专注APT攻击与防御

https://micropoor.blogspot.com/

本课是针对前第1-20课时的msfvenom生成payload的自动补全命令补充。虽

msfvenom强大，同样有着非常繁琐的参数，参数强大，意味着会增加工作效率，但它并不

像MSF有命令补全功能，故本课吸取前20课经验，自动补全msfvenom的参数。

需要zsh的支持：

1 root@John:~# cat /etc/shells

2 # /etc/shells: valid login shells

3 /bin/sh

4 /bin/dash

5 /bin/bash

6 /bin/rbash

7 /usr/bin/screen

8 /bin/zsh

9 /usr/bin/zsh

10 /usr/bin/tmux

11 root@John:~# echo $SHELL

12 /bin/bash

复制附录A到~/.oh-my-zsh/custom/plugins/msfvenom文件夹下（注：没有msfvenom

目录，创建即可）

1 root@John:~/.oh‐my‐zsh/custom/plugins/msfvenom# pwd

2 /root/.oh‐my‐zsh/custom/plugins/msfvenom

3 root@John:~/.oh‐my‐zsh/custom/plugins/msfvenom# ls

4 \_msfvenom

编辑~/.zshrc文件：

1 root@John:~# nano ~/.zshrc

1 root@John:~# nano ~/.zshrc

2 root@John:~# cat ~/.zshrc

3 plugins=(msfvenom)

更新：

1 root@John:~# source ~/.zshrc

效果如下：

附录A：

1 #compdef msfvenom

2 #autoload

3 #

4 # zsh completion for msfvenom in Metasploit Framework Project (http s://www.metasploit.com)

5 #

6 # github: https://github.com/Green‐m/msfvenom‐zsh‐completion

7 #

8 # author: Green‐m (greenm.xxoo@gmail.com)

9 #

10 # license: GNU General Public License v3.0

11 #

12 # Copyright (c) 2018, Green‐m

13 # All rights reserved.

14 # 15

16 VENOM\_CACHE\_FILE=~/.zsh/venom‐cache 17

18 venom‐clear‐cache() {

19 rm $VENOM\_CACHE\_FILE

20 } 21

22 venom‐cache‐payloads() { 23

24 if [ ‐x "$(command ‐v msfvenom)" ]

25 then

26 VENOM="msfvenom"

27 elif [ ‐n "$\_comp\_command1" ]

28 then

29 VENOM=$\_comp\_command1

30 else

31 echo "Cound not find msfvenom path in system env, please run msfvenom with path."

32 fi 33

34 if [[ ! ‐d ${VENOM\_CACHE\_FILE:h} ]]; then

35 mkdir ‐p ${VENOM\_CACHE\_FILE:h}

36 fi 37

38 if [[ ! ‐f $VENOM\_CACHE\_FILE ]]; then

39 echo ‐n "(...caching Metasploit Payloads...)"

40 $VENOM ‐‐list payload|grep ‐e "^.\*\/" | awk '{print $1}' >> $VENOM\_CA CHE\_FILE

41 fi

42 } 43

44

45 \_msfvenom() { 46

47 local curcontext="$curcontext" state line

48 typeset ‐A opt\_args 49

50 \_arguments ‐C \

51 '(‐h ‐‐help)'{‐h,‐‐help}'[show help]' \

52 '(‐l ‐‐list)'{‐l,‐‐list}'[List all modules for type. Types are: paylo ads, encoders, nops, platforms, archs, encrypt, formats, all]' \

53 '(‐p ‐‐payload)'{‐p,‐‐payload}'[Payload to use (‐‐list payloads to li st, ‐‐list‐options for arguments). Specify ‐ or STDIN for custom]' \

54 '(‐‐list‐options)‐‐list‐options[List ‐‐payload <value> standard, adva nced and evasion options]' \

55 '(‐f ‐‐format)'{‐f,‐‐format}'[Output format (use ‐‐list formats to li st)]' \

56 '(‐e ‐‐encoder)'{‐e,‐‐encoder}'[The encoder to use (use ‐‐list encode rs to list)]' \

57 '(‐‐smallest)‐‐smallest[Generate the smallest possible payload using all available encoders]' \

58 '(‐‐encrypt)‐‐encrypt[The type of encryption or encoding to apply to the shellcode (use ‐‐list encrypt to list)]' \

59 '(‐‐encrypt‐key)‐‐encrypt‐key[A key to be used for ‐‐encrypt]' \

60 '(‐‐encrypt‐iv)‐‐encrypt‐iv[An initialization vector for ‐‐encrypt]' \

61 '(‐a ‐‐arch)'{‐a,‐‐arch}'[the architecture to use for ‐‐payload and ‐ ‐encoders (use ‐‐list archs to list)]' \

