertently leaving a backup or temporary copy of the file(s) in the

webroot.

- Default Installations: Often, a lot of unnecessary files and folders are installed by default. For instance, IIS installations include demo applications. Be sure to remove any files or folders that are not required for application to work properly.
- Development Backup: Source code back up should not be stored and left available on the webroot.

Further QA can include test cases to look for the presence of backup files in the webroot to ensure none are left in publicly accessible folders of the web application.

Reference

OWASP - Review Old, Backup and Unreferenced Files for Sensitive Information (OTG-CONFIG-004)
CWE - 200 Information Exposure

Classifications

CWE-200: Information Exposure
http://cwe.mitre.org/data/definitions/200.html

Kingdom: Environment
https://vulncat.fortify.com/

Web Server Misconfiguration: Unprotected File (1368)

Summary

The file debug.txt was located. This type of file is usually left by a developer or web master to test a certain function of the web application or web server. Leaving test scripts available on the server is a very unsecure practice. The types of information that can be gleaned from test scripts include fixed authentication session id's, usernames and passwords, locations or pointers to confidential areas of the web site, and proprietary source code. With this type of information available to an attacker , they can either use it to totally breach the security of the site or use it as a stepping stone to retrieve other sensitive data. Recommendations include removing this file from the production server.

Fix

For Security Operations:
Remove the application from the server. Inform developers and administrators to remove test applications from servers when they are no longer needed. While they are in use, be sure to protect them using HTTP basic authentication.

For Development:

Contact your security or network operations team and request they investigate the issue.

For QA:
Contact your security or network operations team and request they investigate the issue.

Reference

Classifications

CWE-284: Access Control (Authorization) Issues
http://cwe.mitre.org/data/definitions/284.html

CWE-200: Information Exposure

http://cwe.mitre.org/data/definitions/200.html

Kingdom: Environment
https://vulncat.fortify.com/

Web Server Misconfiguration: Unprotected File (2083)

Summary

Webinspect has detected a backup file by replacing the extension with .old on the target server. The severity of the threats posed by the web-accessible backup files depends on the sensitivity of the information stored in original document. Based on that information, the attacker can gain sensitive information about the site architecture, database and network access credential details, encryption keys, and so forth from these files. The attacker can use information obtained to craft precise targeted attacks, which may not otherwise be feasible, against the application.

Execution

Browse to http://zero.webappsecurity.com:80/index.old and inspect the content. Response should be a return with HTTP status code 200 and should not match target site's file not found response.

Implication

An attacker can use the information obtained from the backup file of a sensitive document to craft a precise targeted attack against the web application. Such attacks can include, but are not limited to, SQL injection, remote file system access to overwrite or inject malware, and database manipulation.

Fix

- Webroot Security Policy: Implement a security policy that prohibits storage of backup files in webroot.
- Temporary Files: Many tools and editors automatically create temporary files or backup files in the webroot. Be careful when editing files on a production server to avoid inadvertently leaving a backup or temporary copy of the file(s) in the webroot.
- Default Installations: Often, a lot of unnecessary files and folders are installed by default. For instance, IIS installations include demo applications. Be sure to remove any files or folders that are not required for application to work properly.
- Development Backup: Source code back up should not be stored and left available on the webroot.

Further QA can include test cases to look for the presence of backup files in the webroot to ensure none are left in publicly accessible folders of the web application.

Reference

OWASP - Review Old, Backup and Unreferenced Files for Sensitive Information (OTG-CONFIG-004)
CWE - 200 Information Exposure

Classifications

CWE-200: Information Exposure
http://cwe.mitre.org/data/definitions/200.html

Kingdom: Environment
https://vulncat.fortify.com/

Insecure Transport (4722)

Summary

Any area of a web application that possibly contains sensitive information or access to privileged functionality such as remote site administration functionality should utilize SSL or another form of encryption to prevent login information from being sniffed or otherwise intercepted or stolen. http://zero.webappsecurity.com:80/forgot-password.html has failed this policy. Recommendations include ensuring that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

Implication

An attacker who exploited this design vulnerability would be able to utilize the information to escalate their method of attack, possibly leading to impersonation of a legitimate user, the theft of proprietary data, or execution of actions not intended by the application developers.

Fix

For Security Operations:
Ensure that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

For Development:
Ensure that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

For QA:
Test the application not only from the perspective of a normal user, but also from the perspective of a malicious one.

Reference

Classifications

CWE-287: Improper Authentication
http://cwe.mitre.org/data/definitions/287.html

Kingdom: Security Features
https://vulncat.fortify.com/

Often Misused: Login (10595)

Summary

An unencrypted login form has been discovered. Any area of a web application that possibly contains sensitive information or access to privileged functionality such as remote site administration functionality should utilize SSL or another form of encryption to prevent login information from being sniffed or otherwise intercepted or stolen. If the login form is being served over SSL, the page that the form is being submitted to MUST be accessed over SSL. Every link/URL present on that page (not just the form action) needs to be served over HTTPS. This will prevent Man-in-the-Middle attacks on the login form. Recommendations include ensuring that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

Implication

An attacker who exploited this design vulnerability would be able to utilize the information to escalate their method of attack, possibly leading to impersonation of a legitimate user, the theft of proprietary data, or execution of actions not intended by the application developers.

Fix

Ensure that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

Reference

Advisory:http://www.kb.cert.org/vuls/id/466433

Classifications

CWE-287: Improper Authentication
http://cwe.mitre.org/data/definitions/287.html

Kingdom: API Abuse
https://vulncat.fortify.com/

CWE-311: Missing Encryption of Sensitive Data
http://cwe.mitre.org/data/definitions/311.html

Cross-Frame Scripting (11293)

Summary

A Cross-Frame Scripting (XFS) vulnerability can allow an attacker to load the vulnerable application inside an HTML iframe tag on a malicious page. The attacker could use this weakness to devise a Clickjacking attack to conduct phishing, frame sniffing, social engineering or Cross-Site Request Forgery attacks.

Clickjacking
The goal of a Clickjacking attack is to deceive the victim (user) into interacting with UI elements of the attacker's choice on the target web site without their knowledge and then executing privileged functionality on the victim's behalf. To achieve this goal, the attacker must exploit the XFS vulnerability to load the attack target inside an iframe tag, hide it using Cascading Style Sheets (CSS) and overlay the phishing content on the malicious page. By placing the UI elements on the phishing page so they overlap with those on the page targeted in the attack, the attacker can ensure that the victim must interact with the UI elements on the target page not visible to the victim.

WebInspect has detected a page which potentially handles sensitive information using an HTML form with a password input field and is missing XFS protection.

This response is not protected by a valid X-Frame-Options header and a valid Content-Security-Policy(CSP) frame-ancestors directive header.Furthermore,
An effective frame-busting technique was not observed while loading this page inside a frame.

Execution

Create a test page containing an HTML iframe tag whose src attribute is set to http://zero.webappsecurity.com:80/login.html. Successful framing of the target page indicates that the application is susceptibile to XFS.

Note that WebInspect will report only one instance of this check across each host within the scope of the scan. The other visible pages on the site may, however, be vulnerable to XFS as well and therefore should be protected against it with an appropriate fix.

Implication

A Cross-Frame Scripting weakness could allow an attacker to embed the vulnerable application inside an iframe. Exploitation of this weakness could result in:

Hijacking of user events such as keystrokes
Theft of sensitive information
Execution of privileged functionality through combination with Cross-Site Request Forgery attacks

Fix

The Content Security Policy (CSP) frame-ancestors directive obsoletes the X-Frame-Options header. Both provide for a policy-based mitigation technique against cross-frame scripting vulnerabilities. The difference is that while the X-Frame-Options technique only checks against the top-level document's location, the CSP frame-ancestors header checks for conformity from all ancestors.

If both CSP frame-ancestors and X-Frame-Options headers are present and supported, the CSP directive will prevail. WebInspect recommends using both CSP frame-ancestors and X-Frame-Options headers as CSP is not supported by Internet Explorer and many older versions of other browsers.

In addition, developers must also use client-side frame busting JavaScript as a protection against XFS. This will enable users of older browsers that do not support the X-Frame-Options header to also be protected from Clickjacking attacks.

