1 程序简介

Railgun 为一款 GUI 界面的渗透工具，将部分人工经验转换为自动化，集成了渗透过程中常用到的一些功能，目前集成了端口扫描、端口爆破、web 指纹扫描、漏洞扫描、漏洞利用以及编码转换功能，后续会持续更新。

1.1 Tips

1. 工具为 64 位程序。

2. 工具设置有启动密码，密码请联系作者。

3. 工具设有托盘，关闭窗口不会关闭程序，需通过托盘操作。双击托盘可还原或最小化。

4. 如程序出现问题，可点击重启应用快速重启应用。

5. 程序退出可通过托盘，或者选择菜单栏 文件-退出。

1.2 更新日志

2020-08-22
v1.2.8

1. 信息收集：新增结果栏复制及粘贴功能

2. 端口扫描新增多平台命令行版本，结果可直接导入 GUI 界面。

3．新增目录扫描模块，除正常目录扫描功能，还可一键生成自定义目录字典，方便定向扫描。

2020-08-03
v1.2.7

1．端口扫描：新增 RPC 网卡扫描

2．漏洞扫描：增加 thinkphp poc

3．漏洞利用：新增 shiro550 新检测方法、CVE-2020-14645

4．漏洞利用：优化反序列化 gadget 探测、dnslog 回显及分段传输上传通用回调函数，weblogic/s2/thinkphp 均增加如上功能

6．漏洞利用：dnslog 增加 ceye 接口调用

2020-07-09
v1.2.5

1．漏洞利用模块：新增 F5 RCE exp 及 poc；新增 shiro padding oracle 利用；shiro rce 新增分段上传文件功能。

2．优化漏洞利用及漏洞扫描框架

v1.2.4

1．漏洞利用模块：优化部分 exp

v1.2.3

1．dnslog 脚本及相关模块优化。

2．增加各个模块异常日志记录。

3．修复 tomcat-AJP_LFI GBK 乱码问题。

4．漏洞扫描模块：修复 CVE-2019-2725 漏判问题。

2020-05-29
v1.1.0.9

1．修复各模块由于 UI 库更新的 bug。

2．编码转换：增加自动装换功能。

3．端口扫描： 增加 netbios 探测 windows 主机存活。

4．优化了 shiro 的 poc 验证过程。

5．优化了 shiro 的 exp 利用。

2020-05-17
1.1.0.8

1．UI 库更新，从 go 版 delphi 迁移至 lazarus。

2．爆破模块：修复 mssql 爆破 bug。

3．漏洞利用模块： 新增自定义 header 及自定义请求包。

2020-05-01
v1.1.0.7

1．端口扫描：双击端口会自动打开网页访问，无论是否识别为 web。

2．端口扫描：新增推荐端口范围选择按钮。

3．smb 低版本爆破 bug 修复。

4．漏洞利用模块：设置未启用功能隐藏。

5．web 指纹：新增 shiro 指纹识别开关（由于需要设置 cookie）。

6．新增域名扫描模块（Demo），目前只支持域名解析。

7．菜单新增爆破字典 UI 设置

8．端口扫描增加域名识别,可导入 IP 列表。

9．增加信息收集各模块联动功能，将域名及 IP 进行关联。

2020-04-25
V1.1.0.6

1．增加通达 OA 任意用户登录漏洞

