# **Security Report**

### Scenario: appPenHaxHar.scenario

Mon Oct 14 09:07:45 UTC 2019

Step: 1.1) http://hax.tor.hu/

#### **Alert Detail**

Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/robots.txt
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/sitemap.xml
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/sitemap.xml?query=query+AND+1%3D1++
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/

WASC Id	14
CWE Id	933
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
·	the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/board/?query=query+AND+1%3D1++
Parameter	X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable
	the web browser's XSS protection mechanism. The following values would attempt
	to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.
Other information	com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and
	Safari (WebKit). Note that this alert is only raised if the response body
	could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	Engure that the web browser's VCC filter is enabled by setting the V VCC Protection
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
	de.com/2014/03/guidelines-for-setting-security-headers/
WASC Id CWE Id	14 933
-	
Low(Medium)	X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This
	allows older versions of Internet Explorer and Chrome to perform MIME-sniffing
Description	on the response body, potentially causing the response body to be interpreted
2 000p.ii.o	and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content
	type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/board/?query=query+AND+1%3D1++
Parameter	X-Content-Type-Options
	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still
Other information	concern for browsers sniffing pages away from their actual content type. At
	"High" threshold this scanner will not alert on client or server error responses.
Attack	Engure that the application (upb parter acts the Content Type header appropriately
	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web
Solution	pages. If possible, ensure that the end user uses a standards-compliant and
	modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Deference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php
Reference	/List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/board/?query=query+AND+1%3D1++
Parameter	X-Frame-Options
Other information	
Attack	
	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed
Solution	only by pages on your server (e.g. it's part of a FRAMESET) then you'll want
Solution	to use SAMEORIGIN, otherwise if you never expect the page to be framed, you
	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
	px
WASC Id	15
CWE Id	16 Web Browser VSS Brotestian Not Enabled
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/login/
Parameter	X-XSS-Protection
	The X-XSS-Protection HTTP response header allows the web server to enable or disable
	the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.
Other information	com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection
	HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
	could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
	HTTP response header to '1'.

Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/login/
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/login/
Parameter	X-Frame-Options
Other information	
Attack	
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
WASC Id	15
CWE Id	16
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/html/haxtor.css
Parameter	X-Content-Type-Options This issue still applies to expert type pages (404, 403, 500, etc.) so these pages.
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php/List_of_useful_HTTP_headers
WASC Id CWE Id	15 16
Low(Medium)	Web Browser XSS Protection Not Enabled
	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Description	the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/board/
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection

	HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
Attack	could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
Solution	HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
Reference	de.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)  Description	X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current
	(early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/board/
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/board/
Parameter	X-Frame-Options
Other information	
Attack	Mant made in Wale brown as a support the V France Ontiona LITTE bandar France itle act
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
WASC Id	15
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/haxmin/
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	, and a non-zone longery.
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
Goldtion	HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content
	type (if one is set), rather than performing MIME-sniffing.

URL	http://hax.tor.hu/haxmin/
Parameter	•
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At
	"High" threshold this scanner will not alert on client or server error responses.
Attack	
	Ensure that the application/web server sets the Content-Type header appropriately,
	and that it sets the X-Content-Type-Options header to 'nosniff' for all web
Solution	pages. If possible, ensure that the end user uses a standards-compliant and
	modern web browser that does not perform MIME-sniffing at all, or that can
	be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php
W(V CC 14	/List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
•	attacks.
URL	http://hax.tor.hu/haxmin/
Parameter	X-Frame-Options
Other information	
Attack	
,ao.:	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set
	on all web pages returned by your site (if you expect the page to be framed
Colution	only by pages on your server (e.g. it's part of a FRAMESET) then you'll want
Solution	to use SAMEORIGIN, otherwise if you never expect the page to be framed, you
	should use DENY. ALLOW-FROM allows specific websites to frame the web page
	in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
	px
WASC Id	15
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Description	the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/peek/
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	could potertially contain an 700 payload (with a text-based content type, with a non-zero length).
	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
Solution	HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/quidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	Private IP Disclosure
Description	A private IP (such as 10.x.x.x, 172.x.x.x, 192.168.x.x) or an Amazon EC2 private hostname (for example, ip-10-0-56-78) has been found in the HTTP response body. This information might be helpful for further attacks targeting internal systems.
URL	http://hax.tor.hu/peek/
Parameter	mp., maxico mur pooliv
	10.0.0 5
Other information	10.0.0.5
Attack	Description with the principle ID address from the LITTD responses hady. For comments, use ICD/ACD/DLID
Solution	Remove the private IP address from the HTTP response body. For comments, use JSP/ASP/PHP comment instead of HTML/JavaScript comment which can be seen by client browsers.
Reference	https://tools.ietf.org/html/rfc1918
WASC Id	13
CWE Id	200
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/peek/
Parameter	X-Content-Type-Options
	• • •
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages

	are often still affected by injection issues, in which case there is still
	concern for browsers sniffing pages away from their actual content type. At
	"High" threshold this scanner will not alert on client or server error responses.
Attack	
	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web
Solution	pages. If possible, ensure that the end user uses a standards-compliant and
	modern web browser that does not perform MIME-sniffing at all, or that can
	be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List of useful HTTP headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
	attacks.
URL Parameter	http://hax.tor.hu/peek/ X-Frame-Options
Other information	λ-riame-options
Attack	
Allack	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set
Solution	on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page
Reference	in supported web browsers). http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
WASC Id	px 15
CWE Id	16
Low(Medium)	X-Content-Type-Options Header Missing
Low(Medium)	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This
	allows older versions of Internet Explorer and Chrome to perform MIME-sniffing
Description	on the response body, potentially causing the response body to be interpreted
Docomption	and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content
	type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css
Parameter	X-Content-Type-Options
	This issue still applies to error type pages (401, 403, 500, etc) as those pages
Other information	are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
	Ensure that the application/web server sets the Content-Type header appropriately,
Solution	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and
Solution	modern web browser that does not perform MIME-sniffing at all, or that can
	be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php
WASC Id	/List_of_useful_HTTP_headers 15
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Description	the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/irc/
Parameter	X-XSS-Protection
	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt
	to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.
Other information	com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection
	HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
	could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
WASC Id	de.com/2014/03/guidelines-for-setting-security-headers/
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
_ow(ivicalarii)	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This
Description	allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted

	and displayed as a content type other than the declared content type. Current
	(early 2014) and legacy versions of Firefox will use the declared content
LIDI	type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/irc/
Parameter	X-Content-Type-Options
	This issue still applies to error type pages (401, 403, 500, etc) as those pages
Other information	are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At
	"High" threshold this scanner will not alert on client or server error responses.
Attack	right and did the deather without don't should be derived should be desired.
Allack	Ensure that the application/web server sets the Content-Type header appropriately,
	and that it sets the X-Content-Type-Options header to 'nosniff' for all web
Solution	pages. If possible, ensure that the end user uses a standards-compliant and
	modern web browser that does not perform MIME-sniffing at all, or that can
	be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php
	/List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
,	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
Description	attacks.
URL	http://hax.tor.hu/irc/
Parameter	X-Frame-Options
Other information	
Attack	
/ ttdot	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set
	on all web pages returned by your site (if you expect the page to be framed
Solution	only by pages on your server (e.g. it's part of a FRAMESET) then you'll want
Solution	to use SAMEORIGIN, otherwise if you never expect the page to be framed, you
	should use DENY. ALLOW-FROM allows specific websites to frame the web page
	in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
WASC Id	px 45
	15
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
•	the 'X-XSS-Protection' HTTP response header on the web server
URL	the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B
•	·
URL	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable
URL	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt
URL Parameter	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.
URL	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection
URL Parameter	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and
URL Parameter	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection
URL Parameter	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
URL Parameter  Other information  Attack	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
URL Parameter  Other information	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
URL Parameter  Other information  Attack Solution	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
URL Parameter  Other information  Attack Solution  Reference	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
URL Parameter  Other information  Attack Solution  Reference WASC Id	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  14 933
URL Parameter  Other information  Attack Solution  Reference WASC Id	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  14  933  X-Content-Type-Options Header Missing
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  14 933
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id  Low(Medium)	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  14 933  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  14 933  X-Content-Type-Options Header Missing The Anti-MIME-sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id  Low(Medium)	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/ 14 933  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id  Low(Medium)  Description	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/ 14 933  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection on the com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco.de.com/2014/03/guidelines-for-setting-security-headers/  14 933  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id  Low(Medium)  Description	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/ 14 933  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing. http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-Content-Type-Options
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection  HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/  14 933  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-Content-Type-Options  This issue still applies to error type pages (401, 403, 500, etc) as those pages
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/ 14 933  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing. http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-Content-Type-Options This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection  HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/  14 933  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if no is set), rather than performing MIME-sniffing.  http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-Content-Type-Options  This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/ 14 933  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing. http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-Content-Type-Options This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection  HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  14  933  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. (if one is set), rather than performing MIME-sniffing. http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-Content-Type-Options  This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection  HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/  14 933  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if no is set), rather than performing MIME-sniffing.  http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-Content-Type-Options  This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/  14  9333  X-Content-Type-Options Header Missing The Anti-MIME-sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (for no is set), rather than performing MIME-sniffing.  http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-Content-Type-Options This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information  Attack	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (Webkit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/  14  333  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-Content-Type-Options This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately, and that it
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information  Attack	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection Com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/ 14 933  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing. http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-Content-Type-Option This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At 'High' threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information  Attack	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/  14  933  X-Content-Type-Options Header Missing The Anti-MilME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Interpreted in the performing MIME-sniffing.  http://hax.tor.hu/login/grueny=javascript%3Aalert%281%29%3B  X-Content-Type-Options This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At 'High' threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id  Low(Medium)  Description  URL Parameter  Other information  Attack  Solution  Reference	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode-block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alter is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco.de.com/2014/03/guidelines-for-setting-security-headers/ 14 933  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff for all web pages. If possible, ensure that does not perform MIME-sniffing at all, or that can be directed
URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information  Attack  Solution	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/  14  933  X-Content-Type-Options Header Missing The Anti-MilME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Interpreted in the performing MIME-sniffing.  http://hax.tor.hu/login/grueny=javascript%3Aalert%281%29%3B  X-Content-Type-Options This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At 'High' threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header

CWE Id Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
URL	attacks. http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B
Parameter	X-Frame-Options
Other information	A Figure Options
Attack	
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
WASC Id	15
CWE Id	16
Low(Medium)	Cross-Domain JavaScript Source File Inclusion
Description	The page includes one or more script files from a third-party domain.
URL	http://hax.tor.hu/links/
Parameter	http://pagead2.googlesyndication.com/pagead/show_ads.js
Other information	
Attack	
Solution	Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.
Reference	
WASC Id	15
CWE Id	829
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/links/
Parameter	X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable
Other information	the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/links/
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing. http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php
Reference	/List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
_ 555	attacks.

	http://hax.tor.hu/links/
	X-Frame-Options
	ATTAINE OPIOIS
Attack	M
	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed
	only by pages on your server (e.g. it's part of a FRAMESET) then you'll want
	to use SAMEORIGIN, otherwise if you never expect the page to be framed, you
	should use DENY. ALLOW-FROM allows specific websites to frame the web page
	in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
	px
	15
	16
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
	http://hax.tor.hu/irc/?query=query%22+AND+%221%22%3D%221%22++
	X-XSS-Protection
	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt
	to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.
	com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection
	HTTP response header is currently supported on Internet Explorer, Chrome and
	Safari (WebKit). Note that this alert is only raised if the response body
	could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
	HTTP response header to '1'.
	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
	de.com/2014/03/guidelines-for-setting-security-headers/
	933
Low(Medium)	X-Content-Type-Options Header Missing
	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This
	allows older versions of Internet Explorer and Chrome to perform MIME-sniffing
	on the response body, potentially causing the response body to be interpreted
•	and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content
	type (if one is set), rather than performing MIME-sniffing.
	http://hax.tor.hu/irc/?query=query%22+AND+%221%22%3D%221%22++
	X-Content-Type-Options
	This issue still applies to error type pages (401, 403, 500, etc) as those pages
	are often still affected by injection issues, in which case there is still
Ciner Information	concern for browsers sniffing pages away from their actual content type. At
	"High" threshold this scanner will not alert on client or server error responses.
Attack	
	Ensure that the application/web server sets the Content-Type header appropriately,
	and that it sets the X-Content-Type-Options header to 'nosniff' for all web
	pages. If possible, ensure that the end user uses a standards-compliant and
	modern web browser that does not perform MIME-sniffing at all, or that can
	be directed by the web application/web server to not perform MIME-sniffing.
	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List of useful HTTP headers
	15
	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
·	attacks.
	http://hax.tor.hu/irc/?query=query%22+AND+%221%22%3D%221%22++
Parameter	X-Frame-Options
	X-Frame-Options
Parameter	X-Frame-Options
Parameter Other information Attack	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set
Parameter Other information Attack	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed
Parameter Other information Attack	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want
Parameter Other information Attack Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you
Parameter Other information Attack Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page
Parameter Other information Attack Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Parameter Other information Attack Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
Parameter Other information Attack Solution Reference	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
Parameter Other information Attack  Solution  Reference WASC Id	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
Parameter Other information Attack  Solution  Reference WASC Id CWE Id	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15
Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15 16  Web Browser XSS Protection Not Enabled
Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15

URL	http://hax.tor.hu/board/?query=query+AND+1%3D1
Parameter	
raiaiiielei	X-XSS-Protection  The X-XSS Protection HTTP response header allows the web converte enable or disable.
	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt
	to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.
Other information	com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection
	HTTP response header is currently supported on Internet Explorer, Chrome and
	Safari (WebKit). Note that this alert is only raised if the response body
Attack	could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
Solution	HTTP response header to '1'.
Deference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
Reference	de.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This
	allows older versions of Internet Explorer and Chrome to perform MIME-sniffing
Description	on the response body, potentially causing the response body to be interpreted
	and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content
	type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/board/?query=query+AND+1%3D1
Parameter	X-Content-Type-Options
	This issue still applies to error type pages (401, 403, 500, etc) as those pages
Other information	are often still affected by injection issues, in which case there is still
other information	concern for browsers sniffing pages away from their actual content type. At
Attook	"High" threshold this scanner will not alert on client or server error responses.
Attack	Encurs that the application (such converted the Content Type header appropriately
	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web
Solution	pages. If possible, ensure that the end user uses a standards-compliant and
	modern web browser that does not perform MIME-sniffing at all, or that can
	be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
011 L 14	
Medium(Medium)	
Medium(Medium)	X-Frame-Options Header Not Set
Medium(Medium)  Description	
,	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
Description	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
Description URL	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1
Description URL Parameter	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1
Description URL Parameter Other information	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set
Description URL Parameter Other information	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed
Description URL Parameter Other information	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want
Description URL Parameter Other information Attack	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you
Description URL Parameter Other information Attack	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want
Description URL Parameter Other information Attack Solution	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
Description URL Parameter Other information Attack Solution Reference	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
Description URL Parameter Other information Attack  Solution  Reference WASC Id	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
Description URL Parameter Other information Attack Solution Reference WASC Id CWE Id	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15
Description URL Parameter Other information Attack  Solution  Reference WASC Id	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description URL	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15 16  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++  X-XSS-Protection
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description URL	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium) Description URL Parameter	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++  X-XSS-Protection
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description URL	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; report-shtp://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium) Description URL Parameter	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection" HTTP response header on the web server http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium) Description URL Parameter	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++  X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium) Description URL Parameter	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium) Description URL Parameter  Other information	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++  X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter  Other information	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++  X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKir), Note that this alert is only raised if the response bedy could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium) Description URL Parameter  Other information  Attack Solution	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode-block X-XSS-Protection: 1; report-http://www.example.com/xss The following values would disable it X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection  http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter  Other information  Attack Solution  Reference	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/haxmin/?query=query%27+AND+%271%2730b/271%27++  X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report-http://www.example.com/xss The following values would disable it: X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection  HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter  Other information  Attack Solution  Reference WASC Id	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY, ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++  X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-SSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKil). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection  HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracoddec.com/2014/03/guidelines-for-setting-security-headers/
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter  Other information  Attack Solution  Reference	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/haxmin/?query=query%27+AND+%271%2730b/271%27++  X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report-http://www.example.com/xss The following values would disable it: X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection  HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/