X-Frame-Options
Developers can use this header to instruct the browser about appropriate actions to perform if their site is included inside an iframe.Developers must set the X-Frame-Options header to one of the following permitted values:

- DENY

Deny all attempts to frame the page

- SAMEORIGIN

The page can be framed by another page only if it belongs to the same origin as the page being framed

- ALLOW-FROM origin

Developers can specify a list of trusted origins in the origin attribute. Only pages on origin are permitted to load this page inside an iframe

Content-Security-Policy: frame-ancestors

Developers can use the CSP header with the frame-ancestors directive, which replaces the X-Frame-Options header, to instruct the browser about appropriate actions to perform if their site is included inside an iframe. Developers can set the frame-ancestors attribute to one of the following permitted values:

- 'none'

Equivalent to "DENY" - deny all attempts to frame the page

- 'self'

Equivalent to "SAMEORIGIN" - the page can be framed by another page only if it belongs to the same origin as the page being framed

- <host-source>

Equivalent to "ALLOW-FROM" - developers can specify a list of trusted origins which maybe host name or IP address or URL scheme. Only pages on this list of trusted origin are permitted to load this page inside an iframe

- <scheme-source>

Developers can also specify a schema such as http: or https: that can frame the page.

Reference

Frame Busting:
Busting Frame Busting: A Study of Clickjacking Vulnerabilities on Popular Sites
OWASP: Busting Frame Busting

OWASP:
Clickjacking

Content-Security-Policy (CSP)
CSP: frame-ancestors

Specification:
Content Security Policy Level 2
X-Frame-Options IETF Draft

Server Configuration:
IIS
Apache, nginx

HP 2012 Cyber Security Report
The X-Frame-Options header - a failure to launch

Classifications

CWE-352: Cross-Site Request Forgery (CSRF)
http://cwe.mitre.org/data/definitions/352.html

Kingdom: Security Features
https://vulncat.fortify.com/

Expression Language Injection (11310)

Summary

WebInspect has detected an Expression Language (EL) injection vulnerability. EL injection vulnerabilities are introduced when an application fails to sufficiently validate untrusted user data before assigning it to attribute values of certain Spring MVC JSP tags.

Expression Language allows JSP pages to easily access application data stored in user-defined JavaBeans components as well the implicit objects. In addition, JSP pages can also invoke arbitrary public and static methods and perform arithmetic operations using EL expressions.

By allowing attackers to inject EL expressions through insufficiently validated user input, an application could grant unauthorized access to sensitive application and server information. Expression Language injection could also let attackers bypass HTTPOnly access restrictions imposed on cookies by exploiting access to the implicit ***cookie***object made available in EL expressions.

The affected spring framework versions include

- 3.0.0 to 3.0.5
- 2.5.0 to 2.5.6.SEC02 (community releases)
- 2.5.0 to 2.5.7.SR01 (subscription customers)

Execution

Click http://zero.webappsecurity.com:80/search.html?searchTerm=${7166%2b2327} to verify the vulnerability in a web browser.

Implication

Expression Language injection vulnerabilities can be used to steal sensitive application information as well as bypass HTTPOnly cookie access restrictions. The impact depends on the information available within the application's context.

Fix

The vulnerability can be fixed by upgrading to Spring framework versions 3.1 and above.

For versions below 3.1 (3.0.6 onwards, 2.5.6.SEC03 onwards and 2.5.7.SR02 onwards), set the value of ***springJspExpressionSupport***context parameter to ***false***.

Reference

Vendor:
SpringSource

Advisory:
Expression Language Injection

CVE:
CVE-2011-2730

Expression Language:
Expression Language Specification

Classifications

CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
http://cwe.mitre.org/data/definitions/78.html

Kingdom: Input Validation and Representation
https://vulncat.fortify.com/

Expression Language Injection (11319)

Summary

WebInspect has detected a Expression Language (EL) injection vulnerability leading to HTTPOnly restriction bypass. EL injection vulnerabilities are introduced when an application fails to sufficiently validate untrusted user data before assigning it to attribute values of certain Spring MVC JSP tags.

Expression Language allows JSP pages to easily access application data stored in user-defined JavaBeans components as well the implicit objects. In addition, JSP pages can also invoke arbitrary public and static methods and perform arithmetic operations using EL expressions.

By allowing attackers to inject EL expressions through insufficiently validated user input, an application could grant unauthorized access to sensitive application and server information. Expression Language injection could also let attackers bypass HTTPOnly access restrictions imposed on cookies by exploiting access to the implicit ***cookie***object made available in EL expressions.

The affected spring framework versions include

- 3.0.0 to 3.0.5
- 2.5.0 to 2.5.6.SEC02 (community releases)
- 2.5.0 to 2.5.7.SR01 (subscription customers)

Execution

Click http://zero.webappsecurity.com:80/search.html?searchTerm=${cookie%5B%22JSESSIONID%22%5D%2Evalue} to verify the vulnerability in a web browser.

Implication

Using Expression Language injection attackers can bypass HTTPOnly restrictions on cookies and steal sensitive information using EL expressions.

Fix

The vulnerability can be fixed by upgrading to Spring framework versions 3.1 and above.

For versions below 3.1 (3.0.6 onwards, 2.5.6.SEC03 onwards and 2.5.7.SR02 onwards), set the value of ***springJspExpressionSupport***context parameter to ***false***.

Reference

Vendor:
SpringSource

Advisory:
Expression Language Injection

CVE:
CVE-2011-2730

Expression Language:
Expression Language Specification

Classifications

CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
http://cwe.mitre.org/data/definitions/78.html

Kingdom: Input Validation and Representation
https://vulncat.fortify.com/

Scan Name:	Site: http://zero.webappsecurity.com/		
Policy:	Standard	**Crawl Sessions:**	239
Scan Date:	12/30/2019 3:57:58 PM	**Vulnerabilities:**	103
Scan Version:	19.2.0.184	**Scan Duration:**	9 minutes : 58 seconds
Scan Type:	Site	**Client:**	FF

Server: http://zero.webappsecurity.com:80

Critical Issues

Poor Error Handling: Unhandled Exception (742) View Description

CWE: 388,497,200

Kingdom: Errors

Page: http://zero.webappsecurity.com:80/account/

Cross-Site Scripting: Reflected (5649) View Description

CWE: 79,80,82,83,87,116,692,811

Kingdom: Input Validation and Representation

Page: http://zero.webappsecurity.com:80/faq.html?question=1%3c%73%43%72%49%70%54%3e%61%6c%65%72%74%28%38%35%37%36%35%29%3c%2f%73%43%72%49%70%54%3e

---

Page: http://zero.webappsecurity.com:80/search.html?searchTerm=12345%3c%73%43%72%3c%53%63%52%69%50%74%3e%49%70%54%3e%61%6c%65%72%74%28%34%37%32%37%34%29%3c%2f%73%43%72%3c%53%63%52%69%50%74%3e%49%70%54%3e

---

Cross-Site Scripting: Reflected (5650) View Description

CWE: 79,80,82,83,87,116,692,811

Kingdom: Input Validation and Representation

Page: http://zero.webappsecurity.com:80/sendFeedback.html

Privacy Violation: Social Security Number (10834) View Description

CWE: 359,200

Kingdom: Security Features

Page: http://zero.webappsecurity.com:80/admin/users.html

High Issues

Web Server Misconfiguration: Unprotected File (157) View Description

CWE: 200

Kingdom: Environment

Page: http://zero.webappsecurity.com:80/server-status

Web Server Misconfiguration: Unprotected File (708) View Description

CWE: 200

Kingdom: Environment

Page: http://zero.webappsecurity.com:80/faq.html.bak

Web Server Misconfiguration: Unprotected File (709) View Description

CWE: 200

Kingdom: Environment

Page: http://zero.webappsecurity.com:80/index.html.old

Web Server Misconfiguration: Unprotected File (1368) View Description

CWE: 284,200

Kingdom: Environment

Page: http://zero.webappsecurity.com:80/debug.txt

Web Server Misconfiguration: Unprotected File (2083) View Description

CWE: 200

Kingdom: Environment

Page: http://zero.webappsecurity.com:80/index.old

Insecure Transport (4722) View Description

CWE: 287

Kingdom: Security Features

Page: http://zero.webappsecurity.com:80/forgot-password.html

Page: http://zero.webappsecurity.com:80/login.html

Often Misused: Login (10595) View Description

CWE: 287,311

Kingdom: API Abuse

Page: http://zero.webappsecurity.com:80/login.html

Page: http://zero.webappsecurity.com:80/forgot-password.html

Cross-Frame Scripting (11293) View Description

CWE: 352

Kingdom: Security Features

Page: http://zero.webappsecurity.com:80/login.html

Expression Language Injection (11310) View Description

CWE: 78

Kingdom: Input Validation and Representation

Page: http://zero.webappsecurity.com:80/search.html?searchTerm=${6228%2b1320}

Expression Language Injection (11319) View Description

CWE: 78

Kingdom: Input Validation and Representation

Page: http://zero.webappsecurity.com:80/search.html?searchTerm=${cookie%5B%22JSESSIONID%22%5D%2Evalue}