2．增加 phpcms9.6.0 前台上传漏洞

3．dnslog 设置 API 配置文件存储。

4．爆破模块新增用户密码导入。

5．端口扫描，识别为 web 协议点击 IP 会自动打开浏览器访问

6．各类扫描结果可多选复制至剪切板。

7．unicode 编码 bug 修复。

2020-04-17

1．添加 thinkphp 上传 webshell 功能。

2．shiro 反序列化更新密钥并优化 exp

3．dnslog 过滤结果 bug 修复

4．端口扫描增加多次校验机制

2020-04-01
1.1.0.4

1．新增 Jboss CVE-2017-0704 CVE-2017-12149 exp

2．新增重启应用功能

3．新增 MDB 编码

4．新增更新模块

5．编码模块新增拖拽读取文件

6．调整爆破模块设置，更新为 IP 并发*每 IP 并发。

7．新增启动密码。

2020-03-15
1.1.0.2

UI 库 dll 文件保存至资源段运行自动释放，后续无需携带 dll 文件。

增加 UTF7 编码。

更新通达 OAgetshell

URL 编码优化

2020-03-07
1.1.0.1

1. webfinger 进度条没满就退出了，（解决，主要是因为不检测每任务 channel 是否为空）

2. web 指纹 title html 解码（已解决）

3. 爆破首次检测（通过 map 设置 timeout 检测已解决）

4. 导入列表自动添加 http 前缀(已新增，可设置导入域名所自动添加的协议和端口）

5. 暴力破解增加单账号爆破成功则结束该 IP 爆破。

6. 端口 banner 开关（已添加）

7. 解决 webfinger 扫描超时过长问题

8. listview 的 subitem 编辑框偏移错位 bug 修复

2020-03-03
1.1.0.0

将 python 大部分代码迁移至 go，在工具大小和稳定性、速度上都得到了提升。

MS17-010 漏洞 exp 暂时未迁移，后续考虑单独脚本利用。

编码模块只迁移了小部分常用编码，其他类型后续再考虑增加。

新增 TOMCAT-AJP 漏洞的 poc 和 exp。

新增 web 指纹扫描功能，使用 fofa+cms 方式扫描，已经做了不少识别率的优化。

新增端口扫描结果发送至 web 指纹扫描模块，将识别为 http/https 的进行发送。

端口扫描、web 指纹新增结果保存功能，可将结果导出为 csv 格式文件。

漏洞扫描模块新增 chunk 编码扫描以及代理扫描。

weblogic xml 反序列化新增 payload，并增加文件上传功能。

2019-12-25
1.0.2.0

exp 更新 chunk 分块传输，但与代理冲突，开了 chunk 再使用代理会超时。

支持自定义 web、smb 端口及是否开启额外特征识别扫描。

更新系统托盘。

优化启动窗口不置顶问题。

优化了 poc 扫描，标识哪几个 payload 有效。

需要注意的是 chunk 编码是全局的，因为直接修改了 requests 库，后续再优化成单个模块。

2019-12-12
1.0.1.9

更新编码模块，新增 unicode 的 base64 编码。