	The Act MINAT Column hander V Contest Time Ontine and set to be exist. This
	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing
<b>5</b>	on the response body, potentially causing the response body to be interpreted
Description	and displayed as a content type other than the declared content type. Current
	(early 2014) and legacy versions of Firefox will use the declared content
URL	type (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++
Parameter	X-Content-Type-Options
raiaillelei	This issue still applies to error type pages (401, 403, 500, etc) as those pages
Other information	are often still affected by injection issues, in which case there is still
Other information	concern for browsers sniffing pages away from their actual content type. At
A tto alc	"High" threshold this scanner will not alert on client or server error responses.
Attack	Ensure that the application/web server sets the Content-Type header appropriately,
	and that it sets the X-Content-Type-Options header to 'nosniff' for all web
Solution	pages. If possible, ensure that the end user uses a standards-compliant and
	modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
D (	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php
Reference	/List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
	attacks.
URL	http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++
Parameter Other information	X-Frame-Options
Other information Attack	
Allack	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set
	on all web pages returned by your site (if you expect the page to be framed
Solution	only by pages on your server (e.g. it's part of a FRAMESET) then you'll want
Columbia	to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page
	in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
	px
WASC Id	15
CWE Id	16
Laver/Maralines	Mak Drawan VCC Dratastian Nat Enablad
Low(Medium)	Web Browser XSS Protection Not Enabled
Low(Medium)  Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Description URL	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/
Description URL	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt
Description URL Parameter	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.
Description URL	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and
Description URL Parameter	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
Description URL Parameter Other information	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and
Description URL Parameter Other information	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Description URL Parameter Other information	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
Description URL Parameter Other information	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
Description URL Parameter Other information Attack Solution Reference	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
Description URL Parameter  Other information  Attack Solution Reference WASC Id	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  14
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id  Low(Medium)	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection  HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  14 933  Web Browser XSS Protection Not Enabled
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  14
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id  Low(Medium)	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/  X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  14 933  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id  Low(Medium)  Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/  X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  14 933  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description URL	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (Webkit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/  14  933  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/?query=query+AND+1%3D1++  X-XSS-Protection HTTP response header allows the web server to enable or disable
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description URL	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection  HTTP response header to '1'.  https://www.owasp.org/index.ptp/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  14  933  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/?query=query+AND+1%3D1++  X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description URL	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (Webkit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/  14  933  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/?query=query+AND+1%3D1++  X-XSS-Protection HTTP response header allows the web server to enable or disable
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode-block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/  14 933  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/?query=query+AND+1%3D1++ X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode-block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xsS The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  14 933  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/?query=query+AND+1%3D1++ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xsS The following values would disable it: X-SS-Protection: 1 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode-block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/  14 933  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/?query=query+AND+1%3D1++ X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode-block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter  Other information	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xsS The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  14 933  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/?query=query+AND+1%3D1++ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xsS The following values would disable it: X-SS-Protection: 1 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter  Other information	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection" HTTP response header on the web server http://hax.tor.hu/html/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode-block X-XSS-Protection: 1; report-http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: x-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: x-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: x-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: x-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: x-XS
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter  Other information	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection" HTTP response header on the web server http://hax.tor.hu/html/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1, the disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1, report-http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco.de.com/2014/03/guidelines-for-setting-security-headers/  14  933  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection is not enabled, or is disabled by the very because the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode-slock X-XSS-Protection: 1; report-http://www.example.com/xss Intelligent is enabled, by setting the X-XSS-Protection HTTP response header is currently s
Description URL Parameter  Other information  Attack Solution Reference WASC Id CWE Id Low(Medium) Description URL Parameter  Other information  Attack Solution Reference	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection" HTTP response header on the web server http://hax.tor.hu/html/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco.de.com/2014/03/guidelines-for-setting-security-headers/  14  333  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/html/?query=query+AND+1%3D1++  X-XSS-Protection' HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.  Https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco.de.com/2014/03/guidelines-for-setting-security-headers/
Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter  Other information  Attack Solution	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection" HTTP response header on the web server http://hax.tor.hu/html/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1, the disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1, report-http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco.de.com/2014/03/guidelines-for-setting-security-headers/  14  933  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection is not enabled, or is disabled by the configuration of the "X-XSS-Protection is not enabled, or is disabled by the very because the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode-slock X-XSS-Protection: 1; report-http://www.example.com/xss Intelligent is enabled, by setting the X-XSS-Protection HTTP response header is currently s

High(Medium)	Cross Site Scripting (Reflected)
High(Mediaili)	Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied
	code into a user's browser instance. A browser instance can be a standard
	web browser client, or a browser object embedded in a software product such
	as the browser within WinAmp, an RSS reader, or an email client. The code
	itself is usually written in HTML/JavaScript, but may also extend to VBScript,
	ActiveX, Java, Flash, or any other browser-supported technology. When an attacker
	gets a user's browser to execute his/her code, the code will run within the
	security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible
	by the browser. A Cross-site Scripted user could have his/her account hijacked
	(cookie theft), their browser redirected to another location, or possibly
	shown fraudulent content delivered by the web site they are visiting. Cross-site
	Scripting attacks essentially compromise the trust relationship between a
	user and the web site. Applications utilizing browser object instances which
	load content from the file system may execute code under the local machine zone allowing for system compromise. There are three types of Cross-site
Description	Scripting attacks: non-persistent, persistent and DOM-based. Non-persistent
	attacks and DOM-based attacks require a user to either visit a specially crafted
	link laced with malicious code, or visit a malicious web page containing a
	web form, which when posted to the vulnerable site, will mount the attack.
	Using a malicious form will oftentimes take place when the vulnerable resource
	only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript).
	Upon clicking on the malicious link or submitting the malicious form, the
	XSS payload will get echoed back and will get interpreted by the user's browser
	and execute. Another technique to send almost arbitrary requests (GET and
	POST) is by using an embedded client, such as Adobe Flash. Persistent attacks
	occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include
	message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with
	any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the
	code.
URL	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B
Parameter	query
Other information	
Attack	javascript:alert(1);
	Phase: Architecture and Design Use a vetted library or framework that does not allow
Solution	this weakness to occur or provides constructs that make this weakness easier
	to avoid. Examples of libraries and framework
High(Medium)	ISQL Injection
· · · · · · · · · · · · · · · · · · ·	
Description	SQL injection may be possible.
Description URL	
Description	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query
Description URL	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query
Description URL Parameter	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being
Description URL	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query
Description URL Parameter	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data
Description URL Parameter	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected
Description URL Parameter Other information	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Description URL Parameter Other information	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses
Description URL Parameter Other information	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by
Description URL Parameter Other information	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type
Description URL Parameter Other information	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by
Description URL Parameter Other information	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used,
Description URL Parameter Other information Attack	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received
Description URL Parameter Other information Attack	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
Description URL Parameter Other information Attack	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege
Description URL Parameter Other information Attack	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid
Description URL Parameter Other information Attack	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege
Description URL Parameter Other information Attack Solution	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1] and [query AND 1=1 OR 1=1] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by "?" If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
Description URL Parameter Other information Attack Solution	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet
Description URL Parameter Other information Attack Solution Reference WASC Id	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet
Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1] and [query AND 1=1 OR 1=1] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by "?" If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19
Description URL Parameter  Other information  Attack  Solution  Reference WASC Id	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet
Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1] and [query AND 1=1 OR 1=1] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by "?" If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19
Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id  High(Medium)	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1] and [query AND 1=1 OR 1=1] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by "?" If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19 89  SQL Injection
Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id  High(Medium) Description	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1] and [query AND 1=1 OR 1=1] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user, Dosible. In particular, avoid using the 'eas' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19  SQL Injection SQL injection may be possible.
Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id High(Medium) Description URL	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'eas' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19 89  SQL Injection SQL injection my be possible. http://hax.tor.hu/html/haxtor.css?query=query+AND+1%3D1++
Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id High(Medium) Description URL Parameter	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19  89  SQL Injection SQL injection may be possible. http://hax.tor.hu/html/haxtor.css?query=query+AND+1%3D1++ query The page results were successfully manipulated using the boolean conditions [query AND 1=1 ] and [query AND 1=2 ] The parameter value being modified was
Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id High(Medium) Description URL	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do 'not' concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19  SQL Injection SQL injection may be possible. http://hax.tor.hu/html/haxtor.css?query=query+AND+1%3D1++ query The page results were successfully manipulated using the boolean conditions [query AND 1=1 ] and [query AND 1=2 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data
Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id High(Medium) Description URL Parameter	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1] and [query AND 1=1 OR 1=1] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do 'not' concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a Whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  SQL Injection SQL injection may be possible. http://hax.tor.hu/html/haxtor.css?query=query+AND+1%3D1++ query The page results were successfully manipulated using the boolean conditions [query AND 1=1] and [query AND 1=2] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data
Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id  High(Medium) Description URL Parameter  Other information	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1] and [query AND 1=1 OR 1=1] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do 'not' concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'Whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  SQL Injection SQL injection SQL injection may be possible. http://hax.tor.hu/html/haxtor.css?query=query+AND+1%3D1++ query The page results were successfully manipulated using the boolean conditions [query AND 1=1] and [query AND 1=2] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data
Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id High(Medium) Description URL Parameter	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1] and [query AND 1=1 OR 1=1] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do 'not' concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19  89  SQL Injection The page results were successfully manipulated using the boolean conditions [query AND 1=1 ] and [query AND 1=2 ] The parameter value being modified was NOT stripped from the HTML output for the purpose of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting
Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id  High(Medium) Description URL Parameter  Other information	SQL injection may be possible.  http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1] and [query AND 1=1 OR 1=1] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query AND 1=1 OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do 'not' concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'Whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  SQL Injection SQL injection SQL injection may be possible. http://hax.tor.hu/html/haxtor.css?query=query+AND+1%3D1++ query The page results were successfully manipulated using the boolean conditions [query AND 1=1] and [query AND 1=2] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data

JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do \*not\* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top\_10\_2010-A1 https://www.owasp.org/index.php/SQL\_Injection\_Preventi Reference on Cheat Sheet WASC Id 19 CWE Id 89 High(Medium) SQL Injection Description SQL injection may be possible. URL http://hax.tor.hu/sitemap.xml?query=query+AND+1%3D1+--+ Parameter The page results were successfully manipulated using the boolean conditions [query AND 1=1 -- ] and [query AND 1=2 -- ] The parameter value being modified was Other information NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter Attack query AND 1=1 --Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do \*not\* concatenate strings into queries in the stored procedure, Solution or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top\_10\_2010-A1 https://www.owasp.org/index.php/SQL\_Injection\_Preventi Reference on\_Cheat\_Sheet WASC Id 19 CWE Id 89 |High(Medium) SQL Injection Description SQL injection may be possible. http://hax.tor.hu/links/?query=query+AND+1%3D1+--+ URL Parameter The page results were successfully manipulated using the boolean conditions [query AND 1=1 -- ] and [query OR 1=1 -- ] The parameter value being modified was Other information NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter Attack query OR 1=1 --Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do \*not\* concatenate strings into queries in the stored procedure, Solution or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top\_10\_2010-A1 https://www.owasp.org/index.php/SQL\_Injection\_Preventi Reference on\_Cheat\_Sheet WASC Id 19 CWE Id 89 SQL Injection High(Medium) Description SQL injection may be possible. http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27+--+%22+AND+%221%2 **URL** 2%3D%221%22+--+ Parameter The page results were successfully manipulated using the boolean conditions [query' AND '1'='1' -- " AND "1"="1" -- ] and [query' AND '1'='1' -- " AND "1"="2" Other information -- ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The

	vulnerability was detected by successfully restricting the data originally
	returned, by manipulating the parameter
Attack	query' AND '1'='1' " AND "1"="1"
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create
Coldion	dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/stats/detect.txt?query=query+AND+1%3D1
Parameter	query
T dramotor	The page results were successfully manipulated using the boolean conditions [query
Other information	AND 1=1] and [query OR 1=1] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query OR 1=1
	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type  checking and parameterized queries. If database Stored Procedures can be used,  use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/?query=query+AND+1%3D1++
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query AND 1=1 ] and [query OR 1=1 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query OR 1=1
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type  checking and parameterized queries. If database Stored Procedures can be used,  use them. Do *not* concatenate strings into queries in the stored procedure,  or use 'exec', 'exec immediate', or equivalent functionality! Do not create
Solution	dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/board/?query=query%27+AND+%271%27%3D%271%27++
Davasatas	query
Parameter	117

	The page results were successfully manipulated using the boolean conditions [query'
	AND '1'='1' ] and [query' AND '1'='2' ] The parameter value being modified
Other information	was NOT stripped from the HTML output for the purposes of the comparison Data
	was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	query' AND '1'='1'
, maon	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by
	'?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized gueries. If database Stored Procedures can be used,
	use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create
	dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
D (	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
Reference	on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++
Parameter	query
	The page results were successfully manipulated using the boolean conditions [query
Other information	OR 1=1 ] and [query AND 1=2 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was
	NOT returned for the original parameter. The vulnerability was detected by
A	successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query OR 1=1
	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by
	'?' If the application uses ASP, use ADO Command Objects with strong type
	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create
	dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
D-4	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
Reference	on_Cheat_Sheet
	10
WASC Id	19
CWE Id	89
CWE Id High(Medium)	
CWE Id High(Medium) Description	89  SQL Injection SQL injection may be possible.
CWE Id High(Medium) Description URL	SQL Injection SQL injection may be possible. http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++
CWE Id High(Medium) Description	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query
CWE Id High(Medium) Description URL	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query  The page results were successfully manipulated using the boolean conditions [query"
CWE Id High(Medium) Description URL	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query
CWE Id High(Medium) Description URL Parameter	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by
CWE Id High(Medium) Description URL Parameter Other information	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
CWE Id High(Medium) Description URL Parameter	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter query" AND "1"="1"
CWE Id High(Medium) Description URL Parameter Other information	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
CWE Id High(Medium) Description URL Parameter Other information	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter query" AND "1"="1"  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by
CWE Id High(Medium) Description URL Parameter Other information	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++  query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified  was NOT stripped from the HTML output for the purposes of the comparison Data  was returned for the original parameter. The vulnerability was detected by  successfully restricting the data originally returned, by manipulating the parameter  query" AND "1"="1"  Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type
CWE Id High(Medium) Description URL Parameter Other information	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter query" AND "1"="1"  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by
CWE Id High(Medium) Description URL Parameter Other information	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++  query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified  was NOT stripped from the HTML output for the purposes of the comparison Data  was returned for the original parameter. The vulnerability was detected by  successfully restricting the data originally returned, by manipulating the parameter  query" AND "1"="1"  Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type  checking and parameterized queries. If database Stored Procedures can be used,  use them. Do *not* concatenate strings into queries in the stored procedure,  or use 'exec', 'exec immediate', or equivalent functionality! Do not create
CWE Id High(Medium) Description URL Parameter Other information Attack	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter query" AND "1"="1"  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received
CWE Id High(Medium) Description URL Parameter Other information Attack	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter query" AND "1"="1"  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege
CWE Id High(Medium) Description URL Parameter Other information Attack	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++  query  The page results were successfully manipulated using the boolean conditions [query"  AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified  was NOT stripped from the HTML output for the purposes of the comparison Data  was returned for the original parameter. The vulnerability was detected by  successfully restricting the data originally returned, by manipulating the parameter  query" AND "1"="1"  Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type  checking and parameterized queries. If database Stored Procedures can be used,  use them. Do "not" concatenate strings into queries in the stored procedure,  or use 'exec', 'exec immediate', or equivalent functionality! Do not create  dynamic SQL queries using simple string concatenation. Escape all data received  from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'  of disallowed characters in user input. Apply the principle of least privilege  by using the least privileged database user possible. In particular, avoid
CWE Id High(Medium) Description URL Parameter Other information Attack	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter query" AND "1"="1"  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by "?" If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
CWE Id High(Medium) Description URL Parameter Other information Attack Solution	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++  query  The page results were successfully manipulated using the boolean conditions [query"  AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified  was NOT stripped from the HTML output for the purposes of the comparison Data  was returned for the original parameter. The vulnerability was detected by  successfully restricting the data originally returned, by manipulating the parameter  query" AND "1"="1"  Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type  checking and parameterized queries. If database Stored Procedures can be used,  use them. Do "not" concatenate strings into queries in the stored procedure,  or use 'exec', 'exec immediate', or equivalent functionality! Do not create  dynamic SQL queries using simple string concatenation. Escape all data received  from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'  of disallowed characters in user input. Apply the principle of least privilege  by using the least privileged database user possible. In particular, avoid
CWE Id High(Medium) Description URL Parameter Other information Attack Solution Reference	SQL Injection  SQL Injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter query" AND "1"="1"  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by "?" If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet
CWE Id High(Medium) Description URL Parameter Other information Attack  Solution  Reference WASC Id	SQL Injection  SQL Injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter query" AND "1"="1"  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by "?" If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database users possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet  19
CWE Id High(Medium) Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter query" AND "1"="1"  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by "2" If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19
CWE Id High(Medium) Description URL Parameter Other information Attack  Solution  Reference WASC Id	SQL Injection  SQL Injection may be possible.  http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++ query  The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" AND "1"="2" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter query" AND "1"="1"  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by "?" If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database users possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet  19