Poor Error Handling: Unhandled Exception (742)

Summary

Critical database server error message vulnerabilities were identified in the web application, indicating that an unhandled exception was generated in your web application code. Unhandled exceptions are circumstances in which the application has received user input that it did not expect and does not know how to handle. When successfully exploited, an attacker can gain unauthorized access to the database by using the information recovered from seemingly innocuous error messages to pinpoint flaws in the web application and to discover additional avenues of attack. Recommendations include designing and adding consistent error-handling mechanisms that are capable of handling any user input to your web application, providing meaningful detail to end-users, and preventing error messages that might provide information useful to an attacker from being displayed.

Description

The most common cause of an unhandled exception is a failure to properly sanitize client-supplied data that is used in SQL statements. They can also be caused by a bug in the web application's database communication code, a misconfiguration of database connection settings, an unavailable database, or any other reason that would cause the application's database driver to be unable to establish a working session with the server. The problem is not that web applications generate errors. All web applications in their normal course of operation will at some point receive an unhandled exception. The problem lies not in that these errors were received, but rather in how they are handled. Any error handling solution needs to be well-designed, and uniform in how it handles errors. For instance, assume an attacker is attempting to access a specific file. If the request returns an error File not Found, the attacker can be relatively sure the file does not exist. However, if the error returns "Permission Denied," the attacker has a fairly good idea that the specific file does exist. This can be helpful to an attacker in many ways, from determining the operating system to discovering the underlying architecture and design of the application.

The error message may also contain the location of the file that contains the offending function. This may disclose the webroot's absolute path as well as give the attacker the location of application "include" files or database configuration information. A fundamental necessity for a successful attack upon your web application is reconnaissance. Database server error messages can provide information that can then be utilized when the attacker is formulating his next method of attack. It may even disclose the portion of code that failed.

Be aware that this check is part of unknown application testing which seeks to uncover new vulnerabilities in both custom and commercial software. Because of this, there are no specific patches or remediation information for this issue. Please note that this vulnerability may be a false positive if the page it is flagged on is technical documentation relating to a database server.

Execution

The ways in which an attacker can exploit the conditions that caused the error depend on its cause. In the case of SQL injection, the techniques that are used will vary from database server to database server, and even query to query. An in-depth guide to SQL Injection attacks is available at http://download.hpsmartupdate.com/asclabs/sql_injection.pdf, or in the SQL Injection vulnerability information, accessible via the Policy Manager. Primarily, the information gleaned from database server error messages is what will allow an attacker to conduct a successful attack after he combines his various findings.

Implication

The severity of this vulnerability depends on the reason that the error message was generated. In most cases, it will be the result of the web application attempting to use an invalid client-supplied argument in a SQL statement, which means that SQL injection will be possible. If so, an attacker will at least be able to read the contents of the entire database arbitrarily. Depending on the database server and the SQL statement, deleting, updating and adding records and executing arbitrary commands may also be possible. If a software bug or bug is responsible for triggering the error, the potential impact will vary, depending on the circumstances. The location of the application that caused the error can be useful in facilitating other kinds of attacks. If the file is a hidden or include file, the attacker may be able to gain more information about the mechanics of the web application, possibly even the source code. Application source code is likely to contain usernames, passwords, database connection strings and aids the attacker greatly in discovering new vulnerabilities.

Fix

For Development:

From a development perspective, the best method of preventing problems from arising from database error messages is to adopt secure programming techniques that prevent problems that might arise from an attacker discovering too much information about the architecture and design of your web application. The following recommendations can be used as a basis for that.

- Stringently define the data type (for instance, a string, an alphanumeric character, etc) that the application will accept.
- Use what is good instead of what is bad. Validate input for improper characters.

- Do not display error messages to the end user that provide information (such as table names) that could be utilized in orchestrating an attack.
- Define the allowed set of characters. For instance, if a field is to receive a number, only let that field accept numbers.
- Define the maximum and minimum data lengths for what the application will accept.
- Specify acceptable numeric ranges for input.

For Security Operations:

The following recommendations will help in implementing a secure database protocol for your web application. Be advised each database has its own method of secure lock down.

- **ODBC Error Messaging:** Turn off ODBC error messaging in your database server. Never display raw ODBC or other errors to the end user. See Removing Detailed Error Messages below, or consult your database server's documentation, for more information.
- **Uniform Error Codes:** Ensure that you are not inadvertently supplying information to an attacker via the use of inconsistent or "conflicting" error messages. For instance, don't reveal unintended information by utilizing error messages such as Access Denied, which will also let an attacker know that the file he seeks actually exists. Have consistent terminology for files and folders that do exist, do not exist, and which have read access denied.
- **Informational Error Messages:** Ensure that error messages do not reveal too much information. Complete or partial paths, variable and file names, row and column names in tables, and specific database errors should never be revealed to the end user. Remember, an attacker will gather as much information as possible, and then add pieces of seemingly innocuous information together to craft a method of attack.
- **Proper Error Handling:** Utilize generic error pages and error handling logic to inform end users of potential problems. Do not provide system information or other data that could be utilized by an attacker when orchestrating an attack.
- **Stored Procedures:** Consider using stored procedures. They require a very specific parameter format, which makes them less susceptible to SQL Injection attacks.
- **Database Privileges:** Utilize a least-privileges scheme for the database application. Ensure that user accounts only have the limited functionality that is actually required. All database mechanisms should deny access until it has been granted, not grant access until it has been denied.

For QA:

In reality, simple testing can usually determine how your web application will react to different input errors. More expansive testing must be conducted to cause internal errors to gauge the reaction of the site. If the unhandled exception occurs in a piece of in-house developed software, consult the developer. If it is in a commercial package, contact technical support.

The best course of action for QA associates to take is to ensure that the error handling scheme is consistent. Do you receive a different type of error for a file that does not exist as opposed to a file that does? Are phrases like "Permission Denied" utilized which could reveal the existence of a file to an attacker?

Reference

Apache:
Apache HTTP Server Version 1.3 Custom Error Responses
Apache HTTP Server Version 2.0 Custom Error Responses

Microsoft:
Description of Microsoft Internet Information Services (IIS) 5.0 and 6.0 status codes