更新 dnslog 模块，自定义 API 获取 dnslog 结果，并支持 dnslog 拼接解码还

原执行结果。

2019-12-05
1.0.1.8

更新编码模块，增加 java.lang.Runtime.exec 编码

可使用 ctrl+enter 快捷进行编码转换

增加编码状态保存功能，可以在至多 3 种编码配置间切换,使用 ctrl+数字键可

快速切换

2019-12-02
1.0.1.7

更新 rdp 爆破模块，可支持 win7 以上系统爆破

优化 exp 模板，更易于扩展。

优化 shiro exp，可遍历 key 查找正确 key，且可自定义 key

2019-11-16
1.0.1.6

修改编码模块 UI

新增程序启动界面，缓解启动慢问题

2019-11-11
1.0.1.5

新增 ysoserial payload，增加 shiro 反序列化。

2019-11-05
1.0.1.4

暴力破解模块进行优化，修复 bug。

2019-11-04
1.0.1.4beta

更新了暴力破解模块，该模块支持端口较少，后续会继续增加

漏洞利用模块新增批量攻击功能，通过设置多行 URL 即可

工具更新成至 64 位，后续只支持 win7 以上 64 位系统运行

1.0.1.3

新增编码转换模块

rdp/smb 支持多端口同时漏洞扫描，端口格式参考端口扫描

1.0.1.2

poc 漏洞选择 UI 优化。

新增泛微 OA/seeyon OA 等 exp

exp 新增超时设置

增加更新检测功能

工具汉化

修复子线程卡主线程的 bug

1.0.1.0

重构了 exp 部分，方便后续新的 exp 可以以插件形式更新。

漏扫部分新增 web 漏洞扫描，可以选择多个漏洞类型，批量扫描。

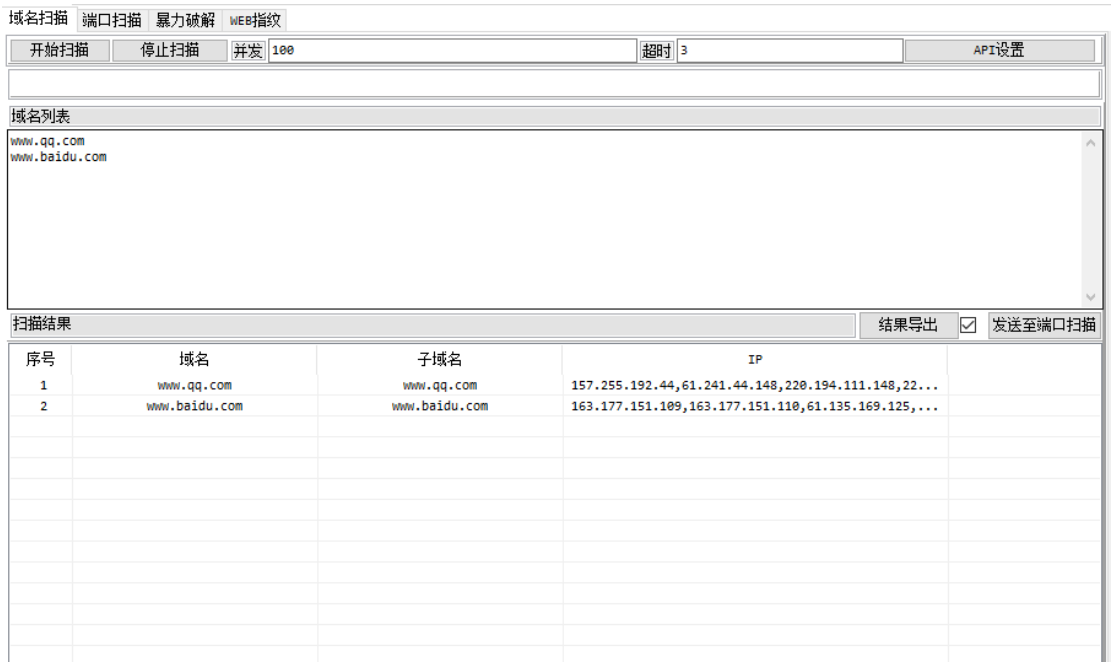
扫描方式改为多进程方式，提高扫描速度。

2 信息收集

目前信息收集可将域名扫描、端口扫描、web 指纹联动，可输入已收集的子域名列表，自动完成解析、端口扫描、web 指纹扫描。

2.1 域名扫描

域名扫描目前还只是 demo 版，仅支持域名解析。界面如下



每行输入一个域名，点击”开始扫描”会输出解析结果。

备注：该模块后面暂时不考虑继续开发了，网上如 oneforall 等工具已经很强大，并且能满足当前需要，所以暂停开发。

参数介绍

1．并发数：单线程多任务并发，可多核 CPU 运行。

超时：单位秒。

2. API 设置，由于 UI 库迁移问题，暂时关闭了。

3. 扫描结果，包含域名/子域名/IP 3 个信息，目前来说域名和子域名是一样的，子域名扫描功能还没做，可以配合其他子域名扫描工具使用，解析 IP 使用逗号分隔。

4. 其他

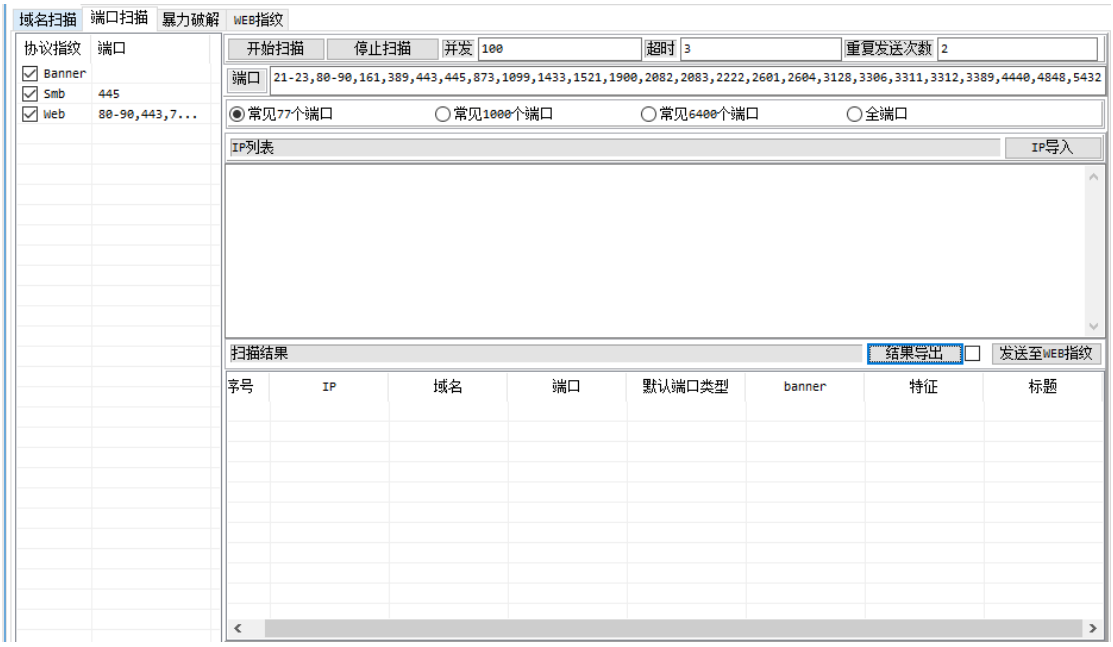
   结果导出：扫描结果导出为 csv。

   发送至端口扫描：将扫描结果 IP 进行整合去重发送至端口扫描，左侧有个勾选框，如果勾选了，在域名扫描结束后会自动发送并执行端口扫描。

   目前这个模块还比较鸡肋，唯一比较好的地方是和端口扫描联动，会将域名和 IP 绑定关系一并发送至端口扫描模块，在端口扫描结果中会显示 IP 和域名的对应关系，具体细节可查看端口扫描部分。