Description	
URL	http://hax.tor.hu/stats?query=query%22+AND+%221%22%3D%221%22++
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query"  AND "1"="1" ] and [query" OR "1"="1" ] The parameter value being modified  was stripped from the HTML output for the purposes of the comparison Data  was NOT returned for the original parameter. The vulnerability was detected  by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query" OR "1"="1"
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/peek/?all=1%27+AND+%271%27%3D%271
Parameter	all
Other information	The page results were successfully manipulated using the boolean conditions [1' AND '1'='1] and [1' AND '1'='2] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully
	restricting the data originally returned, by manipulating the parameter
Attack	1' AND '1'='1  Do not trust client side input, even if there is client side validation in place.
Solution	In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
WASC Id	on_Cheat_Sheet 19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/irc/?query=query%22+AND+%221%22%3D%221%22+++OR+1%3D1++
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" OR 1=1 ] and [query" AND "1"="1" AND 1=2 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query" AND "1"="1" OR 1=1
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/html?query=query%25
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query%] and [query%] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query%
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec' immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/board/?query=query%25
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query%] and [query%] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving
Attack	more data than originally returned, by manipulating the parameter query%
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec' immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
WASC Id	on_Cheat_Sheet 19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/irc/?query=query%25
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query%] and [query%] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query%
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	,,

	but minimized its impact. Creat the minimum database access that is necessary for the application
D (	but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
Reference	on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B+ASC+++
Parameter	query
	The original page results were successfully replicated using the "ORDER BY" expression
Other information	[javascript:alert(1); ASC ] as the parameter value The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison
Attack	javascript:alert(1); ASC
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
D-f	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
Reference	on_Cheat_Sheet
WASC Id	19
CWE Id	89
Low(Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/welcome/
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
•	the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/welcome/
Parameter  Other information	X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
Attack	could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
Solution	HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/welcome/
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	

Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/welcome/
Parameter	X-Frame-Options
Other information Attack	
Allack	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set
Solution	on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
WASC Id	15
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/links/?query=query+AND+1%3D1++
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Cross-Domain JavaScript Source File Inclusion
Description	The page includes one or more script files from a third-party domain.
URL	http://hax.tor.hu/links/?query=query+AND+1%3D1++
Parameter Other information	http://pagead2.googlesyndication.com/pagead/show_ads.js
Attack	
Solution	Ensure JavaScript source files are loaded from only trusted sources, and the sources
Reference	can't be controlled by end users of the application.
WASC Id	15
CWE Id	829
Low(Medium)	Web Browser XSS Protection Not Enabled
	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Description	the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/links/?query=query+AND+1%3D1++
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
	could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This
Description	allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current

	(early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/links/?query=query+AND+1%3D1++
Parameter	X-Content-Type-Options
	This issue still applies to error type pages (401, 403, 500, etc) as those pages
Other information	are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At
	"High" threshold this scanner will not alert on client or server error responses.
Attack	
	Ensure that the application/web server sets the Content-Type header appropriately,
Solution	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and
	modern web browser that does not perform MIME-sniffing at all, or that can
	be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/qq622941%28v=vs.85%29.aspx https://www.owasp.org/index.php
Reference	/List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
URL	attacks. http://hax.tor.hu/links/?query=query+AND+1%3D1++
Parameter	X-Frame-Options
Other information	
Attack	
	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set
	on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want
Solution	to use SAMEORIGIN, otherwise if you never expect the page to be framed, you
	should use DENY. ALLOW-FROM allows specific websites to frame the web page
	in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
Reference	px
WASC Id	15
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
	A cookie has been set without the HttpOnly flag, which means that the cookie can
Description	be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this
	is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/
Parameter	HAXTOR
Other information Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
	A cookie has been set without the HttpOnly flag, which means that the cookie can
Description	be accessed by JavaScript. If a malicious script can be run on this page then
·	the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id CWE Id	13
Low(Medium)	Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can
December tiens	be accessed by JavaScript. If a malicious script can be run on this page then
Description	the cookie will be accessible and can be transmitted to another site. If this
LIDI	is a session cookie then session hijacking may be possible.
URL Parameter	http://hax.tor.hu/board/?query=query+AND+1%3D1  HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13

CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
Low(Mcalaill)	A cookie has been set without the HttpOnly flag, which means that the cookie can
Description	be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++
Parameter	HAXTOR
	THAT ON
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
LOW(Mediairi)	, , ,
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/peek/?all=1
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Description	the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/peek/?all=1
Parameter	X-XSS-Protection
	The X-XSS-Protection HTTP response header allows the web server to enable or disable
Other information	the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	Private IP Disclosure
Description	A private IP (such as 10.x.x.x, 172.x.x.x, 192.168.x.x) or an Amazon EC2 private hostname (for example, ip-10-0-56-78) has been found in the HTTP response body. This information might be helpful for further attacks targeting internal systems.
URL	http://hax.tor.hu/peek/?all=1
Parameter	
Other information	10.0.0.5
Attack	
Solution	Remove the private IP address from the HTTP response body. For comments, use JSP/ASP/PHP comment instead of HTML/JavaScript comment which can be seen by client browsers.
Reference	https://tools.ietf.org/html/rfc1918
WASC Id	13
CWE Id	200
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/peek/?all=1
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	The state of the s

Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
,	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
Description	attacks.
URL	http://hax.tor.hu/peek/?all=1
Parameter	X-Frame-Options
Other information	
Attack	Mark and day With house are assessed the Vi France Online of LITTO has day France ittered
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
WASC Id	px 15
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
Low(Mcalaill)	A cookie has been set without the HttpOnly flag, which means that the cookie can
Description	be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/peek/?query=query%27+AND+%271%27%3D%271%27++
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/peek/?query=query%27+AND+%271%27%3D%271%27++ X-XSS-Protection
Parameter Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	could pose main an 7000 payload (main a lost based sentent type, main a non-2010 longary).
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/quidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	Private IP Disclosure
Description	A private IP (such as 10.x.x.x, 172.x.x.x, 192.168.x.x) or an Amazon EC2 private hostname (for example, ip-10-0-56-78) has been found in the HTTP response body. This information might be helpful for further attacks targeting internal systems.
URL	http://hax.tor.hu/peek/?query=query%27+AND+%271%27%3D%271%27++
Parameter	,
Other information	10.0.0.5
Attack	
Solution	Remove the private IP address from the HTTP response body. For comments, use JSP/ASP/PHP comment instead of HTML/JavaScript comment which can be seen by client browsers.
Reference	https://tools.ietf.org/html/rfc1918
WASC Id	13
CWE Id	200
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing

	on the response body, potentially causing the response body to be interpreted
	and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content
	type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/peek/?query=query%27+AND+%271%27%3D%271%27++
Parameter	X-Content-Type-Options
T diditioloi	This issue still applies to error type pages (401, 403, 500, etc) as those pages
Other information	are often still affected by injection issues, in which case there is still
Other information	concern for browsers sniffing pages away from their actual content type. At
	"High" threshold this scanner will not alert on client or server error responses.
Attack	
	Ensure that the application/web server sets the Content-Type header appropriately,
Solution	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and
Solution	modern web browser that does not perform MIME-sniffing at all, or that can
	be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php
	/List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
•	attacks.
URL	http://hax.tor.hu/peek/?query=query%27+AND+%271%27%3D%271%27++
Parameter	X-Frame-Options
Other information	
Attack	
	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set
	on all web pages returned by your site (if you expect the page to be framed
Solution	only by pages on your server (e.g. it's part of a FRAMESET) then you'll want
	to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page
	in supported web browsers).
Deference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
Reference	рх
WASC Id	15
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
	A cookie has been set without the HttpOnly flag, which means that the cookie can
Description	be accessed by JavaScript. If a malicious script can be run on this page then
Description	the cookie will be accessible and can be transmitted to another site. If this
	is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/board/
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
	A cookie has been set without the HttpOnly flag, which means that the cookie can
Description	be accessed by JavaScript. If a malicious script can be run on this page then
Description	the cookie will be accessible and can be transmitted to another site. If this
	is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/haxmin/
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
_on(moalain)	A cookie has been set without the HttpOnly flag, which means that the cookie can
Description	be accessed by JavaScript. If a malicious script can be run on this page then
Description	the cookie will be accessible and can be transmitted to another site. If this
	is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/irc/
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
	http://www.owasp.org/index.php/HttpOnly

Reference	
WASC Id	13
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
Low(Ivicaiaiii)	A cookie has been set without the HttpOnly flag, which means that the cookie can
Description	be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/irc/?query=query%22+AND+%221%22%3D%221%22++
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
LOW(MEGIGITI)	
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/peek/
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/login/
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	X-Content-Type-Options Header Missing
	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted
Description	and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/stats/detect.txt
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
	Ensure that the application/web server sets the Content-Type header appropriately,
Solution	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27+++AND+1%3D1++
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query' AND '1'='1' AND 1=1 ] and [query' AND '1'='1' AND 1=2 ] The parameter value being modified was NOT stripped from the HTML output for the
	purposes of the comparison Data was returned for the original parameter. The

	vulnerability was detected by successfully restricting the data originally
	returned, by manipulating the parameter
Attack	query' AND '1'='1' AND 1=1
	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by
	'?' If the application uses ASP, use ADO Command Objects with strong type
	checking and parameterized queries. If database Stored Procedures can be used,
0-1-4:	use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received
	from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
Reference	on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css+AND+1%3D1++
Parameter	query
i arameter	The page results were successfully manipulated using the boolean conditions [/haxtor.css
	AND 1=1 ] and [/haxtor.css AND 1=2 ] The parameter value being modified
Other information	was NOT stripped from the HTML output for the purposes of the comparison Data
	was returned for the original parameter. The vulnerability was detected by
Attack	successfully restricting the data originally returned, by manipulating the parameter /haxtor.css AND 1=1
Allack	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by
	'?' If the application uses ASP, use ADO Command Objects with strong type
	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create
	dynamic SQL queries using simple string concatenation. Escape all data received
	from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
	on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B+AND+1%3D1++
Parameter	query
	The page results were successfully manipulated using the boolean conditions [javascript:alert(1);
Other information	AND 1=1 ] and [javascript:alert(1); OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data
	was NOT returned for the original parameter. The vulnerability was detected
	by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	javascript:alert(1); OR 1=1
	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type
	checking and parameterized queries. If database Stored Procedures can be used,
0.1.6	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create
Solution	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received
Solution	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create
Solution	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid
Solution	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Solution  Reference	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
Reference WASC Id CWE Id	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
Reference WASC Id CWE Id High(Medium)	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet  19  89  SQL Injection
Reference WASC Id CWE Id High(Medium) Description	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet  19 89  SQL Injection SQL injection may be possible.
Reference WASC Id CWE Id High(Medium)	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet  19  89  SQL Injection

	The page results were successfully manipulated using the boolean conditions [query
	AND 1=1 ] and [query AND 1=2 ] The parameter value being modified was
Other information	NOT stripped from the HTML output for the purposes of the comparison Data
	was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	query AND 1=1
rittaon	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by
	'?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used,
	use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create
	dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Deference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
Reference	on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/peek/?query=query+AND+1%3D1++
Parameter	query
	The page results were successfully manipulated using the boolean conditions [query AND 1=1 ] and [query AND 1=2 ] The parameter value being modified was
Other information	stripped from the HTML output for the purposes of the comparison Data was
	returned for the original parameter. The vulnerability was detected by successfully
Attack	restricting the data originally returned, by manipulating the parameter query AND 1=1
Allack	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by
	'?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used,
	use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create
	dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
	on_Cheat_Sheet
WASC Id CWE Id	19 89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL Parameter	http://hax.tor.hu/irc/?query=query%27+AND+%271%27%3D%271%27++
i aiaiiiciti	query  The page results were successfully manipulated using the boolean conditions [query'
	AND '1'='1' ] and [query' AND '1'='2' ] The parameter value being modified
Other information	was stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by
	successfully restricting the data originally returned, by manipulating the parameter
Attack	query' AND '1'='1'
	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type
	checking and parameterized queries. If database Stored Procedures can be used,
Solution	use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create
Colution	dynamic SQL queries using simple string concatenation. Escape all data received
	from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
	SQL injection may be possible.

Description	
URL	http://hax.tor.hu/links/?query=query+AND+1%3D1++%25
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query AND 1=1 %] and [query AND 1=1 XYZABCDEFGHIJ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data
other information	was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	query AND 1=1 %
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/board/?query=query%27+AND+%271%27%3D%271%27++
Parameter	query
r didirioto.	The page results were successfully manipulated using the boolean conditions [query'
	AND '1'='1' ] and [query' AND '1'='2' ] The parameter value being modified
Other information	was stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by
A44 = -1-	successfully restricting the data originally returned, by manipulating the parameter
Attack	query' AND '1'='1'  Do not trust client side input, even if there is client side validation in place.
Solution	In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
	on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/html/haxtor.css?query=query%22+AND+%221%22%3D%221%22++
Parameter	query The page results were successfully manipulated using the boolean conditions [query"
Other information	AND "1"="1" ] and [query" OR "1"="1" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query" OR "1"="1"
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
	on_Cheat_Sheet
WASC Id	19

CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/stats?query%27+AND+%271%27%3D%271%27++
Parameter	query
i arameter	The page results were successfully manipulated using the boolean conditions [query'
Other information	AND '1'='1' ] and [query' OR '1'='1' ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query' OR '1'='1'
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type  checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/?query=query%25
Parameter	
raiaillelei	query  The page results were successfully manipulated using the boolean conditions [query%]
Other information	and [queryXYZABCDEFGHIJ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	query%
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/peek/?all=1+AND+1%3D1++
Parameter	all
Other information	The page results were successfully manipulated using the boolean conditions [1 AND 1=1 ] and [1 OR 1=1 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned
	for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	1 OR 1=1
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privilege database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.

Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/stats/?query=query+AND+1%3D1++
Parameter	query
	The page results were successfully manipulated using the boolean conditions [query
Other information	AND 1=1 ] and [query AND 1=2 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	query AND 1=1
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
	on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query' AND '1'='1' ] and [query' OR '1'='1' ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query' OR '1'='1'
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention Cheat Sheet
WASC Id	on_crieat_sneet
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/board/?query=query+AND+1%3D1%27+AND+%271%27%3D%271
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query AND 1=1' AND '1'='1] and [query AND 1=1' AND '1'='2] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	query AND 1=1' AND '1'='1
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type  checking and parameterized queries. If database Stored Procedures can be used,  use them. Do *not* concatenate strings into queries in the stored procedure,  or use 'exec', 'exec immediate', or equivalent functionality! Do not create  dynamic SQL queries using simple string concatenation. Escape all data received  from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege

	by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/links/?query=query%22+AND+%221%22%3D%221%22++
Parameter	query
	The page results were successfully manipulated using the boolean conditions [query"
Other information	AND "1"="1" ] and [query" OR "1"="1" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data
o in or in ordination	was NOT returned for the original parameter. The vulnerability was detected
A 44 = -1.	by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query" OR "1"="1"
	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by
	'?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used,
	use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create
	dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
	on_Cheat_Sheet
WASC Id CWE Id	19
	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.  http://hax.tor.hu/irc/?query=query%22+AND+%221%22%3D%221%22++%22+AND+%221%22%3
URL	D%221%22++
Parameter	query
	The page results were successfully manipulated using the boolean conditions [query"
	AND "1"="1" " AND "1"="1" ] and [query" AND "1"="1" " OR "1"="1" ] The parameter value being modified was NOT stripped from the HTML output
Other information	for the purposes of the comparison Data was NOT returned for the original
	parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query" AND "1"="1" " OR "1"="1"
	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type
	checking and parameterized queries. If database Stored Procedures can be used,
Solution	use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create
Coldion	dynamic SQL queries using simple string concatenation. Escape all data received
	from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/sitemap.xml?query=query+ASC+++
Parameter	query  The original page results were successfully replicated using the "ORDER BY" expression
Other information	The original page results were successfully replicated using the "ORDER BY" expression [query ASC ] as the parameter value The parameter value being modified
	was NOT stripped from the HTML output for the purposes of the comparison
Attack	query ASC
	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by
Solution	'?' If the application uses ASP, use ADO Command Objects with strong type
	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure,
	or use 'exec', 'exec immediate', or equivalent functionality! Do not create

dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top\_10\_2010-A1 https://www.owasp.org/index.php/SQL\_Injection\_Preventi Reference on Cheat Sheet WASC Id 19 CWE Id 89 High(Medium) SQL Injection Description SQL injection may be possible. URL http://hax.tor.hu/stats/?graph=3-2&h=400&mode=top5comparison&topfrom=11&w=800 graph Parameter The original page results were successfully replicated using the expression [3-2] Other information as the parameter value The parameter value being modified was stripped from the HTML output for the purposes of the comparison Attack Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do \*not\* concatenate strings into queries in the stored procedure, Solution or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top\_10\_2010-A1 https://www.owasp.org/index.php/SQL\_Injection\_Preventi Reference on Cheat Sheet WASC Id 19 CWE Id Low(Medium) Cookie No HttpOnly Flag A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then Description the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible. URL http://hax.tor.hu/welcome/ Parameter **HAXTOR** Other information Attack Solution Ensure that the HttpOnly flag is set for all cookies. Reference http://www.owasp.org/index.php/HttpOnly 13 WASC Id CWE Id Low(Medium) Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of Description the 'X-XSS-Protection' HTTP response header on the web server URI http://hax.tor.hu/welcome/ X-XSS-Protection Parameter The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example. Other information com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length). Attack Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection Solution HTTP response header to '1'. https://www.owasp.org/index.php/XSS\_(Cross\_Site\_Scripting)\_Prevention\_Cheat\_Sheet https://blog.veraco Reference de.com/2014/03/guidelines-for-setting-security-headers/ WASC Id 14 CWE Id 933 Low(Medium) X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted Description and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing URI http://hax.tor.hu/welcome/ Parameter X-Content-Type-Options

Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php/List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/welcome/
Parameter	X-Frame-Options
Other information	
Attack	
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
WASC Id	15
CWE Id	16

## Step: 1.2) http://hax.tor.hu/

### **Alert Detail**

The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type ofter than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. The properties of the properties	Low(Medium)	X-Content-Type-Options Header Missing
Parameter  X-Content-Type-Options This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Attack  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that thee and user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  Reference  MASC Id 15  CWE Id 16  Low(Medium)  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode-block X-XSS-Protection: 1; mode-block X-XSS-Protection  Attack  Solution  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).	Description	allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content
Other information  This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Attack  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  Reference   http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php //.ist. of_useful_HTTP_headers  WASC Id   15  CWE Id   16  Low(Medium)   Web Browser XSS Protection Not Enabled   Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: HTTP response header is currently supported on Internet Explorer, Chrome and Safari (Webkil), Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Attack  Solution   Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.	URL	http://hax.tor.hu/robots.txt
Other information are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At 'High' threshold this scanner will not alert on client or server error responses.  Attack  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  Reference   http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers  WASC Id 15  CWE Id 15  Low(Medium)   Web Browser XSS Protection Not Enabled    Description   Web Browser XSS Protection Not Enabled    URL   http://hax.tor.hu/sitemap.xml    Parameter   X-XSS-Protection HTTP response header on the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection: 1 TTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit), Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Attack   Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.	Parameter	X-Content-Type-Options
Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  Reference	Other information	are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At
Solution and that it sets the X-Content-Type-Options header to 'nosniff for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  Reference http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php //List_of_useful_HTTP_headers  WASC Id 15  CWE Id 16  Low(Medium) Web Browser XSS Protection Not Enabled  Description Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server  URL http://hax.tor.hu/sitemap.xml  Parameter X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; neode-block X-XSS-Protection: 1; neport-http://www.example.  Other information com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Attack  Solution Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.	Attack	
WASC Id  CWE Id  16  Low(Medium)  Web Browser XSS Protection Not Enabled  Description  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server  URL  http://hax.tor.hu/sitemap.xml  Parameter  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.  Other information  Other information  Com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Attack  Solution  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection  HTTP response header to '1'.	Solution	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
CWE Id  Low(Medium)  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server  URL  http://hax.tor.hu/sitemap.xml  Parameter  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.  Other information  Other information  Other information  Com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Attack  Solution  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.	Reference	
Description  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server  URL  http://hax.tor.hu/sitemap.xml  Parameter  X-XSS-Protection  The X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.  Other information  Other information  Other information  Attack  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection  HTTP response header to '1'.	WASC Id	15
Description  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server  URL  http://hax.tor.hu/sitemap.xml  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.  Other information  Other information  Other information  Other information  Attack  Solution  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.	CWE Id	16
the 'X-XSS-Protection' HTTP response header on the web server  URL http://hax.tor.hu/sitemap.xml  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.  Other information com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Attack  Solution Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.	Low(Medium)	Web Browser XSS Protection Not Enabled
Parameter  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.  Other information  Other information  Other information  Other information  Com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Attack  Solution  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.	Description	
The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.  Other information  Other information  Other information  Com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Attack  Solution  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.	URL	http://hax.tor.hu/sitemap.xml
the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.  Other information  Other information  Other information  Com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Attack  Solution  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.	Parameter	X-XSS-Protection
Solution Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.	Other information	the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
HTTP response header to '1'.	Attack	
https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco	Solution	HTTP response header to '1'.
		https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco

Reference	de.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
URL	the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/sitemap.xml?query=query+AND+1%3D1++
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection
	HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/board/?query=query+AND+1%3D1++
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/quidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/board/?query=query+AND+1%3D1++
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	Forms that the coefficient we have seen to the Co. 1. T
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php/List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/board/?query=query+AND+1%3D1++
Parameter	X-Frame-Options
Other information	
Attack	
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEOPIGIN, otherwise if you payer expect the page to be framed you.
	to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page

	in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
	px
WASC Id CWE Id	15
-	16 Web Browner VCC Protection Not Enghlad
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/login/
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	,,
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
Reference	HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
	de.com/2014/03/guidelines-for-setting-security-headers/
WASC Id CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing  The April MIME Spiffing header X Content Type Options was not set to become This
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/login/
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/login/
Parameter	X-Frame-Options
Other information	
Attack	
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
WASC Id	15
CWE Id	16
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/html/haxtor.css
Parameter	X-Content-Type-Options  This issue still applies to error type pages (404, 403, 500, etc) as these pages.
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At

	"High" threshold this scanner will not alert on client or server error responses.
Attack	
	Ensure that the application/web server sets the Content-Type header appropriately,
Solution	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and
	modern web browser that does not perform MIME-sniffing at all, or that can
	be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php
Reference	/List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/board/
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and
	Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
Deference	HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
Reference	de.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933 V. Contont Type Options Header Missing
Low(Medium)	X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This
Description	allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/board/
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List of useful HTTP headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
URL	attacks. http://hax.tor.hu/board/
Parameter	X-Frame-Options
Other information	
Attack	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set
Solution	on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
WASC Id	15
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
URL	the 'X-XSS-Protection' HTTP response header on the web server
Parameter	http://hax.tor.hu/haxmin/ X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable
2 ommadon	

	the web browser's XSS protection mechanism. The following values would attempt
	to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example. com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection
	HTTP response header is currently supported on Internet Explorer, Chrome and
	Safari (WebKit). Note that this alert is only raised if the response body
	could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
Reference	de.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This
	allows older versions of Internet Explorer and Chrome to perform MIME-sniffing
Description	on the response body, potentially causing the response body to be interpreted
Decemption	and displayed as a content type other than the declared content type. Current
	(early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/haxmin/
Parameter	X-Content-Type-Options
r dramotor	This issue still applies to error type pages (401, 403, 500, etc) as those pages
Other information	are often still affected by injection issues, in which case there is still
Other information	concern for browsers sniffing pages away from their actual content type. At
	"High" threshold this scanner will not alert on client or server error responses.
Attack	
	Ensure that the application/web server sets the Content-Type header appropriately,
Solution	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and
Columbia	modern web browser that does not perform MIME-sniffing at all, or that can
	be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php
	/List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
•	attacks.
URL	http://hax.tor.hu/haxmin/
Parameter	X-Frame-Options
Other information	
Attack	
	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed
0.1.6	only by pages on your server (e.g. it's part of a FRAMESET) then you'll want
Solution	to use SAMEORIGIN, otherwise if you never expect the page to be framed, you
	should use DENY. ALLOW-FROM allows specific websites to frame the web page
	in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
WASC Id	15
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
LOW(MEGIAITI)	
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/peek/
Parameter	X-XSS-Protection
i didilictoi	The X-XSS-Protection HTTP response header allows the web server to enable or disable
	the web browser's XSS protection mechanism. The following values would attempt
	to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.
Other information	com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection
	HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
	could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
Colution	HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
	de.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	Private IP Disclosure
Description	A private IP (such as 10.x.x.x, 172.x.x.x, 192.168.x.x) or an Amazon EC2 private
Description	hostname (for example, ip-10-0-56-78) has been found in the HTTP response body. This information might be helpful for further attacks targeting internal systems.
	2007. The information high be neglected for father attacks targeting internal systems.

	http://hax.tor.hu/peek/
Parameter	
Other information	10.0.0.5
Attack	Demove the private ID address from the HTTD response heady. For comments, use ISD/ASD/DHD
Solution	Remove the private IP address from the HTTP response body. For comments, use JSP/ASP/PHP comment instead of HTML/JavaScript comment which can be seen by client browsers.
Reference	https://tools.ietf.org/html/rfc1918
WASC Id	13
CWE Id	200
Low(Medium)	X-Content-Type-Options Header Missing
	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing
Description	on the response body, potentially causing the response body to be interpreted
Description	and displayed as a content type other than the declared content type. Current
	(early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/peek/
Parameter	X-Content-Type-Options
	This issue still applies to error type pages (401, 403, 500, etc) as those pages
Other information	are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At
	"High" threshold this scanner will not alert on client or server error responses.
Attack	
	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web
Solution	pages. If possible, ensure that the end user uses a standards-compliant and
	modern web browser that does not perform MIME-sniffing at all, or that can
Deference	be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php
Reference	/List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16 V. France, Ontions I loader Not Cot
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/peek/
Parameter	X-Frame-Options
Other information	
Attack	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set
	on all web pages returned by your site (if you expect the page to be framed
Solution	only by pages on your server (e.g. it's part of a FRAMESET) then you'll want
Columnia	to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page
Columbia	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
Reference	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
Reference WASC Id CWE Id	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15
Reference WASC Id	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px 15
Reference WASC Id CWE Id	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing
Reference WASC Id CWE Id	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This
Reference WASC Id CWE Id Low(Medium)	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content
Reference WASC Id CWE Id Low(Medium)  Description	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
Reference WASC Id CWE Id Low(Medium)  Description  URL	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css
Reference WASC Id CWE Id Low(Medium)  Description	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
Reference WASC Id CWE Id Low(Medium)  Description  URL	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css X-Content-Type-Options This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still
Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css X-Content-Type-Options  This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At
Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css X-Content-Type-Options This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still
Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css X-Content-Type-Options  This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately,
Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information  Attack	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if no is set), rather than performing MIME-sniffing.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css X-Content-Type-Options  This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web
Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css X-Content-Type-Options  This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can
Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information  Attack  Solution	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css X-Content-Type-Options  This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information  Attack	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css X-Content-Type-Options  This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can
Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information  Attack  Solution  Reference WASC Id	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css X-Content-Type-Options  This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compiliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing. http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing. http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css X-Content-Type-Options This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing. http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php //List_of_useful_HTTP_headers 15
Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information  Attack  Solution  Reference WASC Id	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/htm//haxtor.css?query=%2Fhaxtor.css X-Content-Type-Options This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_o_useful_HTTP_headers 15 16  Web Browser XSS Protection Not Enabled
Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. (If one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/htm/haxtor.css?query=%2Fhaxtor.css  X-Content-Type-Options  This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers  15  16  Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id Low(Medium)	should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.  http://hax.tor.hu/htm//haxtor.css?query=%2Fhaxtor.css X-Content-Type-Options This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_o_useful_HTTP_headers 15 16  Web Browser XSS Protection Not Enabled

LIDI	
URL Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and
	Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/quidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/irc/
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	Engine that the application hugh converges the Content Type hander appropriately
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Medium(Medium)  Description	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
Description URL Parameter	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks. http://hax.tor.hu/irc/
Description URL Parameter Other information	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/irc/ X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page
Description URL Parameter Other information Attack	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/irc/ X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you
Description URL Parameter Other information Attack  Solution  Reference WASC Id	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/irc/ X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/irc/ X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/irc/ X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15 16  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/irc/ X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15 16  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description URL	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/irc/ X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/irc/ X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15 16  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description URL	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/irc/ X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium) Description URL Parameter	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/irc/ X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter  Other information	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks. http://hax.tor.hu/irc/ X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers). http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15 16  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report—lttp://www.example.com/xss The following values would disable it: X-XSS-Protection: 1; report—lttp://www.example.com/xss The following values would disable it: X-XSS-Protection: 1; report—lttp://www.example.com/xss The following values would disable it: X-XSS-Protection: 1; report—lttp://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter  Other information  Attack Solution  Reference	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/irc/ X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter  Other information  Attack Solution  Reference WASC Id CWE Id Low(Medium)	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/irc/  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection rechanism. The following values would attempt to enable it: X-XSS-Protection: 1; report-http://www.example.com/xss The following values would disable it: X-XSS-Protection: 1; report-http://www.example.com/xss The following values would disable it: X-XSS-Protection: 1; report-http://www.example.com/xss The following values would disable it: X-XSS-Protection: 1; report-http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection  HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracodec.orm/2014/03/guidelines-for-setting-securi
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description URL Parameter  Other information  Attack Solution  Reference	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/irc/ X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/