Classifications

CWE-388: Error Handling

http://cwe.mitre.org/data/definitions/388.html

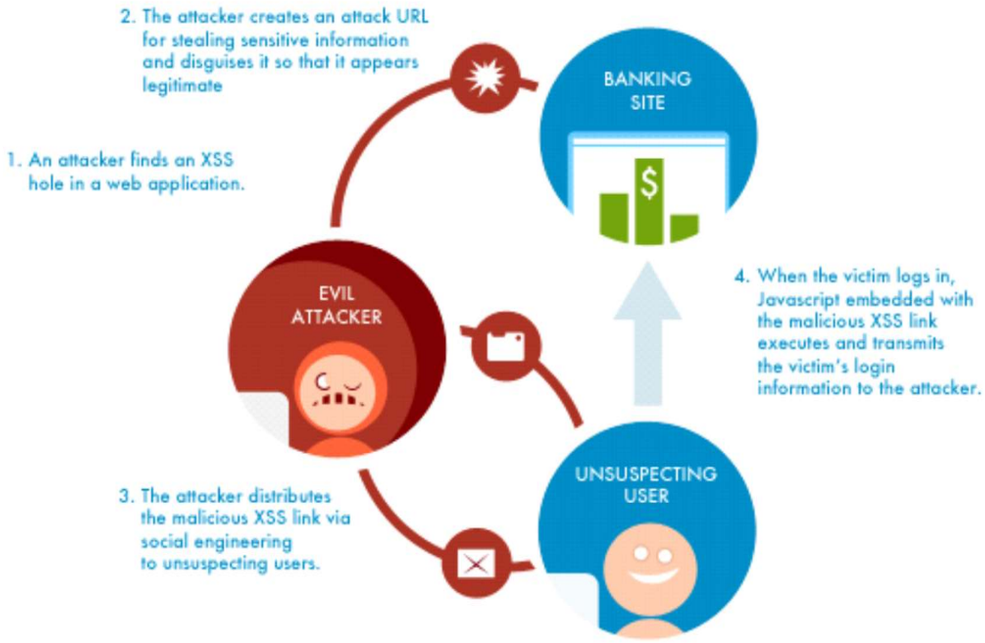
CWE-497: Exposure of System Data to an Unauthorized Control Sphere
http://cwe.mitre.org/data/definitions/497.html

CWE-200: Information Exposure
http://cwe.mitre.org/data/definitions/200.html

Kingdom: Errors
https://vulncat.fortify.com/

Cross-Site Scripting: Reflected (5649)

Summary



Cross-Site Scripting vulnerability found in Get parameter question. The following attack uses plain encoding:

<sCrIpT>alert(85765)</sCrIpT>

A Cross-Site Scripting (XSS) vulnerability was detected in the web application. Cross-Site Scripting occurs when dynamically generated web pages display user input, such as login information, that is not properly validated, allowing an attacker to embed malicious scripts into the generated page and then execute the script on the machine of any user that views the site. In this instance, the web application was vulnerable to an automatic payload, meaning the user simply has to visit a page to

make the malicious scripts execute. If successful, Cross-Site Scripting vulnerabilities can be exploited to manipulate or steal cookies, create requests that can be mistaken for those of a valid user, compromise confidential information, or execute malicious code on end user systems. Recommendations include implementing secure programming techniques that ensure proper filtration of user-supplied data, and encoding all user supplied data to prevent inserted scripts being sent to end users in a format that can be executed.

Execution

View the attack string included with the request to check what to search for in the response. For instance, if "(javascript:alert ('XSS')" is submitted as an attack (or another scripting language), it will also appear as part of the response. This indicates that the web application is taking values from the HTTP request parameters and using them in the HTTP response without first removing potentially malicious data.The response can be viewed in "Web Browser" view in the Vulnerability pane to see the injected popup events in action. Events requiring user interaction (e.g. onmouseover or onclick events) can be triggered by performing the corresponding action (e.g. clicking the injected link). Injection with numeric string in src, or href, attributes indicates that the site is vulnerable to script include or content exfiltration. These can be verified by repeating the request in a browser and intercepting originating network traffic in a web proxy.

Implication

Cross-Site Scripting(XSS) happens when user input from a web client is immediately included via server-side scripts in a dynamically generated web page. Reflected XSS is specifically considered critical when malicious payload can be embedded in a URL (e.g. in query strings of GET requests). An attacker can trick a victim, via phishing attack, to click on a link with vulnerable input which has been altered to include attack code and then sent to the legitimate server. The injected code is then reflected back to the user's browser which executes it.

The implications of successful Cross-Site Scripting attacks are:

- Account hijacking - An attacker can hijack the user's session before the session cookie expires and take actions with the privileges of the user who accessed the URL, such as issuing database queries and viewing the results.
- Malicious script execution - Users can unknowingly execute JavaScript, VBScript, ActiveX, HTML, or even Flash content that has been inserted into a dynamically generated page by an attacker.
- Worm propagation - With Ajax applications, XSS can propagate somewhat like a virus. The XSS payload can autonomously inject itself into pages, and easily re-inject the same host with more XSS, all of which can be done with no hard refresh. Thus, XSS can send multiple requests using complex HTTP methods to propagate itself invisibly to the user.
- Information theft - Via redirection and fake sites, attackers can connect users to a malicious server of the attacker's choice and capture any information entered by the user.
- Denial of Service - Often by utilizing malformed display requests on sites that contain a Cross-Site Scripting vulnerability, attackers can cause a denial of service condition to occur by causing the host site to query itself repeatedly .
- Browser Redirection - On certain types of sites that use frames, a user can be made to think that he is in fact on the original site when he has been redirected to a malicious one, since the URL in the browser's address bar will remains the same. This is because the entire page isn't being redirected, just the frame in which the JavaScript is being executed is redirected.
- Manipulation of user settings - Attackers can change user settings for nefarious purposes.
- Bypass Content-Security-Policy protection - Attackers can inject a malformed tag formation, known as dangling tag injection, which in some cases allows injected script to reuse valid nonce on the page and bypass script source restriction. Additionally dangling tag injection can be used to steal sensitive information embedded in HTML response if browser is able to make a request to the injected link.
- Base tag injection: Attacker can cause relative links on a page to load from a different domain by modifying the base URL for the page via base tag injection.
- Link prefetch injection: While unable to execute script, attackers can use link tag with rel=prefetch that will make browsers pre-fetch the specified link even though it is never rendered and rejected subsequently due to web application enforced cross-site policy (e.g. CSP protections).
- Edge side includes (ESI) Injection - ESI is a markup language used in various HTTP devices, such as reverse proxies and load balancers, that are positioned between client and server. An attacker can inject ESI markup to perform critical attacks such as cross-site scripting and HTTPOnly cookie protection bypass.

Fix

For Development:

Cross-Site Scripting attacks can be avoided by carefully validating all input, and properly encoding all output. When validating user input, verify that it matches the strictest definition of valid input possible. For example, if a certain parameter is supposed to be a number, attempt to convert it to a numeric data type in your programming language.

PHP: intval("0".$_GET['q']);

ASP.NET: int.TryParse(Request.QueryString["q"], out val);

The same applies to date and time values, or anything that can be converted to a stricter type before being used. When accepting other types of text input, make sure the value matches either a list of acceptable values (white-listing), or a strict

regular expression. If at any point the value appears invalid, do not accept it. Also, do not attempt to return the value to the user in an error message.

Most server side scripting languages provide built in methods to convert the value of the input variable into correct, non-interpretable HTML. These should be used to sanitize all input before it is displayed to the client.

PHP: string htmlspecialchars (string string [, int quote_style])

ASP.NET: Server.HTMLEncode (strHTML String)

When reflecting values into JavaScript or another format, make sure to use a type of encoding that is appropriate. Encoding data for HTML is not sufficient when it is reflected inside of a script or style sheet. For example, when reflecting data in a JavaScript string, make sure to encode all non-alphanumeric characters using hex (\xHH) encoding.

If you have JavaScript on your page that accesses unsafe information (like location.href) and writes it to the page (either with document.write, or by modifying a DOM element), make sure you encode data for HTML before writing it to the page. JavaScript does not have a built-in function to do this, but many frameworks do. If you are lacking an available function, something like the following will handle most cases:

s = s.replace(/&/g,'&').replace(/"/i,'"').replace(/</i,'<').replace(/>/i,'>').replace(/'/i,''')

Ensure that you are always using the right approach at the right time. Validating user input should be done as soon as it is received. Encoding data for display should be done immediately before displaying it.

For Security Operations:

Server-side encoding, where all dynamic content is first sent through an encoding function where Scripting tags will be replaced with codes in the selected character set, can help to prevent Cross-Site Scripting attacks.

Many web application platforms and frameworks have some built-in support for preventing Cross-Site Scripting. Make sure that any built-in protection is enabled for your platform. In some cases, a misconfiguration could allow Cross-Site Scripting. In ASP.NET, if a page's EnableViewStateMac property is set to False, the ASP.NET view state can be used as a vector for Cross-Site Scripting.

An IDS or IPS can also be used to detect or filter out XSS attacks. Below are a few regular expressions that will help detect Cross-Site Scripting.