2.2 端口扫描

端口扫描界面如下



端口扫描是使用 goroutine 实现并发扫描的，目前仅支持 TCP 端口扫描。

使用比较简单，输入端口以及 IP 列表，点击开始扫描即可，结果在”扫描结果”实时显示。

其他参数下面一一介绍。

1．并发数：单线程多任务并发，可多核 CPU 运行。

超时：单位秒。

重复发送次数：这里不用去修改，是个还未内置的功能，设置两次连接探测保证目标端口扫描结果可靠。

2．端口默认设置了常见应用端口，如果自定义，格式可参考如下：

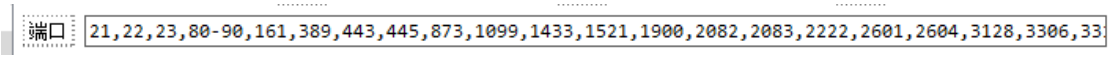
80：只扫描 80 端口

80-90：扫描 80-90 的 11 个端口

80,90：扫描 80 和 90 两个端口

80,90,7001-7010：扫描 80、90、7001 到 7010，一共 12 个端口

示例：



3．新增了常见端口选项设置，点击相应按钮自动切换端口范围，端口范围是参考 nmap。

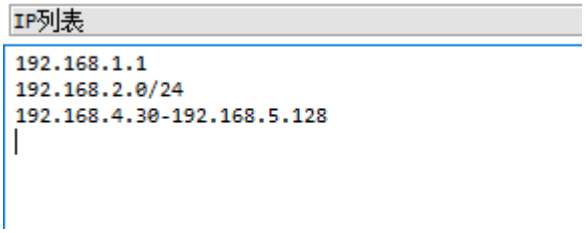
4．IP 列表输入需要扫描 IP，可设置多行，每行设置一个格式，格式参考如下：

192.168.1.1：扫描 192.168.1.1 一个 IP

192.168.1.1-192.168.1.128：扫描 192.168.1.1 到 192.168.1.128 共 128 个 IP

192.168.1.0/24：扫描 192.168.1.0/24 整个 C 段的 IP

示例：



除此之外，还可以手动导入 IP 列表，如果仅仅是 IP 列表，复制粘贴就好了，但这里可以导入域名和 IP 的对应关系，这样扫描结果会直接将域名和 IP 对应起来。

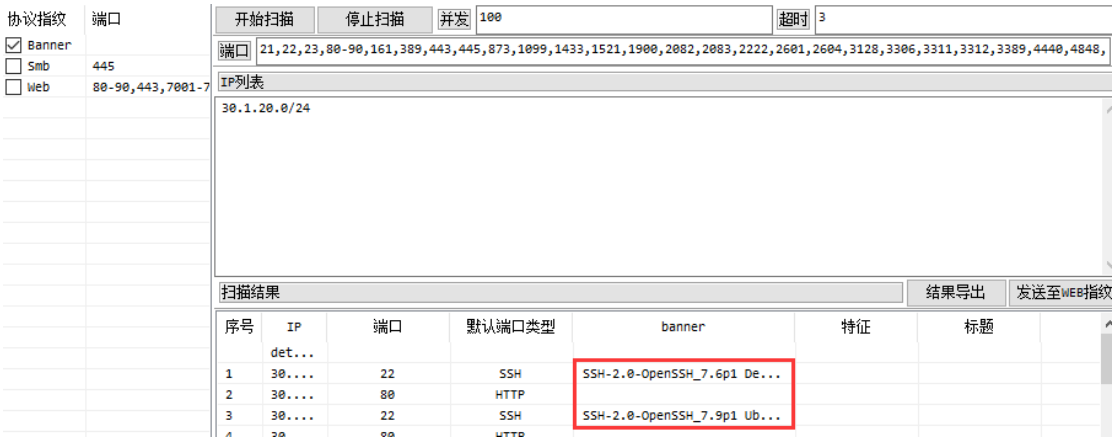
导入格式如下，每行一个域名 IP 对应，用分号分割域名和 IP，一个域名可对应多个 IP，IP 之间用逗号隔开。

```
www.example.com:1.1.1.1,2.2.2.2
www.test1.com:3.3.3.3
```