62 '(‐‐platform)‐‐platform[The platform for ‐‐payload (use ‐‐list platfo rms to list)]' \

63 '(‐o ‐‐out)'{‐o,‐‐out}'[Save the payload to a file]' \

64 '(‐b ‐‐bad‐chars)'{‐b,‐‐bad‐chars}'[Characters to avoid example: "\x0 0\xff"]' \

65 '(‐n ‐‐nopsled)'{‐n,‐‐nopsled}'[Prepend a nopsled of \[length\] size on to the payload]' \

66 '(‐‐encoder‐space)‐‐encoder‐space[The maximum size of the encoded pay load (defaults to the ‐s value)]' \

67 '(‐i ‐‐iterations)'{‐i,‐‐iterations}'[The number of times to encode t he payload]' \

68 '(‐c ‐‐add‐code)'{‐c,‐‐add‐code}'[Specify an additional win32 shellco de file to include]' \

69 '(‐x ‐‐template)'{‐x,‐‐template}'[Specify a custom executable file to use as a template]' \

70 '(‐k ‐‐keep)'{‐k,‐‐keep}'[Preserve the ‐‐template behaviour and injec t the payload as a new thread]' \

71 '(‐v ‐‐var‐name)'{‐v,‐‐var‐name}'[Specify a custom variable name to u se for certain output formats]' \

72 '(‐t ‐‐timeout)'{‐t,‐‐timeout}'[The number of seconds to wait when re ading the payload from STDIN (default 30, 0 to disable)]' \

73 '\*: :($(\_\_msfvenom\_options))' && ret=0 74

75

76 lastword=${words[${#words[@]}‐1]} 77

78 case "$lastword" in

79 (‐p|‐‐payload)

80 \_values 'payload' $(\_\_msfvenom\_payloads)

81 ;; 82

83 (‐l|‐‐list)

84 local lists=('payloads' 'encoders' 'nops' 'platforms' 'archs' 'encryp t' 'formats' 'all')

85 \_values 'list' $lists

86 ;; 87

88 (‐encrypt)

89 local encrypts=('aes256' 'base64' 'rc4' 'xor')

90 \_values 'encrypt' $encrypts

91 ;; 92

93 (‐a|‐‐arch)

94 \_values 'arch' $(\_\_msfvenom\_archs)

95 ;; 96

97 (‐platform)

98 \_values 'platform' $(\_\_msfvenom\_platforms)

99 ;; 100

101 (‐f|‐‐format)

102 \_values 'format' $(\_\_msfvenom\_formats)

103 ;; 104

105 (‐e|‐‐encoder)

106 \_values 'encoder' $(\_\_msfvenom\_encoders)

107 ;; 108

109 (‐o|‐‐out|‐x|‐‐template|‐c|‐‐add‐code)

110 \_files

111 ;; 112

113 (\*) 114

115 ;; 116

117 esac

118 } 119

120

121 \_\_msfvenom\_payloads(){

122 local msf\_payloads 123

124 # we cache the list of packages (originally from the macports plugi n)

125 venom‐cache‐payloads

126 msf\_payloads=`cat $VENOM\_CACHE\_FILE` 127

128 for line in $msf\_payloads; do

129 echo "$line"

130 done

131 } 132

133 \_\_msfvenom\_archs(){

134 local archs

135 archs=(

136 'aarch64'

137 'armbe'

138 'armle'

139 'cbea'

140 'cbea64'

141 'cmd'

142 'dalvik'

143 'firefox'

144 'java'

145 'mips'

146 'mips64'

147 'mips64le'

148 'mipsbe'

149 'mipsle'

150 'nodejs'

151 'php'

152 'ppc'

153 'ppc64'

154 'ppc64le'

155 'ppce500v2'

156 'python'

157 'r'

158 'ruby'

159 'sparc'

160 'sparc64'

161 'tty'

162 'x64'

163 'x86'

164 'x86\_64'

165 'zarch'

166 ) 167

168 for line in $archs; do

169 echo "$line"

170 done 171

172 } 173

174 \_\_msfvenom\_encoders(){

175 local encoders

176 encoders=(

177 'cmd/brace'

178 'cmd/echo'

179 'cmd/generic\_sh'

180 'cmd/ifs'

181 'cmd/perl'

182 'cmd/powershell\_base64'

183 'cmd/printf\_php\_mq'

184 'generic/eicar'

185 'generic/none'

186 'mipsbe/byte\_xori'

187 'mipsbe/longxor'

188 'mipsle/byte\_xori'

189 'mipsle/longxor'

190 'php/base64'

191 'ppc/longxor'

192 'ppc/longxor\_tag'