Regex for a simple XSS attack:
/((\%3C) <)((\%2F) \/)*[a-z0-9\%]+((\%3E) >)/ix

The above regular expression would be added into a new Snort rule as follows:

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"NII Cross-Site Scripting attempt"; flow:to_server,established; pcre:"/((\%3C) <)((\%2F) \/)*[a-z0-9\%]+((\%3E) >)/i"; classtype:Web-application-attack; sid:9000; rev:5;)

Paranoid regex for XSS attacks:
/((\%3C) <)[^\n]+((\%3E) >)/I

This signature simply looks for the opening HTML tag, and its hex equivalent, followed by one or more characters other than the new line, and then followed by the closing tag or its hex equivalent. This may end up giving a few false positives depending upon how your web application and web server are structured, but it is guaranteed to catch anything that even remotely resembles a Cross-Site Scripting attack.

For QA:

Fixes for Cross-Site Scripting defects will ultimately require code based fixes. Read the the following links for more information about manually testing your application for Cross-Site Scripting.

Reference

OWASP Cross-Site Scripting Information
https://www.owasp.org/index.php/XSS

CERT
http://www.cert.org/advisories/CA-2000-02.html

Apache
http://httpd.apache.org/info/css-security/apache_specific.html

SecurityFocus.com
http://www.securityfocus.com/infocus/1768

Classifications

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
http://cwe.mitre.org/data/definitions/79.html

CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)
http://cwe.mitre.org/data/definitions/80.html

CWE-82: Improper Neutralization of Script in Attributes of IMG Tags in a Web Page
http://cwe.mitre.org/data/definitions/82.html

CWE-83: Improper Neutralization of Script in Attributes in a Web Page
http://cwe.mitre.org/data/definitions/83.html

CWE-87: Improper Neutralization of Alternate XSS Syntax
http://cwe.mitre.org/data/definitions/87.html

CWE-116: Improper Encoding or Escaping of Output
http://cwe.mitre.org/data/definitions/116.html

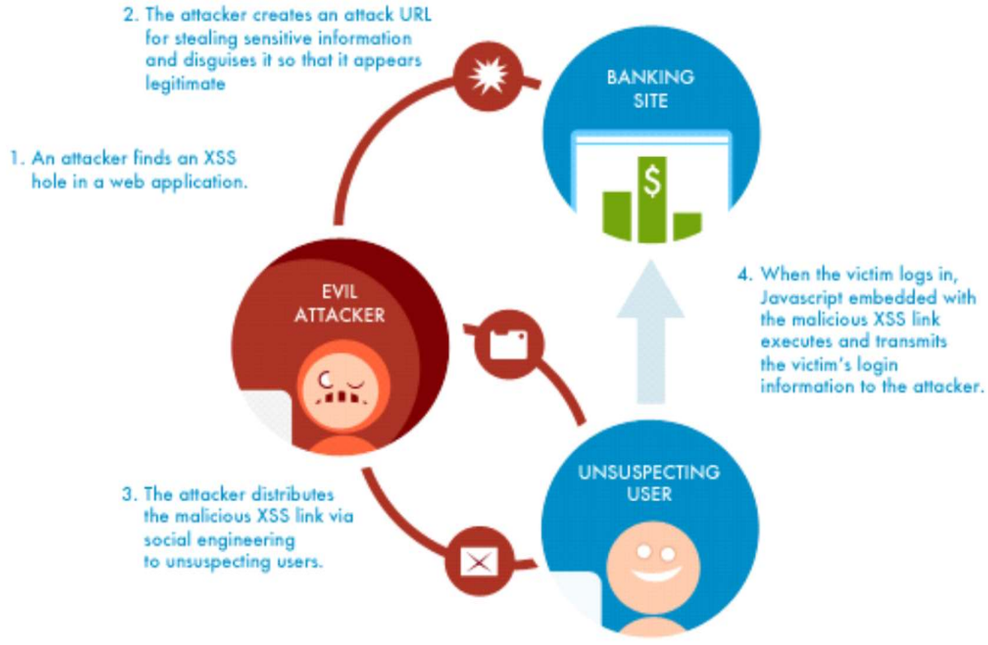
CWE-692: Incomplete Blacklist to Cross-Site Scripting
http://cwe.mitre.org/data/definitions/692.html

CWE-811: OWASP Top Ten 2010 Category A2 - Cross-Site Scripting (XSS)
http://cwe.mitre.org/data/definitions/811.html

Kingdom: Input Validation and Representation
https://vulncat.fortify.com/

Cross-Site Scripting: Reflected (5650)

Summary



Cross-Site Scripting vulnerability found in Post parameter name. Triggering this vulnerability requires user action. The following attack uses plain encoding:

<a HrEf=JaVaScRiPt:alert(71722)>

Cross-Site Scripting occurs when dynamically generated web pages display user input, such as login information, that is not properly validated, allowing an attacker to embed malicious scripts into the generated page and then execute the script on the machine of any user that views the site. User interaction vulnerabilities such as this one require the user to trigger the execution of the malicious scripts via an action such as clicking a link or moving the mouse pointer over text. If successful, Cross-Site Scripting vulnerabilities can be exploited to manipulate or steal cookies, create requests that can be mistaken for those of a valid user, compromise confidential information, or execute malicious code on end user systems. Recommendations include implementing secure programming techniques that ensure proper filtration of user-supplied data, and encoding all user supplied data to prevent inserted scripts being sent to end users in a format that can be executed.

Execution

View the attack string included with the request to check what to search for in the response. For instance, if "(javascript:alert ('XSS')" is submitted as an attack, it will also appear as part of the response. This indicates that the web application is taking values from the HTTP request parameters and using them in the HTTP response without first removing potentially malicious data.

Implication

XSS can generally be subdivided into two categories: stored and reflected attacks. The main difference between the two is in how the payload arrives at the server. Stored attacks are just that...in some form stored on the target server, such as in a database, or via a submission to a bulletin board or visitor log. The victim will retrieve and execute the attack code in his browser when a request is made for the stored information. Reflected attacks, on the other hand, come from somewhere else. This happens when user input from a web client is immediately included via server-side scripts in a dynamically generated web page. Via some social engineering, an attacker can trick a victim, such as through a malicious link or "rigged" form, to submit information which will be altered to include attack code and then sent to the legitimate server. The injected code is then reflected back to the user's browser which executes it because it came from a trusted server. The implication of each

kind of attack is the same.

The main problems associated with successful Cross-Site Scripting attacks are:

- Account hijacking - An attacker can hijack the user's session before the session cookie expires and take actions with the privileges of the user who accessed the URL, such as issuing database queries and viewing the results.
- Malicious script execution - Users can unknowingly execute JavaScript, VBScript, ActiveX, HTML, or even Flash content that has been inserted into a dynamically generated page by an attacker.
- Worm propagation - With Ajax applications, XSS can propagate somewhat like a virus. The XSS payload can autonomously inject itself into pages, and easily re-inject the same host with more XSS, all of which can be done with no hard refresh. Thus, XSS can send multiple requests using complex HTTP methods to propagate itself invisibly to the user.
- Information theft - Via redirection and fake sites, attackers can connect users to a malicious server of the attacker's choice and capture any information entered by the user.
- Denial of Service - Often by utilizing malformed display requests on sites that contain a Cross-Site Scripting vulnerability, attackers can cause a denial of service condition to occur by causing the host site to query itself repeatedly .
- Browser Redirection - On certain types of sites that use frames, a user can be made to think that he is in fact on the original site when he has been redirected to a malicious one, since the URL in the browser's address bar will remains the same. This is because the entire page isn't being redirected, just the frame in which the JavaScript is being executed.
- Manipulation of user settings - Attackers can change user settings for nefarious purposes.

Fix

For Development:

Cross-Site Scripting attacks can be avoided by carefully validating all input, and properly encoding all output. When validating user input, verify that it matches the strictest definition of valid input possible. For example, if a certain parameter is supposed to be a number, attempt to convert it to a numeric data type in your programming language.

PHP: intval("0".$_GET['q']);

ASP.NET: int.TryParse(Request.QueryString["q"], out val);

The same applies to date and time values, or anything that can be converted to a stricter type before being used. When accepting other types of text input, make sure the value matches either a list of acceptable values (white-listing), or a strict regular expression. If at any point the value appears invalid, do not accept it. Also, do not attempt to return the value to the user in an error message.