5. 左侧有协议指纹识别，默认全开启，无需扫描 banner 等信息关闭即可，端口双击即可修改。

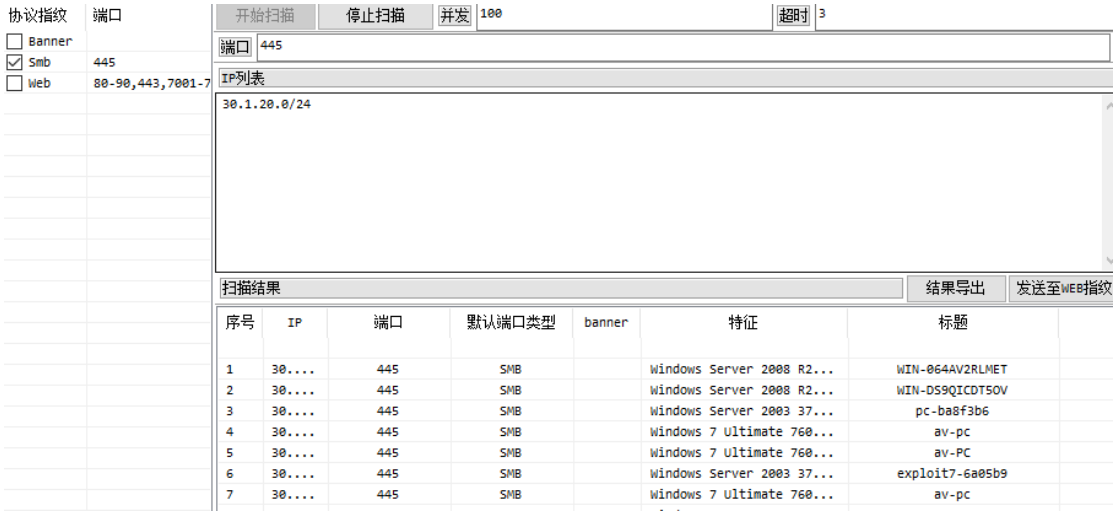
Banner：扫描时会接收 banner 信息进行显示。

示例：



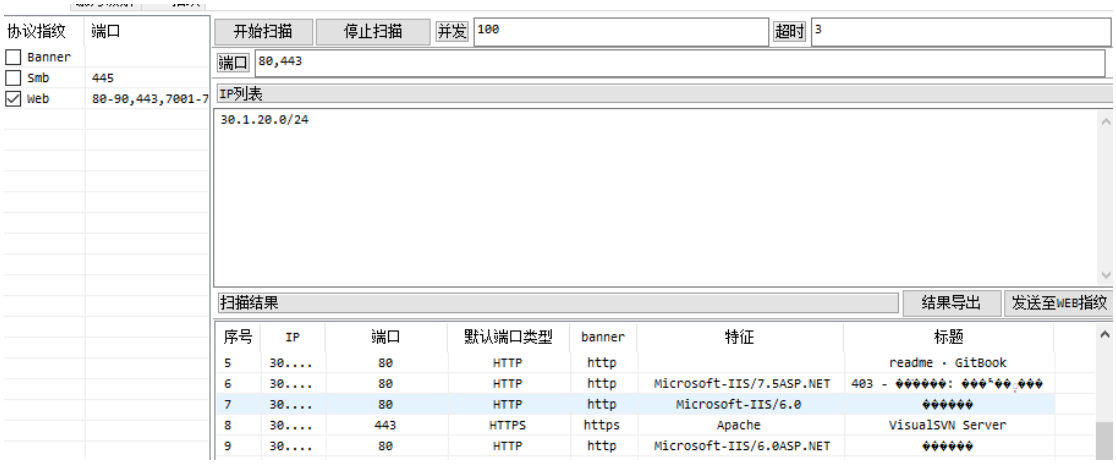
SMB：默认为 445，目前可扫描提取 xp 至 2012 操作系统信息，和主机名（如果为域主机标识域名），如果识别不出操作系统可能为 2016/win10 或特殊原因。（特征栏：操作系统信息，标题栏：主机名及域名）

示例：



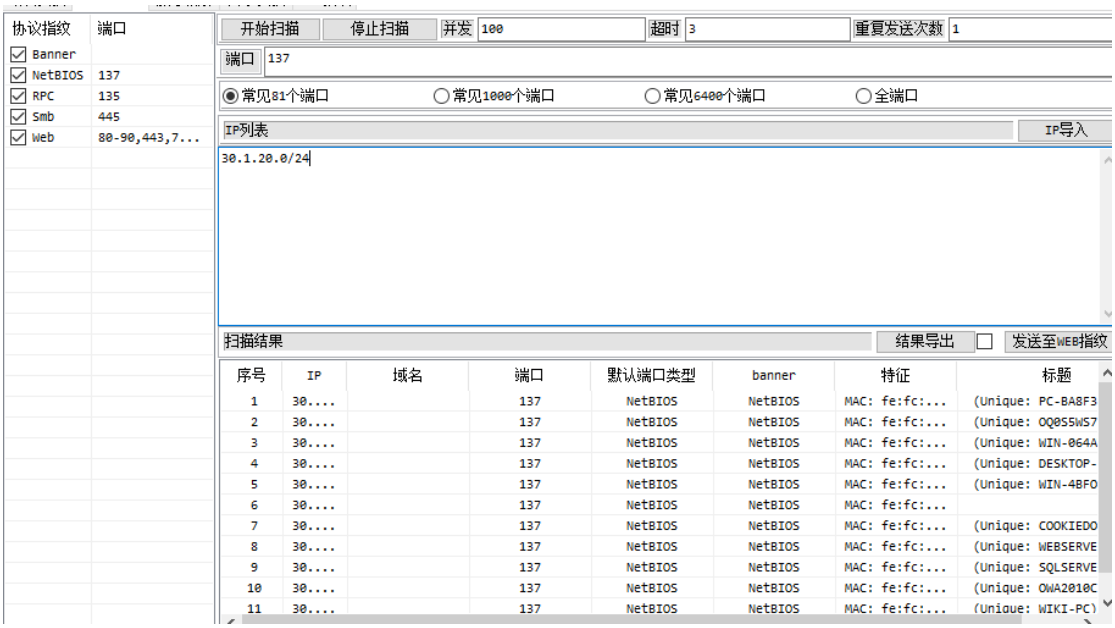
WEB：默认为 80-90,443,7001-7010,8080-8090,8888,9000,9200，会在端口发现基础上发送 GET 请求，自动识别目标是 http 或者 https，并提取返回包里的头部字段里的 x-powered-by 和 server 字段，并获取 body 里的 title 字段，自动进行 302 跳转，目前不会识别 js 跳转。（banner 栏：目标为 http 或 https；特征：头部返回字段信息；标题：body 的 title 字段）