	The Anti MIME Spiffing header V Centent Type Options was not set to 'nospiff' This
	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted
Description	and displayed as a content type other than the declared content type. Current
	(early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B
Parameter	X-Content-Type-Options
	This issue still applies to error type pages (401, 403, 500, etc) as those pages
Other information	are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	right threshold this scanner will not alert on client of server error responses.
Allack	Facility that the confliction for the constraint of the Constraint Time has decreased by
	Ensure that the application/web server sets the Content-Type header appropriately,
Solution	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the type page to be the form MIME of the confirmal management of the
Reference	be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php
	/List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
Description	attacks.
URL	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B
Parameter	X-Frame-Options
Other information	
Attack	
	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set
	on all web pages returned by your site (if you expect the page to be framed
Solution	only by pages on your server (e.g. it's part of a FRAMESET) then you'll want
Solution	to use SAMEORIGIN, otherwise if you never expect the page to be framed, you
	should use DENY. ALLOW-FROM allows specific websites to frame the web page
	in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
	UA
MASC IA	
WASC Id	15
CWE Id	15 16
	15
CWE Id	15 16
CWE Id Low(Medium)	15 16 Cross-Domain JavaScript Source File Inclusion
CWE Id Low(Medium) Description	15 16 Cross-Domain JavaScript Source File Inclusion The page includes one or more script files from a third-party domain.
CWE Id Low(Medium) Description URL Parameter	15 16  Cross-Domain JavaScript Source File Inclusion The page includes one or more script files from a third-party domain. http://hax.tor.hu/links/
CWE Id Low(Medium) Description URL Parameter Other information	15 16  Cross-Domain JavaScript Source File Inclusion The page includes one or more script files from a third-party domain. http://hax.tor.hu/links/
CWE Id Low(Medium) Description URL Parameter Other information Attack	15 16  Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js
CWE Id Low(Medium) Description URL Parameter Other information	15 16 Cross-Domain JavaScript Source File Inclusion The page includes one or more script files from a third-party domain. http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources
CWE Id Low(Medium) Description URL Parameter Other information Attack Solution	15 16  Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js
CWE Id Low(Medium) Description URL Parameter Other information Attack Solution Reference	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.
CWE Id Low(Medium) Description URL Parameter Other information Attack Solution Reference WASC Id	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.
CWE Id Low(Medium) Description URL Parameter Other information Attack Solution Reference WASC Id CWE Id	15 16  Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.
CWE Id Low(Medium) Description URL Parameter Other information Attack Solution Reference WASC Id	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/  http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled
CWE Id Low(Medium) Description URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/  http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/ X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/ X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/ X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/  X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter  Other information	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/  X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter  Other information	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/ X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter  Other information	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/ X-XSS-Protection THTP response header allows the web server to enable or disable the web browser's XSS protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
CWE Id Low(Medium) Description URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium) Description URL Parameter Other information Attack Solution	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/ X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
CWE Id Low(Medium) Description URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium) Description URL Parameter Other information Attack Solution	The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection metahenism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1, report=http://www.example.com/xss The following values would disable it: X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain. http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/  X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 0 The X-XSS-Protection  HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection  HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  CWE Id  CWE Id  CWE Id  CWE Id  COME ID	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/ X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracodec.com/2014/03/guidelines-for-setting-security-headers/ 14
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server  http://hax.tor.hu/links/ X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode-block X-XSS-Protection: 1; report-http://www.example.com/xss The following values would disable it: X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco de.com/2014/03/guidelines-for-setting-security-headers/ 14 933  X-Content-Type-Options Header Missing
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  CWE Id  CWE Id  CWE Id  CWE Id  COME ID	Cross-Domain JavaScript Source File Inclusion The page includes one or more script files from a third-party domain. http://hax.tor.hu/links/ http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/ X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report-http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco.de.com/2014/03/guidelines-for-setting-security-headers/ 14 333  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/ X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-SSS-Protection TTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco dec.com/2014/03/guidelines-for-setting-security-headers/ 14 933  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  CWE Id  CWE Id  CWE Id  CWE Id  COME ID	Cross-Domain JavaScript Source File Inclusion The page includes one or more script files from a third-party domain. http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/ X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode-block X-XSS-Protection: 1; report-ahttp://www.example.com/xss The following values would disable it: X-XSS-Protection: 1; report-ahttp://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this aller is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/ 14 333  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, bettentally causing the response body to be interpreted
CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)  Description  URL  Parameter  Other information  Attack  Solution  Reference  WASC Id  CWE Id  Low(Medium)	Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  http://hax.tor.hu/links/ http://pagead2.googlesyndication.com/pagead/show_ads.js  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  15 829  Web Browser XSS Protection Not Enabled Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server http://hax.tor.hu/links/ X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection rechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-SSS-Protection TTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).  Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco dec.com/2014/03/guidelines-for-setting-security-headers/ 14 933  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing

	type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/links/
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/links/
Parameter	X-Frame-Options
Other information	
Attack	
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
WASC Id	15
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Description	the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/irc/?query=query%22+AND+%221%22%3D%221%22++
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
Reference	HTTP response header to '1'. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
	de.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/irc/?query=query%22+AND+%221%22%3D%221%22++
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At
Attack	"High" threshold this scanner will not alert on client or server error responses.
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16

Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/irc/?query=query%22+AND+%221%22%3D%221%22++
Parameter	
	X-Frame-Options
Other information	
Attack	Most modern Web browners support the V France Ontiona LITTD bandon France it's act
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
WASC Id	px 15
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/board/?query=query+AND+1%3D1
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	Could potentially contain an 700 payload (with a text-based content type, with a non-zero length).
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
	HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
Reference	de.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/board/?query=query+AND+1%3D1
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
, maon	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Solution Reference	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing. http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
Solution  Reference  WASC Id	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing. http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
Solution  Reference  WASC Id  CWE Id	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing. http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers  15 16
Solution  Reference  WASC Id	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing. http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php/List_of_useful_HTTP_headers  15  16  X-Frame-Options Header Not Set
Solution  Reference  WASC Id  CWE Id	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing. http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php/List_of_useful_HTTP_headers  15  16  X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
Solution  Reference WASC Id CWE Id Medium(Medium)	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers  15  16  X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
Solution  Reference WASC Id CWE Id Medium(Medium) Description URL	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers  15  16  X-Frame-Options Header Not Set X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1
Solution  Reference WASC Id CWE Id Medium(Medium) Description URL Parameter	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers  15  16  X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
Solution  Reference WASC Id CWE Id Medium(Medium) Description URL Parameter Other information	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers  15  16  X-Frame-Options Header Not Set X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1
Solution  Reference WASC Id CWE Id Medium(Medium) Description URL Parameter	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing. http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php//List_of_useful_HTTP_headers  15 16  X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks. http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Solution  Reference WASC Id CWE Id  Medium(Medium)  Description URL  Parameter Other information Attack	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers  15  16  X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to sue SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
Solution  Reference WASC Id CWE Id Medium(Medium) Description URL Parameter Other information Attack  Solution  Reference	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers  15  16  X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
Solution  Reference WASC Id CWE Id  Medium(Medium)  Description URL Parameter Other information Attack  Solution	and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers  15  16  X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/board/?query=query+AND+1%3D1  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to see SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as

Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
•	the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++
Parameter	X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable
Other information	the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/haxmin/?query=guery%27+AND+%271%27%3D%271%27++
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php/List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++
Parameter	X-Frame-Options
Other information	
Attack	
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
WASC Id	px 15
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/html/
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/

MV CC 14	44
WASC Id CWE Id	14 933
Low(Medium)	Web Browser XSS Protection Not Enabled
	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Description	the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/html/?query=query+AND+1%3D1++
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
High(Medium)	Cross Site Scripting (Reflected)
Description	code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology. When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execute code under the local machine zone allowing for system compromise. There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based. Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form, which when posted to the vulnerable site, will mount the attack.  Using a malicious form will oftentimes take place when the vulnerable resource only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript).  Upon clicking on the malicious link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send almost arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash. Persistent attacks occur when the malicious code is submitted
URL	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B
Parameter	query
Other information	
Attack	javascript:alert(1);
Solution	Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. Examples of libraries and framework
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/board/?query=query+AND+1%3D1+AND+1%3D1++
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query AND 1=1 AND 1=1 ] and [query AND 1=1 OR 1=1 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack Solution	query AND 1=1 OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received

	from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
_ ,	SQL injection may be possible.
Description URL	http://hax.tor.hu/html/haxtor.css?query=query+AND+1%3D1++
Parameter	
Parameter	query  The page results were successfully manipulated using the boolean conditions [query
Other information	AND 1=1 ] and [query AND 1=2 ] The parameter value being modified was  NOT stripped from the HTML output for the purposes of the comparison Data  was returned for the original parameter. The vulnerability was detected by
	successfully restricting the data originally returned, by manipulating the parameter
Attack	query AND 1=1
	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by
	'?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create
	dynamic SQL queries using simple string concatenation. Escape all data received
	from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/sitemap.xml?query=query+AND+1%3D1++
Parameter	query
	The page results were successfully manipulated using the boolean conditions [query
Other information	AND 1=1 ] and [query AND 1=2 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	query AND 1=1
	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type
Solution	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create
	dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/links/?query=query+AND+1%3D1++
Parameter	query
	The page results were successfully manipulated using the boolean conditions [query AND 1=1 ] and [query OR 1=1 ] The parameter value being modified was
Other information	NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected
	NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Other information  Attack  Solution	NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected

use them. Do \*not\* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top\_10\_2010-A1 https://www.owasp.org/index.php/SQL\_Injection\_Preventi Reference on\_Cheat\_Sheet WASC Id 19 CWE Id 89 High(Medium) SQL Injection Description SQL injection may be possible. http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27+--+%22+AND+%221%2 **URL** 2%3D%221%22+--+ Parameter auerv The page results were successfully manipulated using the boolean conditions [query' AND '1'='1' -- " AND "1"="1" -- ] and [query' AND '1'='1' -- " AND "1"="2" -- ] The parameter value being modified was NOT stripped from the HTML output Other information for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter Attack query' AND '1'='1' -- " AND "1"="1" --Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do \*not\* concatenate strings into queries in the stored procedure, Solution or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top\_10\_2010-A1 https://www.owasp.org/index.php/SQL\_Injection\_Preventi Reference on\_Cheat\_Sheet WASC Id 19 CWE Id 89 High(Medium) SQL Injection Description SQL injection may be possible. URL http://hax.tor.hu/stats/detect.txt?query=query+AND+1%3D1 Parameter query The page results were successfully manipulated using the boolean conditions [query AND 1=1] and [query OR 1=1] The parameter value being modified was NOT stripped Other information from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query OR 1=1 Attack Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do \*not\* concatenate strings into queries in the stored procedure, Solution or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top\_10\_2010-A1 https://www.owasp.org/index.php/SQL\_Injection\_Preventi Reference on Cheat Sheet WASC Id 19 CWE Id 89 High(Medium) SQL Injection Description SQL injection may be possible. URL http://hax.tor.hu/?query=query+AND+1%3D1+--+ Parameter query The page results were successfully manipulated using the boolean conditions [query AND 1=1 -- ] and [query OR 1=1 -- ] The parameter value being modified was Other information stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter Attack query OR 1=1 --

Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create
0.00.01	dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/board/?query=query%27+AND+%271%27%3D%271%27++
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query' AND '1'='1' ] and [query' AND '1'='2' ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by
	successfully restricting the data originally returned, by manipulating the parameter
Attack	query' AND '1'='1'  Do not trust client side input, even if there is client side validation in place.
	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type  checking and parameterized queries. If database Stored Procedures can be used,  use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
0 (	
Description	SQL injection may be possible.
Description URL	SQL injection may be possible.  http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++
Description URL Parameter	http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++
URL	
URL	http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++ query
URL Parameter	http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++ query The page results were successfully manipulated using the boolean conditions [query OR 1=1 ] and [query AND 1=2 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query OR 1=1
URL Parameter Other information Attack	http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query OR 1=1 ] and [query AND 1=2 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure,
URL Parameter Other information	http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++ query The page results were successfully manipulated using the boolean conditions [query OR 1=1 ] and [query AND 1=2 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used,
URL Parameter Other information Attack	http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query OR 1=1 ] and [query AND 1=2 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
URL Parameter  Other information  Attack  Solution  Reference WASC Id	http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query OR 1=1] and [query AND 1=2] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
URL Parameter Other information Attack Solution	http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query OR 1=1 ] and [query AND 1=2 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
URL Parameter  Other information  Attack  Solution  Reference WASC Id	http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query OR 1=1 ] and [query AND 1=2 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database user sons teliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet
URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id	http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query OR 1=1 ] and [query AND 1=2 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database user sossible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19
URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id  High(Medium)	http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query OR 1=1 ] and [query AND 1=2 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query OR 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  SQL Injection
URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id  High(Medium) Description	http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query OR 1=1] and [query AND 1=2] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do 'not' concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet  19  SQL Injection SQL injection may be possible.
URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id  High(Medium) Description URL	http://hax.tor.hu/robots.txt?query=query+OR+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [query OR 1=1 ] and [query AND 1=2 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter query OR 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do 'not' concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'do-worer' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19 89  SQL Injection SQL injection may be possible. http://hax.tor.hu/login/?query=query%22+AND+%221%22%3D%221%22++

	was returned for the existing a respective. The configuration was detected by
	was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	query" AND "1"="1"
	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used,
Solution	use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/stats?query=query%22+AND+%221%22%3D%221%22++
Parameter	query
	The page results were successfully manipulated using the boolean conditions [query"
Other information	AND "1"="1" ] and [query" OR "1"="1" ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query" OR "1"="1"
	Do not trust client side input, even if there is client side validation in place.
Solution	In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/peek/?all=1%27+AND+%271%27%3D%271
Parameter	all
Other information	The page results were successfully manipulated using the boolean conditions [1' AND '1'='1] and [1' AND '1'='2] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	1' AND '1'='1
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create
Column	dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/irc/?query=query%22+AND+%221%22%3D%221%22+++OR+1%3D1++
Parameter	query

	The second secon
	The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" OR 1=1 ] and [query" AND "1"="1" AND 1=2 ] The parameter
Other information	value being modified was stripped from the HTML output for the purposes of
	the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned,
	by manipulating the parameter
Attack	query" AND "1"="1" OR 1=1
	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by
	'?' If the application uses ASP, use ADO Command Objects with strong type
	checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create
	dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
	on_Cheat_Sheet
WASC Id CWE Id	19 89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/html?query=query%25
Parameter	query
. G.Getc.	The page results were successfully manipulated using the boolean conditions [query%]
Other information	and [query%] The parameter value being modified was NOT stripped from the
Other information	HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving
	more data than originally returned, by manipulating the parameter
Attack	query%
	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by
	'?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used,
	use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received
	from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/board/?query=query%25
Parameter	query
	The page results were successfully manipulated using the boolean conditions [query%] and [query%] The parameter value being modified was NOT stripped from the
Other information	HTML output for the purposes of the comparison Data was NOT returned for the
	original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query%
	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by
	'?' If the application uses ASP, use ADO Command Objects with strong type
	checking and parameterized queries. If database Stored Procedures can be used,
Solution	use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create
	dynamic SQL queries using simple string concatenation. Escape all data received
	from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Deference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
Reference	on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection

Description	SQL injection may be possible.
URL	http://hax.tor.hu/irc/?query=query%25
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query%] and [query%] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query%
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
Reference	on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B+ASC+++
Parameter	query
Other information	The original page results were successfully replicated using the "ORDER BY" expression [javascript:alert(1); ASC ] as the parameter value The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison
Attack	javascript:alert(1); ASC
Solution	In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
WASC Id	on_Cheat_Sheet 19
CWE Id	89
Low(Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/welcome/
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
,	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Description	the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/welcome/
Parameter Other information	X-XSS-Protection The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and
Attack	Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).