Most server side scripting languages provide built in methods to convert the value of the input variable into correct, non-interpretable HTML. These should be used to sanitize all input before it is displayed to the client.

PHP: string htmlspecialchars (string string [, int quote_style])

ASP.NET: Server.HTMLEncode (strHTML String)

When reflecting values into JavaScript or another format, make sure to use a type of encoding that is appropriate. Encoding data for HTML is not sufficient when it is reflected inside of a script or style sheet. For example, when reflecting data in a JavaScript string, make sure to encode all non-alphanumeric characters using hex (\xHH) encoding.

If you have JavaScript on your page that accesses unsafe information (like location.href) and writes it to the page (either with document.write, or by modifying a DOM element), make sure you encode data for HTML before writing it to the page. JavaScript does not have a built-in function to do this, but many frameworks do. If you are lacking an available function, something like the following will handle most cases:

s = s.replace(/&/g,'&').replace(/"/i,'"').replace(/</i,'<').replace(/>/i,'>').replace(/'/i,''')

Ensure that you are always using the right approach at the right time. Validating user input should be done as soon as it is received. Encoding data for display should be done immediately before displaying it.

For Security Operations:

Server-side encoding, where all dynamic content is first sent through an encoding function where Scripting tags will be replaced with codes in the selected character set, can help to prevent Cross-Site Scripting attacks.

Many web application platforms and frameworks have some built-in support for preventing Cross-Site Scripting. Make sure that any built-in protection is enabled for your platform. In some cases, a misconfiguration could allow Cross-Site Scripting. In ASP.NET, if a page's EnableViewStateMac property is set to False, the ASP.NET view state can be used as a vector for Cross-Site Scripting.

An IDS or IPS can also be used to detect or filter out XSS attacks. Below are a few regular expressions that will help detect Cross-Site Scripting.

Regex for a simple XSS attack:

/((\%3C) <)((\%2F) \/)*[a-z0-9\%]+((\%3E) >)/ix

The above regular expression would be added into a new Snort rule as follows:

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"NII Cross-Site Scripting attempt"; flow:to_server,established; pcre:"/((\%3C) <)((\%2F) \/)*[a-z0-9\%]+((\%3E) >)/i"; classtype:Web-application-attack; sid:9000; rev:5;)

Paranoid regex for XSS attacks:
/((\%3C) <)[^\n]+((\%3E) >)/I

This signature simply looks for the opening HTML tag, and its hex equivalent, followed by one or more characters other than the new line, and then followed by the closing tag or its hex equivalent. This may end up giving a few false positives depending upon how your web application and web server are structured, but it is guaranteed to catch anything that even remotely resembles a Cross-Site Scripting attack.

For QA:

Fixes for Cross-Site Scripting defects will ultimately require code based fixes.

Reference

OWASP Cross-Site Scripting Information:
https://www.owasp.org/index.php/XSS

Microsoft:
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q252985

Microsoft Anti-Cross Site Scripting Library
https://msdn.microsoft.com/en-us/security/aa973814.aspx

CERT:
http://www.cert.org/advisories/CA-2000-02.html

Apache:
http://httpd.apache.org/info/css-security/apache_specific.html

SecurityFocus.com:
http://www.securityfocus.com/infocus/1768

Classifications

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
http://cwe.mitre.org/data/definitions/79.html

CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)
http://cwe.mitre.org/data/definitions/80.html

CWE-82: Improper Neutralization of Script in Attributes of IMG Tags in a Web Page
http://cwe.mitre.org/data/definitions/82.html

CWE-83: Improper Neutralization of Script in Attributes in a Web Page
http://cwe.mitre.org/data/definitions/83.html

CWE-87: Improper Neutralization of Alternate XSS Syntax
http://cwe.mitre.org/data/definitions/87.html

CWE-116: Improper Encoding or Escaping of Output
http://cwe.mitre.org/data/definitions/116.html

CWE-692: Incomplete Blacklist to Cross-Site Scripting
http://cwe.mitre.org/data/definitions/692.html

CWE-811: OWASP Top Ten 2010 Category A2 - Cross-Site Scripting (XSS)
http://cwe.mitre.org/data/definitions/811.html

Kingdom: Input Validation and Representation
https://vulncat.fortify.com/

Privacy Violation: Social Security Number (10834)

Summary

A critical vulnerability has been detected within your web application due to the presence of one or more Social Security Numbers. If this information is carried over to a production server, it can cause major security problems. Recommendations include not storing this information on your web application.

Implication

Social Security Numbers are a highly sought out prize for attackers, and an item to which a large percentage of time would be dedicated in an effort to find. At a minimum, this can lead to theft of the victim's identity.

Fix

- When sensitive data needs to be available on your web application, mask part of the data so this information is not fully disclosed.

Here are a few examples:

Social Security Numbers:
***-**-1234
123-**-****

- If presence of social security number is being reported in a JWT, please note that unless encrypted JWT tokens do not provide privacy protection. Please do include private information in JWT unless it is securely encrypted.

Reference

Classifications

CWE-359: Privacy Violation
http://cwe.mitre.org/data/definitions/359.html

CWE-200: Information Exposure
http://cwe.mitre.org/data/definitions/200.html

Kingdom: Security Features
https://vulncat.fortify.com/

Web Server Misconfiguration: Unprotected File (157)

Summary

The Apache server status page (server-status) was found. This page contains detailed information about the current use of your web server, including information about the current hosts and requests being processed. This vulnerability is caused by default or incorrect configuration of the httpd.conf file. If exploited, an attacker could view the sensitive system information in the file. Recommendations include editing the web server configuration file to prevent access to the server-status page.

Execution

To verify the exploit, click the following link:http://zero.webappsecurity.com:80/server-status

Implication

A basic requirement for a successful attack upon your web application is reconnaissance. An attacker will employ a variety of methods, including malicious scanning agents and Google searches, to find out as much information about your web application as possible. The attacker can then use that information to formulate the next method of attack. An attacker who discovers sensitive system information has had a large portion of reconnaissance conducted for him or her.

Fix

For Security Operations:

For security reasons, you should restrict access to the server-status page in your web server configuration. To do this, comment out the following lines in the httpd.conf file:

<Location /server-status>
SetHandler server-status
</Location>

For Development:

Unless you are actively involved with implementing the web application server, there is not a wide range of available solutions to prevent problems that can occur from an attacker discovering sensitive system information about your application. Primarily, this problem will be resolved by the web application server administrator or security operations. However, there are certain actions you can take that will help to secure your web application and make it harder for an attacker to conduct a successful attack.

- Ensure that files containing sensitive information are not left publicly accessible, or that comments left inside files do not reveal the locations of directories best left confidential.
- Do not reveal information in pathnames that are publicly displayed. Do not include drive letters or directories outside of the web document root in the pathname when a file must call another file on the web server. Use pathnames that are relative to the current directory or the webroot.
- Do not display error messages to the end user that provide information, such as directory names, that could be used in orchestrating an attack.
- Restrict access to important files or directories only to those who actually need it.

For QA:

This assessment performs the rote tasks of determining the directories and contents that are available via your web application. For reasons of security, it is important to test the web application not only from the perspective of a normal user, but also from that of a malicious one. Whenever possible, adopt the mindset of an attacker when testing your web application for security defects. Access your web application from outside your firewall or IDS. Use Google or another search engine to ensure that searches for vulnerable files or directories do not return information regarding your web application. For example, an attacker will use a search engine, and search for directory listings such as 'index of / cgi-bin'. Make sure that your directory structure is not obvious, and that only files that are necessary are capable of being accessed.

Reference

Apache Documentation
Configuration Files
Apache Module mod_status

Classifications

CWE-200: Information Exposure
http://cwe.mitre.org/data/definitions/200.html

Kingdom: Environment
https://vulncat.fortify.com/

Web Server Misconfiguration: Unprotected File (708)

Summary

Webinspect has detected a backup file with the .bak extension on the target server. The severity of the threats posed by the web-accessible backup files depends on the sensitivity of the information stored in original document. Based on that information, the attacker can gain sensitive information about the site architecture, database and network access credential details, encryption keys, and so forth from these files. The attacker can use information obtained to craft precise targeted attacks, which may not otherwise be feasible, against the application.