示例：



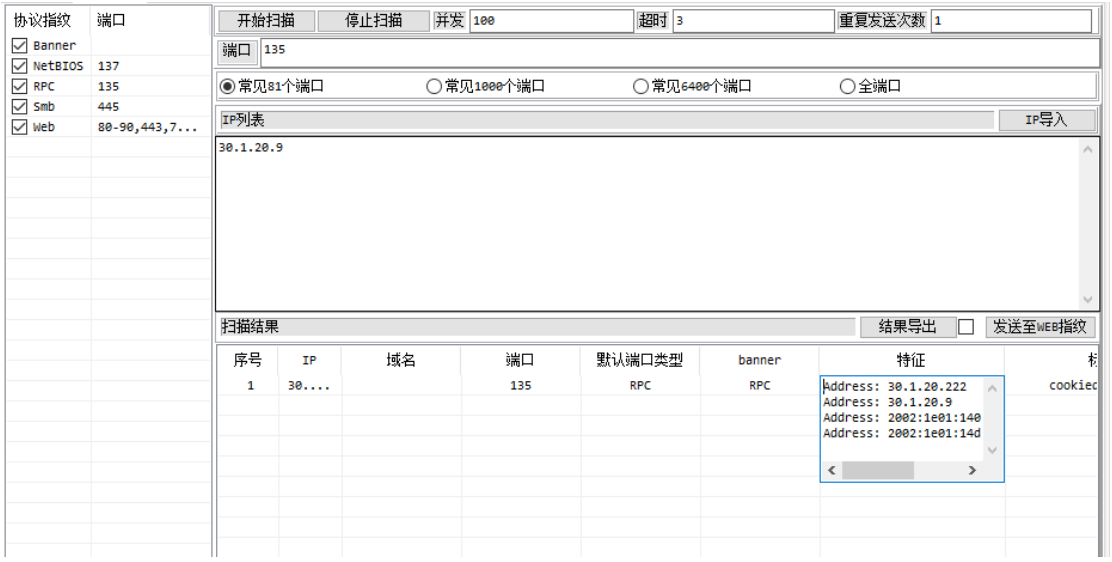
NetBIOS： 默认为 137，是唯一一个使用 UDP 协议扫描端口，可获取目标 MAC 和主机名。

示例：



RPC:默认为 135，通过该协议可扫描目标所有网卡信息，用于发现多网卡主机。

示例：



6．扫描结果

目前包括 IP/域名/端口/默认端口类型/Banner/特征/标题 7 个信息，其中默认端口类型仅仅用于辅助，标识常用端口默认应用类型，并不是指纹识别。(下图缺失域名栏，正常域名栏为空)

扫描结果

	IP	端口	默认端口类型	banner	特征	标题
5	30.1.2…	445	SMB		Windows Server 2008 R2 E…	WIN-064AV2RLMET
6	30.1.2…	22	SSH	SSH-2.0-OpenSSH_7.9p1 Ubuntu-10		
7	30.1.2…	3306	MySQL	B ÿj回Host '30.3.10.9' is not a…		
8	30.1.2…	3306	MySQL	N 5.5.29-		
9	30.1.2…	22	SSH	SSH-2.0-OpenSSH_7.6p1 Debian-4		
10	30.1.2…	80	HTTP	http	Apache/2.4.39 (Win64) Op…	RedTeam-在线工具包
11	30.1.2…	80	HTTP	http	Apache/2.4.38 (Ubuntu)	Apache2 Ubuntu Default Page: It w…
12	30.1.2…	80	HTTP	http	Apache-Coyote/1.1	Apache Shiro Quickstart
13	30.1.2…	80	HTTP	http	Microsoft-HTTPAPI/2.0	Not Found
14	30.1.2…	1433	SQL Server			

如果导入域名 IP 列表

```
1 www.qq.com:157.255.192.44,61.241.44.148,220.194.111.148,220.194.111.149
2 www.baidu.com:163.177.151.109,163.177.151.110,61.135.169.125,61.135.169.121,182.61.200.6,182
```

扫描结果就如下显示，域名栏会显示于该 IP 关联的 1 个或多个域名。

扫描结果 结果导出 发送至WEB指纹

序号	IP	域名	端口	默认端口类型	banner	特征	标题
	detect 9 ports open						
1	163.177.151.109	www.baidu.com,	80	HTTP	http	BWS/1.1	百度一下，...
2	220.194.111.149	www.qq.com,	80	HTTP			
3	182.61.200.6	www.baidu.com,	80	HTTP	http	BWS/1.1	百度一下，...
4	182.61.200.6	www.baidu.com,	443	HTTPS	https	BWS/1.1	百度一下，...
5	182.61.200.7	www.baidu.com,	443	HTTPS	https	BWS/1.1	百度一下，...
6	61.241.44.148	www.qq.com,	80	HTTP			
7	61.241.44.148	www.qq.com,	443	HTTPS			
8	61.135.169.125	www.baidu.com,	80	HTTP	http	BWS/1.1	百度一下，...
9	61.135.169.125	www.baidu.com,	443	HTTPS	https	BWS/1.1	百度一下，...