Reference  Interpolative owasp only indice, pre/XSS_(Cross_Site_Soriging). Prevention_Cheal_Sheet https://tidog.veraco WASC Id  4  COWE Id  33  Low(Medium)  X=Content=Type=Options Header Missing The Anti-Miss Site of the Prevention of the Prevent	Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
## ACONTENT OF TOTAL AND STATE OF THE PROPERTY	Coldion	HTTP response header to '1'.  https://www.owasp.org/index.php/YSS_(Cross_Site_Scripting)_Provention_Cheat_Sheet https://blog.verace.
WASC Id   433   Section	Reference	
Security	WASC Id	•
The Anti-MME-Suffing header X-Content-Type-Options was not set to insentif. This allows older versions of internet Exporer and Chrome to perform MME-suffing on the response body to be interpreted and displayed as a control type other than the destand content type. Current (early, 2014) and slapsey versions of Friedox will use the destand content (see 1) and supplied as a control type of the than the destand content (see 1) and supplied as a control type of Friedox will use the destand content (see 1) and the supplied of Friedox will use the destand content (see 1) and the supplied of Friedox will use the destand content (see 1) and the supplied of Friedox will use the destand content (see 1) and the supplied of Friedox will use the destand content (see 2) and the supplied of Friedox will use the destand content (see 2) and the supplied of Friedox will use the destand content (see 2) and the supplied of Friedox will use the destand content (see 2) and the supplied of Friedox will use the destand content (see 2) and the supplied of Friedox will use the supplied to Friedox will use the supplied of Friedox will use the suppliedox will use the supplie	CWE Id	933
The Anti-MIME-Suffing header X-Content-Type-Options was not set to hossiff. This allows older versions of interiore Explorer and Chrome to perform MIME-suffing on the response body, potentially causing the response body to be interpreted and deplayed as a control type other than the declared content type. Current (serby 2014) and legacy versions of irristors will use the declared content type. Current (serby 2014) and legacy versions of irristors will use the declared content type. Current (serby 2014) and legacy versions of irristors will use the declared content (serby 2014) and serby and the declared content (serby 2014). The serby and the declared content (serby 2014) and the serby and serby a	Low(Medium)	X-Content-Type-Options Header Missing
allows older versions of Internet Explorer and Chrome to perform MIME-antifling on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and liquary response to First four the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and liquary response of First own thus the declared content type. Current (early 2014) and liquary response of First own thus the thorough the property of the prop		
and displayed as a content type other than the declared content type. Current (early 2014) and legal year versions of Fireds will use the declared content to type (if one is self, rather than performing MIME-sniffing).  URL http://www.declared.com/mimes/mime		allows older versions of Internet Explorer and Chrome to perform MIME-sniffing
(certy 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-antiffig.  INTELLIPENTAL IOL INLIVENCEMP  Parameter  X-Content-Type-Options This issue still applies to error type pages (401, 403, 50, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers suffing pages away from their actual content type. At 4 High 'Interhold this scanner will not alter on client or server error responses.  Attack  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to nountif for all web content with the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to nountif for all web content web browers that does not perform MIME-sniffig at all or that can be directed by the web application/web server to not perform MIME-sniffig at all or that can be directed by the web application/web server to not perform MIME-sniffig at all or that can be directed by the web application/web server to not perform MIME-sniffig at all or that can be directed by the web application/web server to not perform MIME-sniffig at all or that can be directed by the web application/web server to not perform MIME-sniffig at all or that can be directed by the web application/web server to perform MIME-sniffig at all or that can be directed by the web application web to server the does not perform MIME-sniffig at all or that can be directed by the web application web to server the does not perform MIME-sniffig at all or that can be directed by the web application web application web application web application web application web application web applications web application web applications web applications web applications web applications web applications at a can be applicated by a performed and application web applications and application web applications.  Solution the specific of the performan	Description	
URL http://max.for.hu/welcome/ Parameter  X-Content-Type-Options This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers smiling pages away from their actual content type. At 'High' threshold this scanner will not alert on client or server error responses.  Attack  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to nosalff for all web pages. If possible, ensure that the end user uses a standards-complant and modern web browser that does not perform MMRE-sniffing at 8 du or that can need to modern web browser that does not perform MMRE-sniffing at 8 du or that can need the content of the		, , , , , , , , , , , , , , , , , , ,
Parameter  X-Content-Type-Cytions This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by rigiction issues, in which case there is still concern for browsers smilling pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.  Attack  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to reserve reror responses.  Solution  Solution  Reference  Application web browser that does not perform MMR—siming at all, or that can be directed by the web application/web server to not perform MMR—siming at all, or that can be directed by the web application/web server to not perform MMR—siming at all, or that can be directed by the web application/web server to not perform MMR—siming at all, or that can be directed by the web application/web server to not perform MMR—siming at all, or that can be directed by the web application/web server to not perform MMR—siming at all, or that can be directed by the web application/web server to not perform MMR—siming at all, or that can be directed by the web application/web server to not perform MMR—siming at all, or that can be directed by the web application/web server to not perform MMR—siming at all, or that can be directed by the web application/web server to not perform MMR—siming at all, or that can be directed by the web application/web application web application.  Reference  WASC Id 15  Cover Id 15  Low (Medium)  Cookie		
This issue still applies to error type pages (401, 403, 500, etc) as those pages are clem still affected by rijection issues, in which case there is still concern for browsers sniffing pages away from their actual content type, At 'High'' threshold this scanner will not all ort on client or server error responses.  Attack  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to nosmiff for all web pages. If possible, ensure that the on user uses at standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing at all, or that can be directed by the web application server to not perform MIME-sniffing at all, or that can be directed by the web application server to not perform MIME-sniffing at all, or that can be directed by the web application server to not perform MIME-sniffing at all, or that can be directed by the web application server to not perform MIME-sniffing at all, or that can be directed by the web application server (e.g. the specified to the MIME-sniffing at all, or that can be directed by the web application server (e.g. the specified perform MIME-sniffing at all, or that can be directed by the web application of the specified performs the specified perfo	URL	http://hax.tor.hu/welcome/
are often still affected by injection issues, in which case there is still content for townsers snifting pages away from their actual content type. At "High" threshold this scanner will not allert on client or server error responses.  Attack  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-antifing at all, or that can be directed by the web application/web server to not perform MIME-antifing at all, or that can be directed by the web application/web server to not perform MIME-antifing at all, or that can be directed by the web application/web server to not perform MIME-antifing at all, or that can be directed by the web application/web server to not perform MIME-antifing at all, or that can be directed by the web application/web server to not perform MIME-antifing at all, or that can be directed by the web application/web server to not perform MIME-antifing.  Reference  WASC Id 15  Carbon Web Intervention Attack  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it pand to a PAMESET) then you'll want be provided used DENY ALL DW-From allows specific websites to frame the web page in supported web browsers).  Reference  px could use DENY ALL DW-From allows specific websites to frame the web page in supported web browsers).  Reference  px could use DENY ALL DW-From allows specific websites to frame the web page in supported web browsers).  Reference  px could use DENY ALL DW-From allows specific websites to frame the web page.  In this provider is a page of the page of the page to be framed. You are seen the page of	Parameter	X-Content-Type-Options
Orneer for browsers suiffing pages away from their actual content type. At High-threshold tils scanner will not allert on client or server error responses.  Attack  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to rosentif for all web pages. If possible, on source that the one user uses at sendands-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  Reference		
Hight threshold this scanner will not alert on client or server error responses.	Other information	
Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header for noanfilf for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  Reference http://msdn.microsoft.com/en-usibitrayri/erg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php //List_of_useful_HTTP_headers  WASC Id 15  CWE Id 16  Medium(Medium) X-Frame-Options Header Not Set  X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  URL http://max.tor.hu/welcome/  Parameter  X-Frame-Options Header Not Set  X-Frame-Options Deader is not included in the HTTP response to protect against 'ClickJacking' attacks.  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed on all web pages returned by your site (if you expect the page to be framed on all web pages returned by your site (if you expect the page to be framed on all web pages returned by your site (if you expect the page to be framed on all web pages in supported web browsers). ALLOW-RROM allows specific websites to frame the web page in supported web browsers. ALLOW-RROM allows specific websites to frame the web page in supported web browsers.  Parameter  Acodie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript, If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL  http://max.tor.hu/links/?query=query+AND+1%3D1+-+  Parameter  Other information  Actack  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the appl		
and that it sets the X-Content-Type-Options header to 'nosniff for all web Solution pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  Reference   http://msdn.microsoft.com/en-us/library/fe/gg622941%28v-vs.85%29.aspx https://www.owasp.org/index.php //list_of_useful_HTTP_headers  WASC Id 15   CWE Id 16   Medium(Medium)   X-Frame-Options Header Not Set   X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  URL   http://msa.tor.hulwelcome/ Parameter   X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  URL   http://msa.tor.hulwelcome/ Parameter   X-Frame-Options   X-Frame-Options   Attack   Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed on all web pages returned by your site (if you expect the page to be framed) should use DENY-ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  Reference   http://ms.ach.com/b/leinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px WASC Id 15   CWE Id 16   Low(Medium)   Cookie No HttpOnly Flag   A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a mallocious script can be run on this page then the cookie will be accessible and cancellate it. If this is a session cookie then session highding may be possible.  URL http://max.tor.hulmiks/?query=query=AND+1%3D1+++   Parameter   HAXTOR   URL http://max.tor.hulmiks/?query=query=AND+1%3D1+++   Haxtor   Haxtor   Haxtor   Haxtor   URL http://max.tor.hulmiks/?query=query=AND+1%3D1+++   Haxtor   Haxtor   Haxtor   URL http://max.tor.hulmiks/?query=query=AND+1%3D1+++   Haxtor   Haxtor   Haxtor   Haxtor	Attack	
pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.  Reference  http://msch.microsoft.com/en-us/library/le/g622941%28vevs.85%29.aspx https://www.owasp.org/index.php //List_of_useful_HTTP_headers  WASC Id 15  CWE Id 16  Medium (Medium)  X-Frame-Options Header Not Set  V-Frame-Options Header Not Set  V-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  URL http://hax.tor.hu/welcome/  X-Frame-Options  Other information  Attack  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  Reference  px  WASC Id 15  CWE Id 16  Low(Medium)  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly Ifag, which means that the cookie can be accessed by Java-Script. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session highsiding may be possible.  URL  http://hax.tor.hu/links/?queny=queny+AND+1%3D1+-+  Parameter  Other information  Attack  Solution  Ensure that the HttpOnly Iflag is set for all cookies.  Reference  http://www.owasp.org/index.php/HttpOnly  WASC Id 16  Low(Medium)  Cross-Domain JavaScript! Source File Inclusion  The page includes one or more script file from a third-party domain.  http://hax.tor.hu/links/?queny=queny+AND+1%3D1+-+  Parameter  Other information  Attack  Solution  Ensure JavaScript source lies are loaded from only trusted sources, and the sources can't be controlled by end users of the application.		
modem web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing. http://msdn.microsoft.com/n-us/library/le/g622941%28vevs.85%29.aspx.https://www.owasp.org/index.php //List_of_useful_HTTP_headers  WASC Id 15  WESC Id 16  Medium(Medium)  X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  URL http://hax.tor.hu/welcome/  Parameter X-Frame-Options  Other information  Attack  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY_ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  Reference  px  WASC Id 15  CWE Id 16  Low(Medium)  Cookie No HttpOnly Flag  A cookie No HttpOnly Flag  A cookie No HttpOnly Flag  A cookie has been set without the HttpOnly lfig, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie with sea seasion cookie then seasion highciking may be possible.  URL http://max.tor.hu/links/?query=query-AND+1%3D1+++  Parameter  HAXTOR  Other information  Attack  Solution  Ensure that the HttpOnly lfag is set for all cookies.  Reference  http://www.owasp.org/index.php/HttpOnly  WASC Id 16  Low(Medium)  Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query-AND+1%3D1+++  Parameter  http://hayascript source lifes are loaded from only trusted sources, and the sources  can't be controlled by end users of the application.	Solution	,
be directed by the web application/web server to not perform MIME-sniffing.  Reference  http://msd.m.inscionsoft.com/en-ushibray/rie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php //List_of_useful_HTTP_headers  WASC Id 15  CWE Id 16  Medium(Medium)  X-Frame-Options Header is not included in the HTTP response to protect against 'ClickJacking' attacks.  URL  http://hax.tor.huwelcome/  Parameter  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  URL  http://hax.tor.huwelcome/  Attack  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMECRIGIN, otherwise if you never septet the page to be framed, you should use DENY, ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  Reference  px  WASC Id 15  CWE Id 16  Low(IMedium)  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly (flag, which means that the cookie can be accessed by JavaScript, if a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session highcking may be possible.  URL  http://hax.tor.hullinks/?query=query+AND+1%3D1+-++  HAXTOR  Other information  Attack  Solution  Ensure that the HttpOnly flag is set for all cookies.  Reference  http://www.owasp.org/index.php/httpOnly  WASC Id 13  COWE Id 16  Low(Medium)  Cross-Dormain JavaScript Source File Inclusion  Pescription  the page includes one or more script files from a third-party domain.  Http://hax.tor.hullinks/?query=query+AND+1%3D1++  Parameter  http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.	Solution	
WASC Id 15  CWE Id Medium(Medium) X-Frame-Options Header Not Set  Wedium(Medium) X-Frame-Options Header Not Set  X-Frame-Options Header is not included in the HTTP response to protect against 'ClickJacking' attacks.  URL http://hax.tor.hu/wilcome/  Parameter X-Frame-Options  Other information  Attack Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed on you by pages on your server (ie., if you expect the page to be framed you should use DENY ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  Reference http://blogs.msdn.com/b/einternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  WASC Id 15  CWE Id 16  Low(Medium) Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript, If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  HAXTOR  Other information  Attack  Solution Ensure that the HttpOnly flag is set for all cookies.  Reference http://www.ovasp.org/index.php/HttpOnly  WASC Id 13  CWE Id 18  Low(Medium) Cross-Domain JavaScript Source File Inclusion  Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.		
MASC Id 15 CWE Id 16 Medium (Medium)  X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  URL http://hax.tor.hu/welcome/  Parameter X-Frame-Options  Other information  Attack  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g., it's part of a FRAMESET) then you'll want to use SAME/CRIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  Reference  http://bogs.msdn.com/b/einternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  WASC Id 15  CWE Id 16  LOW(Medium)  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript, if a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session highsicking may be possible.  URL  http://hax.tor.hulinks/?query=query+AND+1%3D1++  HAXTOR  Other information  Attack  Solution  Ensure that the HttpOnly flag is set for all cookies.  Reference  http://www.owasp.org/index.php/HttpOnly  WASC Id 13  CWSC Id 16  LOW(Medium)  Cross-Domain JavaScript Source File Inclusion  Description  The page includes one or more script files from a third-party domain.  URL  http://hax.tor.hulinks/?query=query+AND+1%3D1++  Parameter  http://hax.tor.h	Reference	
CWE Id  Medium(Medium)  X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  URL  Parameter  Other information  Attack  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed) only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  Reference  http://holgs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  WASC Id  15  CWE Id  Low(Medium)  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL  http://hax.tor.hu/links/?query=query+AND+1%3D1++  HAXTOR  Other information  Attack  Solution  Ensure that the HttpOnly flag is set for all cookies.  Reference  http://www.owasp.org/index.php/HttpOnly  WASC Id  13  CWE Id  Low(Medium)  Cross-Domain JavaScript Source File Inclusion  Description  the page includes one or more script flies from a third-party domain.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter  Other information  Attack  Solution  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.		
Medium(Medium   X-Frame-Options Header Not Set		
Description    X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.		
URL http://hax.tor.hu/welcome/ Parameter X-Frame-Options Other information Attack  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAME-ORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOV-FROM allows specific websites to frame the web page in supported web browsers).  Reference http://blogs.msdn.com/b/leinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px WASC Id 15 CWE Id 16 LOW(Medium) Cookie No HttpOnly Flag A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript if a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Parameter HAXTOR Other information Attack Solution Ensure that the HttpOnly flag is set for all cookies. Reference http://www.owasp.org/index.php/HttpOnly WASC Id 13 CWE Id 16 LOW(Medium) Cross-Domain JavaScript Source File Inclusion Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Parameter http://www.owasp.org/squery-query+AND+1%3D1++ Haxtor Solution The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Haxtor Solution The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Haxtor Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.	ivieaium(ivieaium)	· · · · · · · · · · · · · · · · · · ·
URL http://hax.tor.hu/welcome/ Parameter	Description	
Parameter Other information Attack  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY, ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  Reference	URI	
Other information Attack  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  Reference  px  WASC Id  15  CWE Id  16  Low(Medium)  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL  http://hax.tor.hu/links/?query=query+AND+1%3D1++  HAXTOR  Other information  Attack  Solution  Ensure that the HttpOnly flag is set for all cookies.  Reference  http://www.owasp.org/index.php/HttpOnly  WASC Id  13  CWE Id  16  Low(Medium)  Cross-Domain JavaScript Source File Inclusion  Description  The page includes one or more script files from a third-party domain.  URL  http://hax.tor.hu/links/?query=query+AND+1%3D1++  http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter  http://hax.tor.hu/links/?query=query+AND+1%3D1++  Flag and the flag and the flag and the sources and the sources can't be controlled by end users of the application.  Reference  WASC Id  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.		
Attack  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g., it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  Reference  http://blogs.msdn.com/b/einternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  WASC Id  15  CWE Id  16  Low(Medium)  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL  http://hax.tor.hullinks/?query=query+AND+1%3D1++ Parameter  Other information  Attack  Solution  Ensure that the HttpOnly flag is set for all cookies.  Reference  http://www.owasp.org/index.php/HttpOnly  WASC Id  16  Low(Medium)  Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  URL  http://hax.tor.hullinks/?query=query+AND+1%3D1++  Parameter  http://hax.tor.hullinks/?query=query+AND+1%3D1++  Parameter  http://hax.tor.hullinks/?query=query+AND+1%3D1++  Parameter  http://hax.tor.hullinks/?query=query+AND+1%3D1++  Parameter  http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.		
Solution on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. if is part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  Reference http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  WASC Id 15  CWE Id 16  Low(Medium) Cookie No HttpOnly Flag A cookie No HttpOnly Flag A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session highsking may be possible.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter HAXTOR  Attack Solution Ensure that the HttpOnly flag is set for all cookies.  Reference http://www.owasp.org/index.php/HttpOnly  WASC Id 13  CWE Id 16  Low(Medium) Cross-Domain JavaScript Source File Inclusion  The page includes one or more scrip files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Cheri information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.		
Solution only by pages on your server (e.g., it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  Reference http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px WASC Id 15 CWE Id 16 Low(Medium) Cookie No HttpOnly Flag A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ HAXTOR Other information Attack Solution Ensure that the HttpOnly flag is set for all cookies. Reference http://www.owasp.org/index.php/HttpOnly WASC Id 13 CWE Id 16 Low(Medium) Cross-Domain JavaScript Source File Inclusion Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js Other information Attack Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.		Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set
to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  Reference http://blogs.msdn.com/b/leinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  WASC Id 15  CWE Id 16  Low (Medium) Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  HAXTOR  Other information  Attack Solution Ensure that the HttpOnly flag is set for all cookies. Reference http://www.owasp.org/index.php/HttpOnly  WASC Id 13  CWE Id 16  Low (Medium) Cross-Domain JavaScript Source File Inclusion  The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.		
should use DENY, ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  Reference http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  WASC Id 15  CWE Id 16  Low(Medium) Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter HAXTOR  Other information  Attack Solution Ensure that the HttpOnly flag is set for all cookies.  Reference http://www.owasp.org/index.php/HttpOnly  WASC Id 13  CWE Id 16  Low(Medium) Cross-Domain JavaScript Source File Inclusion  Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.	Solution	
Reference http://blogs.msdn.com/b/leinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px WASC Id 15 CWE Id 16 Low(Medium) Cookie No HttpOnly Flag A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Parameter HAXTOR Other information Attack Solution Ensure that the HttpOnly flag is set for all cookies. Reference http://www.owasp.org/index.php/HttpOnly WASC Id 13 CWE Id 16 Low(Medium) Cross-Domain JavaScript Source File Inclusion Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js Other information Attack Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.		
WASC Id 15  CWE Id 16  Low(Medium) Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter HAXTOR  Other information  Attack Solution Ensure that the HttpOnly flag is set for all cookies.  Reference http://www.owasp.org/index.php//HttpOnly  WASC Id 13  CWE Id 16  Low(Medium) Cross-Domain JavaScript Source File Inclusion  Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.		
WASC Id CWE Id 16 Low(Medium) Cookie No HttpOnly Flag A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Parameter Other information Attack Solution Ensure that the HttpOnly flag is set for all cookies. Reference http://www.owasp.org/index.php/HttpOnly WASC Id 13 CWE Id 16 Low(Medium) Cross-Domain JavaScript Source File Inclusion Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Parameter Other information Attack Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.	Reference	
CWE Id  Low(Medium)  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter HAXTOR  Other information  Attack  Solution Ensure that the HttpOnly flag is set for all cookies.  Reference http://www.owasp.org/index.php/HttpOnly  WASC Id 13  CWE Id 16  Low(Medium)  Cross-Domain JavaScript Source File Inclusion  Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.	MV8C IA	
Low(Medium)  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Parameter HAXTOR  Other information  Attack Solution Ensure that the HttpOnly flag is set for all cookies.  Reference http://www.owasp.org/index.php/HttpOnly  WASC Id 13  CWE Id 16  Low(Medium)  Cross-Domain JavaScript Source File Inclusion  Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.		
A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter HAXTOR  Other information  Attack  Solution Ensure that the HttpOnly flag is set for all cookies.  Reference http://www.owasp.org/index.php/HttpOnly  WASC Id 13  CWE Id 16  Low(Medium) Cross-Domain JavaScript Source File Inclusion  Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.		
be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter HAXTOR  Other information  Attack Solution Ensure that the HttpOnly flag is set for all cookies.  Reference http://www.owasp.org/index.php/HttpOnly  WASC Id 13  CWE Id 16  Low(Medium) Cross-Domain JavaScript Source File Inclusion  Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference  WASC Id 15	LOW(MEGIGITI)	1 7 5
the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter HAXTOR  Other information  Attack  Solution Ensure that the HttpOnly flag is set for all cookies.  Reference http://www.owasp.org/index.php/HttpOnly  WASC Id 13  CWE Id 16  Low(Medium) Cross-Domain JavaScript Source File Inclusion  Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference  WASC Id 15	D : "	
URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Parameter HAXTOR  Other information Attack Solution Ensure that the HttpOnly flag is set for all cookies. Reference http://www.owasp.org/index.php/HttpOnly WASC Id 13 CWE Id 16  Low(Medium) Cross-Domain JavaScript Source File Inclusion  Description The page includes one or more script files from a third-party domain. URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information Attack Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference WASC Id 15	Description	
Parameter Other information Attack Solution Ensure that the HttpOnly flag is set for all cookies. Reference http://www.owasp.org/index.php/HttpOnly WASC Id 13 CWE Id 16  Low(Medium) Cross-Domain JavaScript Source File Inclusion Description The page includes one or more script files from a third-party domain. URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js Other information Attack Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference WASC Id 15		
Other information Attack Solution	-	
Attack Solution Ensure that the HttpOnly flag is set for all cookies.  Reference http://www.owasp.org/index.php/HttpOnly WASC Id 13 CWE Id 16  Low(Medium) Cross-Domain JavaScript Source File Inclusion  Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference WASC Id 15		HAXTOR
Solution Ensure that the HttpOnly flag is set for all cookies.  Reference http://www.owasp.org/index.php/HttpOnly  WASC Id 13  CWE Id 16  Low(Medium) Cross-Domain JavaScript Source File Inclusion  Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference  WASC Id 15		
Reference http://www.owasp.org/index.php/HttpOnly WASC Id 13 CWE Id 16  Low(Medium) Cross-Domain JavaScript Source File Inclusion  Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference  WASC Id 15		Francisco that the Little Only flow is not feer all and in
WASC Id  CWE Id  16  Low(Medium)  Description  The page includes one or more script files from a third-party domain.  URL  http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter  Other information  Attack  Solution  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference  WASC Id  15		
CWE Id 16  Low(Medium) Cross-Domain JavaScript Source File Inclusion  Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference  WASC Id 15		
Low(Medium)  Description  The page includes one or more script files from a third-party domain.  URL  http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter  http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference  WASC Id  15		
Description The page includes one or more script files from a third-party domain.  URL http://hax.tor.hu/links/?query=query+AND+1%3D1++  Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference  WASC Id  The page includes one or more script files from a third-party domain.  It is page includes one or more script files from a third-party domain.  It is page includes one or more script files from a third-party domain.  It is page includes one or more script files from a third-party domain.  It is page includes from a third-party domain.  It is page		
URL http://hax.tor.hu/links/?query=query+AND+1%3D1++ Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js Other information Attack Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference WASC Id 15		·
Parameter http://pagead2.googlesyndication.com/pagead/show_ads.js  Other information  Attack  Solution Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference  WASC Id 15	•	
Other information Attack  Solution  Reference  WASC Id  Attack  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.		
Attack  Solution  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference  WASC Id  15		nttp://pageauz.googiesyndication.com/pagead/show_ads.js
Solution  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.  Reference  WASC Id  15		
Can't be controlled by end users of the application.  Reference  WASC Id  15		Ensure JavaScript source files are loaded from only trusted sources, and the sources
Reference WASC Id 15	Solution	
WASC Id 15	Reference	,
		15
CWE Id 829	CWE Id	829
Low(Medium) Web Browser XSS Protection Not Enabled	Low(Medium)	Web Browser XSS Protection Not Enabled
Web Proyect VCC Protection is not applied or in displied by the configuration of		
the 'X-XSS-Protection' HTTP response header on the web server	Describitori	the 'X-XSS-Protection' HTTP response header on the web server
http://hax.tor.hu/links/?query=query+AND+1%3D1++		http://hax.tor.hu/links/?query=query+AND+1%3D1++