Execution

Browse to http://zero.webappsecurity.com:80/faq.html.bak and inspect the content. Response should be a return with HTTP status code 200 and should not match target site's file not found response.

Implication

An attacker can use the information obtained from the backup file of a sensitive document to craft a precise targeted attack against the web application. Such attacks can include, but are not limited to, SQL injection, remote file system access to overwrite or inject malware, and database manipulation.

Fix

- Webroot Security Policy: Implement a security policy that prohibits storage of backup files in webroot.
- Temporary Files: Many tools and editors automatically create temporary files or backup files in the webroot. Be careful when editing files on a production server to avoid inadvertently leaving a backup or temporary copy of the file(s) in the webroot.
- Default Installations: Often, a lot of unnecessary files and folders are installed by default. For instance, IIS installations include demo applications. Be sure to remove any files or folders that are not required for application to work properly.
- Development Backup: Source code back up should not be stored and left available on the webroot.

Further QA can include test cases to look for the presence of backup files in the webroot to ensure none are left in publicly accessible folders of the web application.

Reference

OWASP - Review Old, Backup and Unreferenced Files for Sensitive Information (OTG-CONFIG-004)
CWE - 200 Information Exposure

Classifications

CWE-200: Information Exposure
http://cwe.mitre.org/data/definitions/200.html

Kingdom: Environment

https://vulncat.fortify.com/

Web Server Misconfiguration: Unprotected File (709)

Summary

Webinspect has detected a backup file with the .old extension on the target server. The severity of the threats posed by the web-accessible backup files depends on the sensitivity of the information stored in original document. Based on that information, the attacker can gain sensitive information about the site architecture, database and network access credential details, encryption keys, and so forth from these files. The attacker can use information obtained to craft precise targeted attacks, which may not otherwise be feasible, against the application.

Execution

Browse to http://zero.webappsecurity.com:80/index.html.old and inspect the content. Response should be a return with HTTP status code 200 and should not match target site's file not found response.

Implication

An attacker can use the information obtained from the backup file of a sensitive document to craft a precise targeted attack against the web application. Such attacks can include, but are not limited to, SQL injection, remote file system access to overwrite or inject malware, and database manipulation.

Fix

- Webroot Security Policy: Implement a security policy that prohibits storage of backup files in webroot.
- Temporary Files: Many tools and editors automatically create temporary files or backup files in the webroot. Be careful when editing files on a production server to avoid inadvertently leaving a backup or temporary copy of the file(s) in the webroot.
- Default Installations: Often, a lot of unnecessary files and folders are installed by default. For instance, IIS installations include demo applications. Be sure to remove any files or folders that are not required for application to work properly.
- Development Backup: Source code back up should not be stored and left available on the webroot.

Further QA can include test cases to look for the presence of backup files in the webroot to ensure none are left in publicly accessible folders of the web application.

Reference

OWASP - Review Old, Backup and Unreferenced Files for Sensitive Information (OTG-CONFIG-004)
CWE - 200 Information Exposure

Classifications

CWE-200: Information Exposure
http://cwe.mitre.org/data/definitions/200.html

Kingdom: Environment
https://vulncat.fortify.com/

Web Server Misconfiguration: Unprotected File (1368)

Summary

The file debug.txt was located. This type of file is usually left by a developer or web master to test a certain function of the web application or web server. Leaving test scripts available on the server is a very unsecure practice. The types of information that can be gleaned from test scripts include fixed authentication session id's, usernames and passwords, locations or pointers to confidential areas of the web site, and proprietary source code. With this type of information available to an attacker , they can either use it to totally breach the security of the site or use it as a stepping stone to retrieve other sensitive data. Recommendations include removing this file from the production server.

Fix

For Security Operations:
Remove the application from the server. Inform developers and administrators to remove test applications from servers when they are no longer needed. While they are in use, be sure to protect them using HTTP basic authentication.

For Development:

Contact your security or network operations team and request they investigate the issue.

For QA:
Contact your security or network operations team and request they investigate the issue.

Reference

Classifications

CWE-284: Access Control (Authorization) Issues
http://cwe.mitre.org/data/definitions/284.html

CWE-200: Information Exposure
http://cwe.mitre.org/data/definitions/200.html

Kingdom: Environment
https://vulncat.fortify.com/

Web Server Misconfiguration: Unprotected File (2083)

Summary

Webinspect has detected a backup file by replacing the extension with .old on the target server. The severity of the threats posed by the web-accessible backup files depends on the sensitivity of the information stored in original document. Based on that information, the attacker can gain sensitive information about the site architecture, database and network access credential details, encryption keys, and so forth from these files. The attacker can use information obtained to craft precise targeted attacks, which may not otherwise be feasible, against the application.

Execution

Browse to http://zero.webappsecurity.com:80/index.old and inspect the content. Response should be a return with HTTP status code 200 and should not match target site’s file not found response.

Implication

An attacker can use the information obtained from the backup file of a sensitive document to craft a precise targeted attack against the web application. Such attacks can include, but are not limited to, SQL injection, remote file system access to overwrite or inject malware, and database manipulation.

Fix

- Webroot Security Policy: Implement a security policy that prohibits storage of backup files in webroot.
- Temporary Files: Many tools and editors automatically create temporary files or backup files in the webroot. Be careful when editing files on a production server to avoid inadvertently leaving a backup or temporary copy of the file(s) in the webroot.
- Default Installations: Often, a lot of unnecessary files and folders are installed by default. For instance, IIS installations include demo applications. Be sure to remove any files or folders that are not required for application to work properly.
- Development Backup: Source code back up should not be stored and left available on the webroot.

Further QA can include test cases to look for the presence of backup files in the webroot to ensure none are left in publicly accessible folders of the web application.

Reference

OWASP - Review Old, Backup and Unreferenced Files for Sensitive Information (OTG-CONFIG-004)
CWE - 200 Information Exposure

Classifications

CWE-200: Information Exposure
http://cwe.mitre.org/data/definitions/200.html

Kingdom: Environment
https://vulncat.fortify.com/

Insecure Transport (4722)

Summary

Any area of a web application that possibly contains sensitive information or access to privileged functionality such as remote site administration functionality should utilize SSL or another form of encryption to prevent login information from being sniffed or otherwise intercepted or stolen. http://zero.webappsecurity.com:80/forgot-password.html has failed this policy. Recommendations include ensuring that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

Implication

An attacker who exploited this design vulnerability would be able to utilize the information to escalate their method of attack, possibly leading to impersonation of a legitimate user, the theft of proprietary data, or execution of actions not intended by the application developers.

Fix

For Security Operations:
Ensure that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

For Development:
Ensure that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

For QA:
Test the application not only from the perspective of a normal user, but also from the perspective of a malicious one.

Reference

Classifications

CWE-287: Improper Authentication
http://cwe.mitre.org/data/definitions/287.html

Kingdom: Security Features
https://vulncat.fortify.com/

Often Misused: Login (10595)

Summary

An unencrypted login form has been discovered. Any area of a web application that possibly contains sensitive information or access to privileged functionality such as remote site administration functionality should utilize SSL or another form of encryption to prevent login information from being sniffed or otherwise intercepted or stolen. If the login form is being served over SSL, the page that the form is being submitted to MUST be accessed over SSL. Every link/URL present on that page (not just the form action) needs to be served over HTTPS. This will prevent Man-in-the-Middle attacks on the login form. Recommendations include ensuring that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

Implication

An attacker who exploited this design vulnerability would be able to utilize the information to escalate their method of attack, possibly leading to impersonation of a legitimate user, the theft of proprietary data, or execution of actions not intended by the application developers.

Fix

Ensure that sensitive areas of your web application have proper encryption protocols in place to prevent login information and other data that could be helpful to an attacker from being intercepted.

Reference

Advisory:http://www.kb.cert.org/vuls/id/466433

Classifications

CWE-287: Improper Authentication
http://cwe.mitre.org/data/definitions/287.html

Kingdom: API Abuse
https://vulncat.fortify.com/

CWE-311: Missing Encryption of Sensitive Data
http://cwe.mitre.org/data/definitions/311.html

Cross-Frame Scripting (11293)

Summary

A Cross-Frame Scripting (XFS) vulnerability can allow an attacker to load the vulnerable application inside an HTML iframe tag on a malicious page. The attacker could use this weakness to devise a Clickjacking attack to conduct phishing, frame sniffing, social engineering or Cross-Site Request Forgery attacks.