6、其他

扫描结果 结果导出 发送至WEB指纹

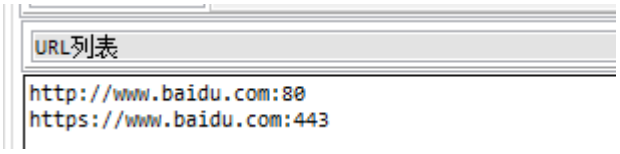
结果导出：可将扫描结果导出为 csv 格式文件。

发送至 WEB 指纹：将结果 banner 识别为 http 或 https 的格式化为合规 URL 发送到 web 指纹的“URL 列表”里。(左侧有个勾选框，如果勾选了则扫描结束会自动发送并执行 web 指纹扫描)

如下会格式化为 https://47.103.195.0:443

47.103.195.0	443	HTTPS	https	Apache Tomc...

如果域名栏有内容，则发送至指纹扫描的是域名而不是 IP。



最后这里有个未提示的隐藏功能，在扫描结果中，双击指定行的端口栏，会自动打开浏览器访问，无论是否识别为 web 应用。

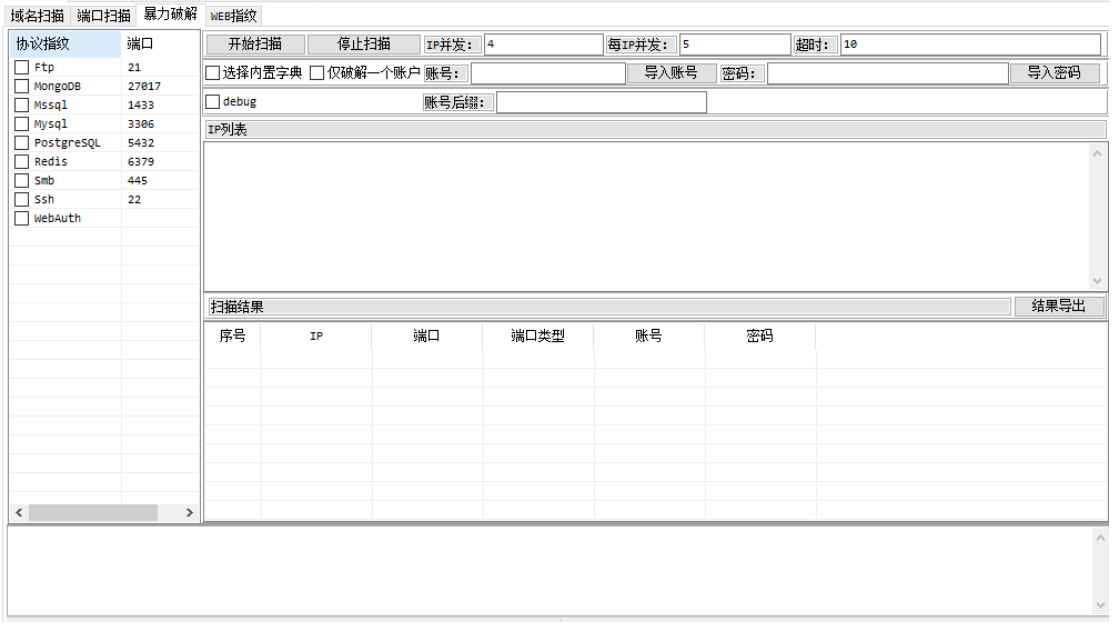
如果 banner 正常识别为 http 或 https，格式和上述发送至 web 指纹的 URL 一致，包括自动使用域名去访问（如果域名栏不为空）。

如果 banner 未识别 http 或 https，统一使用 http 去访问。

2.3 暴力破解

暴力破解模块参考了超级弱口令工具，包括界面和口令库，优化了爆破速度和误判率等。

整体界面如下



配置大致流程：

（1）勾选需要爆破的协议以及端口设置

（2）输入 IP 地址（和端口扫描格式一样）

（3）手动填入单个账号密码，或勾选“选择内置字典”

（4）开始扫描，结果会实时显示

参数详解：

1. 左边协议是目前已支持爆破的，下面介绍下这些协议注意事项

   ftp:用的比较少，暂无问题发现，待续

   ssh:测试速度快，暂无发现误判情况，推荐使用。

   Smb:由于 golang 对 SMBv1 支持不好，winxp/03 使用 net use 去探测，建议使用单并发，否则容易误判，win7 以上可正常配置爆破。