193 'ruby/base64'

194 'sparc/longxor\_tag'

195 'x64/xor'

196 'x64/xor\_dynamic'

197 'x64/zutto\_dekiru'

198 'x86/add\_sub'

199 'x86/alpha\_mixed'

200 'x86/alpha\_upper'

201 'x86/avoid\_underscore\_tolower'

202 'x86/avoid\_utf8\_tolower'

203 'x86/bloxor'

204 'x86/bmp\_polyglot'

205 'x86/call4\_dword\_xor'

206 'x86/context\_cpuid'

207 'x86/context\_stat'

208 'x86/context\_time'

209 'x86/countdown'

210 'x86/fnstenv\_mov'

211 'x86/jmp\_call\_additive'

212 'x86/nonalpha'

213 'x86/nonupper'

214 'x86/opt\_sub'

215 'x86/service'

216 'x86/shikata\_ga\_nai'

217 'x86/single\_static\_bit'

218 'x86/unicode\_mixed'

219 'x86/unicode\_upper'

220 'x86/xor\_dynamic'

221 )

222

223 for line in $encoders; do

224 echo "$line"

225 done

226 } 227

228 \_\_msfvenom\_platforms(){

229 local platforms

230 platforms=(

231 'aix'

232 'android'

233 'apple\_ios'

234 'bsd'

235 'bsdi'

236 'cisco'

237 'firefox'

238 'freebsd'

239 'hardware'

240 'hpux'

241 'irix'

242 'java'

243 'javascript'

244 'juniper'

245 'linux'

246 'mainframe'

247 'multi'

248 'netbsd'

249 'netware'

250 'nodejs'

251 'openbsd'

252 'osx'

253 'php'

254 'python'

255 'r'

256 'ruby'

257 'solaris'

258 'unix'

259 'unknown'

260 'windows'

261 )

262

263 for line in $platforms; do

264 echo "$line"

265 done

266 } 267

268 \_\_msfvenom\_formats(){

269 local formats

270 formats=(

271 'asp'

272 'aspx'

273 'aspx‐exe'

274 'axis2'

275 'dll'

276 'elf'

277 'elf‐so'

278 'exe'

279 'exe‐only'

280 'exe‐service'

281 'exe‐small'

282 'hta‐psh'

283 'jar'

284 'jsp'

285 'loop‐vbs'

286 'macho'

287 'msi'

288 'msi‐nouac'

289 'osx‐app'

290 'psh'

291 'psh‐cmd'

292 'psh‐net'

293 'psh‐reflection'

294 'vba'

295 'vba‐exe'

296 'vba‐psh'

297 'vbs'

298 'war'

299 'bash'

300 'c'

301 'csharp'

302 'dw'

303 'dword'

304 'hex'

305 'java'

306 'js\_be'

307 'js\_le'

308 'num'

309 'perl'

310 'pl'

311 'powershell'

312 'ps1'

313 'py'

314 'python'

315 'raw'

316 'rb'

317 'ruby'

318 'sh'

319 'vbapplication'

320 'vbscript'

321 ) 322

323 for line in $formats; do

324 echo "$line"

325 done

326 } 327

328 # For most common options, not accurately

329 \_\_msfvenom\_options(){

330 local options

331 options=(

332 LHOST= \

333 LPORT= \

334 EXITFUNC= \

335 RHOST= \

336 StageEncoder= \

337 AutoLoadStdapi= \

338 AutoRunScript= \

339 AutoSystemInfo= \

340 AutoVerifySession= \

341 AutoVerifySessionTimeout= \

342 EnableStageEncoding= \

343 EnableUnicodeEncoding= \

344 HandlerSSLCert= \

345 InitialAutoRunScript= \

346 PayloadBindPort= \

347 PayloadProcessCommandLine= \

348 PayloadUUIDName= \

349 PayloadUUIDRaw= \

350 PayloadUUIDSeed= \

351 PayloadUUIDTracking= \

352 PrependMigrate= \

353 PrependMigrateProc= \

354 ReverseAllowProxy= \

355 ReverseListenerBindAddress= \

356 ReverseListenerBindPort= \

357 ReverseListenerComm= \

358 ReverseListenerThreaded= \

359 SessionCommunicationTimeout= \

360 SessionExpirationTimeout= \

361 SessionRetryTotal= \

362 SessionRetryWait= \

363 StageEncoder= \

364 StageEncoderSaveRegisters= \

365 StageEncodingFallback= \

366 StagerRetryCount= \

367 StagerRetryWait= \

368 VERBOSE= \

369 WORKSPACE=

370 ) 371

372 echo $options

373 } 374

375

376 #\_msfvenom "$@"

Micropoor