Clickjacking
The goal of a Clickjacking attack is to deceive the victim (user) into interacting with UI elements of the attacker's choice on the target web site without their knowledge and then executing privileged functionality on the victim's behalf. To achieve this goal, the attacker must exploit the XFS vulnerability to load the attack target inside an iframe tag, hide it using Cascading Style Sheets (CSS) and overlay the phishing content on the malicious page. By placing the UI elements on the phishing page so they overlap with those on the page targeted in the attack, the attacker can ensure that the victim must interact with the UI elements on the target page not visible to the victim.

WebInspect has detected a page which potentially handles sensitive information using an HTML form with a password input field and is missing XFS protection.

This response is not protected by a valid X-Frame-Options header and a valid Content-Security-Policy(CSP) frame-ancestors directive header.Furthermore,
An effective frame-busting technique was not observed while loading this page inside a frame.

Execution

Create a test page containing an HTML iframe tag whose src attribute is set to http://zero.webappsecurity.com:80/login.html. Successful framing of the target page indicates that the application is susceptibile to XFS.

Note that WebInspect will report only one instance of this check across each host within the scope of the scan. The other visible pages on the site may, however, be vulnerable to XFS as well and therefore should be protected against it with an appropriate fix.

Implication

A Cross-Frame Scripting weakness could allow an attacker to embed the vulnerable application inside an iframe. Exploitation of this weakness could result in:

Hijacking of user events such as keystrokes
Theft of sensitive information
Execution of privileged functionality through combination with Cross-Site Request Forgery attacks

Fix

The Content Security Policy (CSP) frame-ancestors directive obsoletes the X-Frame-Options header. Both provide for a policy-based mitigation technique against cross-frame scripting vulnerabilities. The difference is that while the X-Frame-Options technique only checks against the top-level document's location, the CSP frame-ancestors header checks for conformity from all ancestors.

If both CSP frame-ancestors and X-Frame-Options headers are present and supported, the CSP directive will prevail. WebInspect recommends using both CSP frame-ancestors and X-Frame-Options headers as CSP is not supported by Internet Explorer and many older versions of other browsers.

In addition, developers must also use client-side frame busting JavaScript as a protection against XFS. This will enable users of older browsers that do not support the X-Frame-Options header to also be protected from Clickjacking attacks.

X-Frame-Options
Developers can use this header to instruct the browser about appropriate actions to perform if their site is included inside an iframe.Developers must set the X-Frame-Options header to one of the following permitted values:

- DENY

Deny all attempts to frame the page

- SAMEORIGIN

The page can be framed by another page only if it belongs to the same origin as the page being framed

- ALLOW-FROM origin

Developers can specify a list of trusted origins in the origin attribute. Only pages on origin are permitted to load this page inside an iframe

Content-Security-Policy: frame-ancestors

Developers can use the CSP header with the frame-ancestors directive, which replaces the X-Frame-Options header, to instruct the browser about appropriate actions to perform if their site is included inside an iframe. Developers can set the frame-ancestors attribute to one of the following permitted values:

- 'none'

Equivalent to "DENY" - deny all attempts to frame the page

- 'self'

Equivalent to "SAMEORIGIN" - the page can be framed by another page only if it belongs to the same origin as the page being framed

- <host-source>

Equivalent to "ALLOW-FROM" - developers can specify a list of trusted origins which maybe host name or IP address or URL scheme. Only pages on this list of trusted origin are permitted to load this page inside an iframe

- <scheme-source>

Developers can also specify a schema such as http: or https: that can frame the page.

Reference

Frame Busting:
Busting Frame Busting: A Study of Clickjacking Vulnerabilities on Popular Sites
OWASP: Busting Frame Busting

OWASP:
Clickjacking

Content-Security-Policy (CSP)
CSP: frame-ancestors

Specification:
Content Security Policy Level 2
X-Frame-Options IETF Draft

Server Configuration:
IIS
Apache, nginx

HP 2012 Cyber Security Report
The X-Frame-Options header - a failure to launch

Classifications

CWE-352: Cross-Site Request Forgery (CSRF)
http://cwe.mitre.org/data/definitions/352.html

Kingdom: Security Features
https://vulncat.fortify.com/

Expression Language Injection (11310)

Summary

WebInspect has detected an Expression Language (EL) injection vulnerability. EL injection vulnerabilities are introduced when an application fails to sufficiently validate untrusted user data before assigning it to attribute values of certain Spring MVC JSP tags.

Expression Language allows JSP pages to easily access application data stored in user-defined JavaBeans components as well the implicit objects. In addition, JSP pages can also invoke arbitrary public and static methods and perform arithmetic operations using EL expressions.

By allowing attackers to inject EL expressions through insufficiently validated user input, an application could grant unauthorized access to sensitive application and server information. Expression Language injection could also let attackers bypass HTTPOnly access restrictions imposed on cookies by exploiting access to the implicit ***cookie***object made available in EL expressions.

The affected spring framework versions include

- 3.0.0 to 3.0.5
- 2.5.0 to 2.5.6.SEC02 (community releases)
- 2.5.0 to 2.5.7.SR01 (subscription customers)

Execution

Click http://zero.webappsecurity.com:80/search.html?searchTerm=${6228%2b1320} to verify the vulnerability in a web browser.

Implication

Expression Language injection vulnerabilities can be used to steal sensitive application information as well as bypass HTTPOnly cookie access restrictions. The impact depends on the information available within the application's context.

Fix

The vulnerability can be fixed by upgrading to Spring framework versions 3.1 and above.

For versions below 3.1 (3.0.6 onwards, 2.5.6.SEC03 onwards and 2.5.7.SR02 onwards), set the value of ***springJspExpressionSupport***context parameter to ***false***.

Reference

Vendor:
SpringSource

Advisory:
Expression Language Injection

CVE:
CVE-2011-2730

Expression Language:
Expression Language Specification

Classifications

CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
http://cwe.mitre.org/data/definitions/78.html

Kingdom: Input Validation and Representation
https://vulncat.fortify.com/

Expression Language Injection (11319)

Summary

WebInspect has detected a Expression Language (EL) injection vulnerability leading to HTTPOnly restriction bypass. EL injection vulnerabilities are introduced when an application fails to sufficiently validate untrusted user data before assigning it to attribute values of certain Spring MVC JSP tags.

Expression Language allows JSP pages to easily access application data stored in user-defined JavaBeans components as well the implicit objects. In addition, JSP pages can also invoke arbitrary public and static methods and perform arithmetic operations using EL expressions.

By allowing attackers to inject EL expressions through insufficiently validated user input, an application could grant unauthorized access to sensitive application and server information. Expression Language injection could also let attackers bypass HTTPOnly access restrictions imposed on cookies by exploiting access to the implicit ***cookie***object made available in EL expressions.

The affected spring framework versions include

- 3.0.0 to 3.0.5
- 2.5.0 to 2.5.6.SEC02 (community releases)
- 2.5.0 to 2.5.7.SR01 (subscription customers)

Execution

Click http://zero.webappsecurity.com:80/search.html?searchTerm=${cookie%5B%22JSESSIONID%22%5D%2Evalue} to verify the vulnerability in a web browser.

Implication

Using Expression Language injection attackers can bypass HTTPOnly restrictions on cookies and steal sensitive information using EL expressions.

Fix

The vulnerability can be fixed by upgrading to Spring framework versions 3.1 and above.

For versions below 3.1 (3.0.6 onwards, 2.5.6.SEC03 onwards and 2.5.7.SR02 onwards), set the value of ***springJspExpressionSupport***context parameter to ***false***.

Reference

Vendor:
SpringSource

Advisory:
Expression Language Injection

CVE:
CVE-2011-2730

Expression Language:
Expression Language Specification

Classifications

CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command

Injection')
http://cwe.mitre.org/data/definitions/78.html

Kingdom: Input Validation and Representation
https://vulncat.fortify.com/