URL	
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	Could potentially Contain an ASS payload (with a text-based Content type, with a non-zero length).
	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
Solution	HTTP response header to '1'.  https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
Reference	de.com/2014/03/quidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/links/?query=query+AND+1%3D1++
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	Faces that the application has been applied to the Control Translation to t
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php
WASC Id	/List_of_useful_HTTP_headers 15
CWE Id	16
CWE IU	10
NA - diverse (NA - diverse)	
Medium(Medium)	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
Medium(Medium)  Description	X-Frame-Options Header Not Set
	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
Description	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
Description URL	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++
Description URL Parameter	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++
Description URL Parameter Other information	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page
Description URL Parameter Other information Attack	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
Description URL Parameter Other information Attack  Solution  Reference	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
Description URL Parameter Other information Attack Solution	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15
Description URL Parameter Other information Attack  Solution  Reference WASC Id	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/  HAXTOR
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack Solution	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/  HAXTOR  Ensure that the HttpOnly flag is set for all cookies.
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack Solution Reference	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15 16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/  HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id CWE Id CWE Id CWE ID COMPANY OF THE PARAMETER OF THE PARAM	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible. http://hax.tor.hu/  HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly  13  16
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack Solution Reference WASC Id	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  15  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/  HAXTOR  Ensure that the HttpOnly flag is set for all cookies. http://www.owasp.org/index.php/HttpOnly  13  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is the cookie will be accessible and can be transmitted to another site. If this is the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie can be transmitted to another site. If this is the cookie can be transmitted to another site. If this is the cookie can be transmitted to another site. If this is the cookie can be transmitted to another site. If this is the cookie can be transmitted to another site. If this is the cookie can be transmitted to another site. If this is the cookie was calculated and the transmitted to another site. If this
Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.  http://hax.tor.hu/links/?query=query+AND+1%3D1++  X-Frame-Options  Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).  http://blogs.msdn.com/b/einternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px  I5  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/  HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then

Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	
WASC Id	http://www.owasp.org/index.php/HttpOnly
	13
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/board/?query=query+AND+1%3D1
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	
	13
CWE Id	
Low(Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
LOW(MEGIAITI)	
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/peek/?all=1
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/peek/?all=1
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.
Other information	com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco
WASC Id	de.com/2014/03/guidelines-for-setting-security-headers/
CWE Id	
	933
Low(Medium)	Private IP Disclosure
Description	A private IP (such as 10.x.x.x, 172.x.x.x, 192.168.x.x) or an Amazon EC2 private hostname (for example, ip-10-0-56-78) has been found in the HTTP response body. This information might be helpful for further attacks targeting internal systems.
URL	http://hax.tor.hu/peek/?all=1
Parameter	
Other information	10.0.0.5

Attack	
Solution	Remove the private IP address from the HTTP response body. For comments, use JSP/ASP/PHP
	comment instead of HTML/JavaScript comment which can be seen by client browsers.
Reference WASC Id	https://tools.ietf.org/html/rfc1918
CWE Id	13 200
Low(Medium)	X-Content-Type-Options Header Missing  The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing
Description	on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/peek/?all=1
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can
Reference	be directed by the web application/web server to not perform MIME-sniffing.  http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php
WASC Id	/List_of_useful_HTTP_headers 15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
URL	attacks.
Parameter	http://hax.tor.hu/peek/?all=1
Other information	X-Frame-Options
Attack	
Allduk	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set
Solution	on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
WASC Id	15
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/peek/?query=query%27+AND+%271%27%3D%271%27++
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
	the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/peek/?query=query%27+AND+%271%27%3D%271%27++
Parameter Other information	X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	1011 2010 1011gtil).
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection
	HTTP response header to '1'.  https://www.evecsp.org/index.php/VSS_(Cross_Site_Seripting)_Prevention_Chect_Sheet https://blog.verses_
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veraco

	de.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	Private IP Disclosure
Description	A private IP (such as 10.x.x.x, 172.x.x.x, 192.168.x.x) or an Amazon EC2 private hostname (for example, ip-10-0-56-78) has been found in the HTTP response body. This information might be helpful for further attacks targeting internal systems.
URL	http://hax.tor.hu/peek/?query=query%27+AND+%271%27%3D%271%27++
Parameter	
Other information	10.0.0.5
Attack	
Solution	Remove the private IP address from the HTTP response body. For comments, use JSP/ASP/PHP comment instead of HTML/JavaScript comment which can be seen by client browsers.
Reference	https://tools.ietf.org/html/rfc1918
WASC Id CWE Id	13 200
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/peek/?query=query%27+AND+%271%27%3D%271%27++
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/peek/?query=query%27+AND+%271%27%3D%271%27++
Parameter	X-Frame-Options
Other information Attack	
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
WASC Id	15
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/board/
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id CWE Id	13
	16 Cookie No HttpOply Flog
Low(Medium)	Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can
Description	be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this

URL	http://hax.tor.hu/haxmin/
Parameter	HAXTOR
Other information	TRUCTOR
Attack	
Solution	Engure that the UttoOply flag is get for all englise
	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
	A cookie has been set without the HttpOnly flag, which means that the cookie can
Description	be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/irc/
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/irc/?query=query%22+AND+%221%22%3D%221%22++
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	
WASC Id	http://www.owasp.org/index.php/HttpOnly
	13
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this
	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then
Description URL	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/
Description  URL  Parameter	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
Description  URL  Parameter  Other information	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/
Description  URL  Parameter  Other information  Attack	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/ HAXTOR
Description  URL Parameter Other information Attack Solution	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.
Description  URL Parameter Other information Attack Solution Reference	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly
Description  URL Parameter Other information Attack Solution Reference WASC Id	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible. http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies. http://www.owasp.org/index.php/HttpOnly
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly  13  16
Description  URL Parameter Other information Attack Solution Reference WASC Id	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly  13  16  Cookie No HttpOnly Flag
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/  HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly  13  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly  13  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description  URL	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly  13  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly  13  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/login/
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly  13  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/login/
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly  13  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/login/ HAXTOR
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack Solution	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible. http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies. http://www.owasp.org/index.php/HttpOnly  13  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible. http://hax.tor.hu/login/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack Solution Reference	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly  13  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/login/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack Solution Reference WASC Id	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly  13  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/login/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly  13
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id CWE Id	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly  13  16  Cookie No HttpOnly Flag  A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/login/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly  13  16
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack Solution Reference WASC Id	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly 13 16  Cookie No HttpOnly Flag A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.  http://hax.tor.hu/login/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies.  http://www.owasp.org/index.php/HttpOnly 13 16  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible. http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies. http://www.owasp.org/index.php/HttpOnly 13 16  Cookie No HttpOnly Flag A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible. http://hax.tor.hu/login/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies. http://www.owasp.org/index.php/HttpOnly 13 16  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type.
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description  URL	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible. http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies. http://www.owasp.org/index.php/HttpOnly  13  16  Cookie No HttpOnly Flag A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible. http://hax.tor.hu/login/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies. http://www.owasp.org/index.php/HttpOnly  13  16  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. (if one is set), rather than performing MIME-sniffing. http://hax.tor.hu/stats/detect.txt
Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description  URL Parameter Other information Attack Solution Reference WASC Id CWE Id Low(Medium)  Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible. http://hax.tor.hu/peek/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies. http://www.owasp.org/index.php/HttpOnly 13 16  Cookie No HttpOnly Flag A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible. http://hax.tor.hu/login/ HAXTOR  Ensure that the HttpOnly flag is set for all cookies. http://www.owasp.org/index.php/HttpOnly 13 16  X-Content-Type-Options Header Missing The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type.

	concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	Tight the shold this scarner will not alert on client of server entri responses.
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27+++AND+1%3D1++
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query' AND '1'='1' AND 1=1 ] and [query' AND '1'='1' AND 1=2 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	query' AND '1'='1' AND 1=1
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
CWE Id High(Medium)	
CWE Id High(Medium) Description	89
CWE Id High(Medium) Description URL	SQL Injection
CWE Id High(Medium) Description	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css+AND+1%3D1++ query
CWE Id High(Medium) Description URL	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css+AND+1%3D1++
CWE Id High(Medium) Description URL Parameter	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [/haxtor.css AND 1=1 ] and [/haxtor.css AND 1=2 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by
CWE Id High(Medium) Description URL Parameter Other information	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [/haxtor.css AND 1=1 ] and [/haxtor.css AND 1=2 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter /haxtor.css AND 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
CWE Id High(Medium) Description URL Parameter Other information Attack Solution Reference	SQL injection  SQL injection may be possible.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [/haxtor.css AND 1=1 ] and [/haxtor.css AND 1=2 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter /haxtor.css AND 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
CWE Id High(Medium) Description URL Parameter Other information Attack Solution  Reference WASC Id	SQL injection  SQL injection may be possible.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [/haxtor.css AND 1=1] and [/haxtor.css AND 1=2] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter /haxtor.css AND 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do 'not' concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
CWE Id High(Medium) Description URL Parameter Other information Attack Solution  Reference WASC Id CWE Id	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [/haxtor.css AND 1=1 ] and [/haxtor.css AND 1=2 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter /haxtor.css AND 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database user possible. In particular, avoid using the 'sa' or 'db-owner' database user sess. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet
CWE Id High(Medium) Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id High(Medium)	SQL injection  SQL injection may be possible.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [/haxtor.css AND 1=1] and [/haxtor.css AND 1=2] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter /haxtor.css AND 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do 'not' concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
CWE Id High(Medium) Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id High(Medium) Description	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [/haxtor.css AND 1=1 ] and [/haxtor.css AND 1=2 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter /haxtor.css AND 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database user possible. In particular, avoid using the 'sa' or 'db-owner' database user sess. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet
CWE Id High(Medium) Description URL Parameter Other information Attack  Solution  Reference WASC Id CWE Id High(Medium)	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [/haxtor.css AND 1=1 ] and [/haxtor.css AND 1=2 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter /haxtor.css AND 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database user possible. In particular, avoid using the 'sa' or 'db-owner' database user sess. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19  SQL_Injection
CWE Id High(Medium) Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id High(Medium) Description	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [/haxtor.css AND 1=1] and [/haxtor.css AND 1=2 -] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter /haxtor.css AND 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by "?" If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19  89  SQL Injection SQL Injection SQL injection may be possible. http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B+AND+1%3D1++ query
CWE Id High(Medium) Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id High(Medium) Description URL	SQL Injection SQL injection may be possible. http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css+AND+1%3D1++ query The page results were successfully manipulated using the boolean conditions [/haxtor.css AND 1=1 ] and [/haxtor.css AND 1=2] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter /haxtor.css AND 1=1 Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do 'not' concatenate strings into queries in the stored procedure, or use 'exec,' exec inmediate,' or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19  SQL Injection SQL injection sQL injection may be possible. http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B+AND+1%3D1++ query The page results were successfully manipulated using the boolean conditions [javascript:alert(1); AND 1=1 ] and [javascript:alert(1); OR 1=1] The parameter value being modified was NOT stripped from t
CWE Id High(Medium) Description URL Parameter  Other information  Attack  Solution  Reference WASC Id CWE Id High(Medium) Description URL Parameter	SQL Injection  SQL injection may be possible.  http://hax.tor.hu/html/haxtor.css?query=%2Fhaxtor.css+AND+1%3D1++ query  The page results were successfully manipulated using the boolean conditions [/haxtor.css AND 1=1 -] and [/haxtor.css AND 1=2] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter //haxtor.css AND 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do 'not' concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes is impact. Grant the minimum database access that is necessary for the application. https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet  9  SQL Injection SQL injection may be possible. http://hax.tor.hu/login/?query=javascript%3Aalert%281%29%3B+AND+1%3D1++ query The page results were successfully manipulated using the boolean conditions [javascript:alert(1); AND 1=1 -] and [javascript:alert(1); OR 1=1] The parameter value being modified was NOT stripped from the TML cutypt o

Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create
Column .	dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/html?query=query+AND+1%3D1++
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query AND 1=1 ] and [query AND 1=2 ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	query AND 1=1
Audok	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used,
Solution	use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/peek/?query=query+AND+1%3D1++
Parameter	query
	The page results were successfully manipulated using the boolean conditions [query
Other information	AND 1=1 ] and [query AND 1=2 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	restricting the data originally returned, by manipulating the parameter
Attack	query AND 1=1
Solution	query AND 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure,
	query AND 1=1  Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
	query AND 1=1  Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
Solution  Reference WASC Id	query AND 1=1  Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet
Solution  Reference  WASC Id  CWE Id	query AND 1=1  Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19
Solution  Reference WASC Id CWE Id High(Medium)	query AND 1=1  Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet
Solution  Reference WASC Id CWE Id High(Medium) Description	query AND 1=1  Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19
Solution  Reference WASC Id CWE Id High(Medium) Description URL	query AND 1=1  Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19  SQL Injection
Solution  Reference WASC Id CWE Id High(Medium) Description	query AND 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19 89  SQL Injection SQL injection may be possible.  http://hax.tor.hu/irc/?query=query%27+AND+%271%27%3D%271%27++ query
Solution  Reference WASC Id CWE Id High(Medium) Description URL	query AND 1=1  Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet  19  SQL Injection  SQL Injection may be possible.  http://hax.tor.hu/irc/?query=query%27+AND+%271%27%3D%271%27++

	was returned for the original parameter. The vulnerability was detected by
	successfully restricting the data originally returned, by manipulating the parameter
Attack	query' AND '1'='1'
	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by
	'?' If the application uses ASP, use ADO Command Objects with strong type
	checking and parameterized queries. If database Stored Procedures can be used,
Solution	use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create
Coldion	dynamic SQL queries using simple string concatenation. Escape all data received
	from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/links/?query=query+AND+1%3D1++%25
Parameter	query
. didiliotol	The page results were successfully manipulated using the boolean conditions [query
	AND 1=1 %] and [query AND 1=1 XYZABCDEFGHIJ] The parameter value being
Other information	modified was NOT stripped from the HTML output for the purposes of the comparison Data
	was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	query AND 1=1 %
	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type
	checking and parameterized queries. If database Stored Procedures can be used,
Solution	use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received
	from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
WASC Id	on_Cheat_Sheet 19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/board/?query=query%27+AND+%271%27%3D%271%27++
Parameter	query
. aramoto.	The page results were successfully manipulated using the boolean conditions [query'
	AND '1'='1' ] and [query' AND '1'='2' ] The parameter value being modified
Other information	was stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by
	successfully restricting the data originally returned, by manipulating the parameter
Attack	query' AND '1'='1'
	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type
	checking and parameterized queries. If database Stored Procedures can be used,
Colution	use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received
	from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
	on_Cheat_Sheet
WASC Id	19
CWE Id	89
Liab/Maduuss)	
High(Medium)	SQL Injection
Description	SQL injection may be possible.
_ · · · /	•

	The page results were successfully manipulated using the boolean conditions [query"
	AND "1"="1" ] and [query" OR "1"="1" ] The parameter value being modified
Other information	was NOT stripped from the HTML output for the purposes of the comparison Data
	was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	guery" OR "1"="1"
rittaon	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by
	'?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized gueries. If database Stored Procedures can be used,
	use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create
	dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.  https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
Reference	on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/stats?query=query%27+AND+%271%27%3D%271%27++
Parameter	query
	The page results were successfully manipulated using the boolean conditions [query'
	AND '1'='1' ] and [query' OR '1'='1' ] The parameter value being modified
Other information	was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected
	by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query' OR '1'='1'
	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses
	JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type
	checking and parameterized queries. If database Stored Procedures can be used,
	use them. Do *not* concatenate strings into queries in the stored procedure,
Solution	or use 'exec', 'exec immediate', or equivalent functionality! Do not create
	dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Deference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
Reference	on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/?query=query%25
Parameter	query
	The page results were successfully manipulated using the boolean conditions [query%]
Other information	and [queryXYZABCDEFGHIJ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned
Other information	for the original parameter. The vulnerability was detected by successfully
	restricting the data originally returned, by manipulating the parameter
Attack	query%
	Do not trust client side input, even if there is client side validation in place.
	In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by
	'?' If the application uses ASP, use ADO Command Objects with strong type
	checking and parameterized queries. If database Stored Procedures can be used,
Solution	use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create
- Jiduon	dynamic SQL queries using simple string concatenation. Escape all data received
	from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist'
	of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid
	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
WASC Id	on_Cheat_Sheet 19
CWE Id	89
VVVL IU	
High(Medium)	SQL Injection SQL injection may be possible.

Description	
URL	http://hax.tor.hu/peek/?all=1+AND+1%3D1++
Parameter	all
Other information	The page results were successfully manipulated using the boolean conditions [1 AND 1=1 ] and [1 OR 1=1 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	1 OR 1=1
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/stats/?query=query+AND+1%3D1++
Parameter	
Other information	The page results were successfully manipulated using the boolean conditions [query AND 1=1 ] and [query AND 1=2 ] The parameter value being modified was stripped from the HTML output for the purposes of the comparison Data was
Cutor information	returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	query AND 1=1
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/haxmin/?query=query%27+AND+%271%27%3D%271%27++
Parameter	
ı arameter	query  The page results were successfully manipulated using the boolean conditions [query'
Other information	AND '1'=1' ] and [query' OR '1'='1' ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query' OR '1'='1'
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
	on_Cheat_Sheet
WASC Id	19

CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/board/?query=query+AND+1%3D1%27+AND+%271%27%3D%271
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query AND 1=1' AND '1'='1] and [query AND 1=1' AND '1'='2] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Attack	query AND 1=1' AND '1'='1
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/links/?query=guery%22+AND+%221%22%3D%221%22++
Parameter	query
Other information	The page results were successfully manipulated using the boolean conditions [query" AND "1"="1" ] and [query" OR "1"="1" ] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query" OR "1"="1"
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/irc/?query=query%22+AND+%221%22%3D%221%22++%22+AND+%221%22%3
Parameter	D%221%22++
i aiaiiicici	query  The page results were successfully manipulated using the boolean conditions [query"  AND "1"="1" " AND "1"="1" ] and [query" AND "1"="1" " OR "1"="1"  The assemble point modified was NOT stripped from the UTML output
Other information	] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter
Attack	query" AND "1"="1" " OR "1"="1"
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege
	by using the least privileged database user possible. In particular, avoid

	using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection,
	but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
	on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/sitemap.xml?query=query+ASC+++
Parameter	query
Other information	The original page results were successfully replicated using the "ORDER BY" expression [query ASC ] as the parameter value The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison
Attack	query ASC
Solution	Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side. If the application uses  JDBC, use PreparedStatement or CallableStatement, with parameters passed by  '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
D-f	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Preventi
Reference	on_Cheat_Sheet
WASC Id	19
CWE Id	89
High(Medium)	SQL Injection
Description	SQL injection may be possible.
URL	http://hax.tor.hu/stats/?graph=3-2&h=400&mode=top5comparison&topfrom=11&w=800
Parameter	graph
	The original page results were successfully replicated using the expression [3-2]
Other information	as the parameter value The parameter value being modified was stripped from the HTML output for the purposes of the comparison
Attack	3-2
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention Cheat Sheet
WASC Id	19
CWE Id	89
Low(Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this
URL	is a session cookie then session hijacking may be possible.  http://hax.tor.hu/welcome/
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16
Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/welcome/
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable
	the web browser's XSS protection mechanism. The following values would attempt

	to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example. com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and
	Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	Could potentially contain an ASS payload (with a text-based content type, with a non-zero length).
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/welcome/
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/welcome/
Parameter	X-Frame-Options
Other information	
Attack	
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
WASC Id	15
CWE Id	16

# Step: 1.3) http://hax.tor.hu/irc/

### **Alert Detail**

The alert list is empty

Step: 1.4) http://hax.tor.hu/board/

#### **Alert Detail**

The alert list is empty

Step: 1.5) http://hax.tor.hu/peek/

**Alert Detail** 

Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
	the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/peek/
Parameter Other information	X-XSS-Protection  The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body
Attack	could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	Private IP Disclosure
Description	A private IP (such as 10.x.x.x, 172.x.x.x, 192.168.x.x) or an Amazon EC2 private hostname (for example, ip-10-0-56-78) has been found in the HTTP response body. This information might be helpful for further attacks targeting internal systems.
URL	http://hax.tor.hu/peek/
Parameter	
Other information	10.0.0.5
Attack	
Solution	Remove the private IP address from the HTTP response body. For comments, use JSP/ASP/PHP comment instead of HTML/JavaScript comment which can be seen by client browsers.
Reference	https://tools.ietf.org/html/rfc1918
WASC Id	13
CWE Id	200
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/peek/
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set  X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking'
Description	attacks.
URL	http://hax.tor.hu/peek/
Parameter	X-Frame-Options
Other information	
Attack Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as
WASC Id	px 15
CWE Id	15 16
Low(Medium)	Web Browser XSS Protection Not Enabled
	Web Browser XSS Protection is not enabled, or is disabled by the configuration of
Description	the 'X-XSS-Protection' HTTP response header on the web server

URL	http://hax.tor.hu/peek/?all=1
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	Private IP Disclosure
Description	A private IP (such as 10.x.x.x, 172.x.x.x, 192.168.x.x) or an Amazon EC2 private hostname (for example, ip-10-0-56-78) has been found in the HTTP response body. This information might be helpful for further attacks targeting internal systems.
URL	http://hax.tor.hu/peek/?all=1
Parameter	
Other information	10.0.0.5
Attack	
Solution	Remove the private IP address from the HTTP response body. For comments, use JSP/ASP/PHP comment instead of HTML/JavaScript comment which can be seen by client browsers.
Reference	https://tools.ietf.org/html/rfc1918
WASC Id	13
CWE Id	200
Low(Medium)	X-Content-Type-Options Header Missing
Low(Mcdiairi)	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This
Description	allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/peek/?all=1
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/peek/?all=1
Parameter	X-Frame-Options
Other information	
Attack	
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
WASC Id	15
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/peek/
J. \_	

Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16

# Step: 1.6) http://hax.tor.hu/haxmin/

## **Alert Detail**

Low(Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	http://hax.tor.hu/haxmin/
Parameter	X-XSS-Protection
Other information	The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it: X-XSS-Protection: 1; mode=block X-XSS-Protection: 1; report=http://www.example.com/xss The following values would disable it: X-XSS-Protection: 0 The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit). Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).
Attack	3,7
Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Reference	https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/
WASC Id	14
CWE Id	933
Low(Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://hax.tor.hu/haxmin/
Parameter	X-Content-Type-Options
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Attack	
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php /List_of_useful_HTTP_headers
WASC Id	15
CWE Id	16
Medium(Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://hax.tor.hu/haxmin/
Parameter	X-Frame-Options
Other information	
Attack	
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.as px
WASC Id	15
CWE Id	16
Low(Medium)	Cookie No HttpOnly Flag
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can

	be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://hax.tor.hu/haxmin/
Parameter	HAXTOR
Other information	
Attack	
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	http://www.owasp.org/index.php/HttpOnly
WASC Id	13
CWE Id	16