   mssql: 测试暂无问题发现，能准确识别。

   mysql:测试暂无问题发现，能准确识别，目标可能开启封锁 IP 需注意

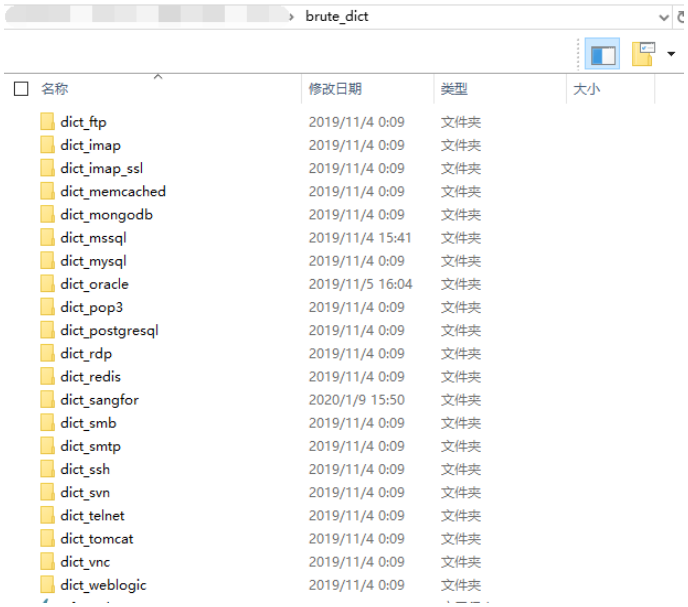
   PostgreSQL:未测试过，待续

   MongoDB:未测试过，待续

   Redis:测试暂无问题发现，可测试未授权。

   WebAuth: 用于 basic auth 认证，比如 tomcat 管理后台爆破。

2. 并发设置里，IP 并发即同时爆破几个 IP，每 IP 并发即每个 IP 并发爆破数。目前可手动设置单用户密码，或勾选“选择内置字典”，会自动调用 brute_dict 目录下的不同协议字典。（请勿修改字典名）

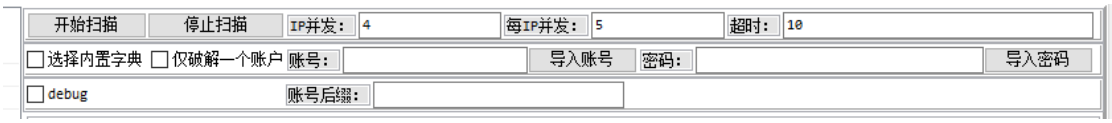


账号后缀用于域用户爆破或者邮箱爆破，会自动和账号组成如 system@163.com 的用户名。

3. 仅破解一个账户表示，如果指定 IP 的指定协议已经爆破出用户密码，则不再进行该 IP+协议的爆破。

4. 账号密码除了选择内置，还可点击导入按钮，手动导入自定义的账号密码表。

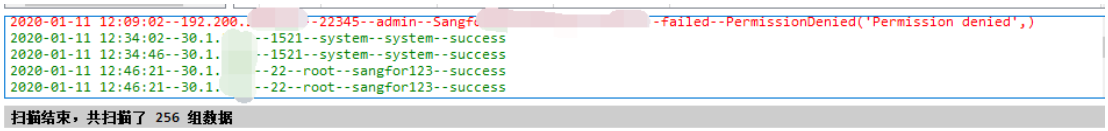
5. Debug 用于在最下面输出框里输出详细错误日志用于排查，默认不开启。



6. 扫描结果如下显示

序号	IP	端口	端口类型	账号	密码
1	30.1...	22	Ssh	root	sangfor123
2	30.1...	22	Ssh	root	sangfor123

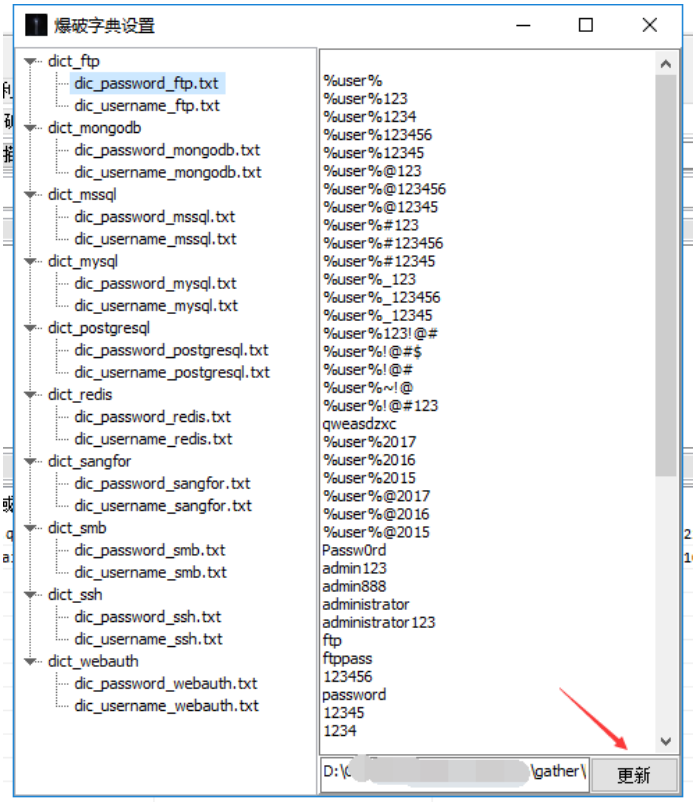
7. 最下面的窗口用于回显成功或者失败信息，主要用来判断失败原因，目前刚做，后续还需要优化这块日志显示，通过“debug”开启日志打印。



整体示例如下（最新版本并发调整为 IP 并发*每 IP 并发）

扫描结果可通过如下按钮导出为 csv 文件。

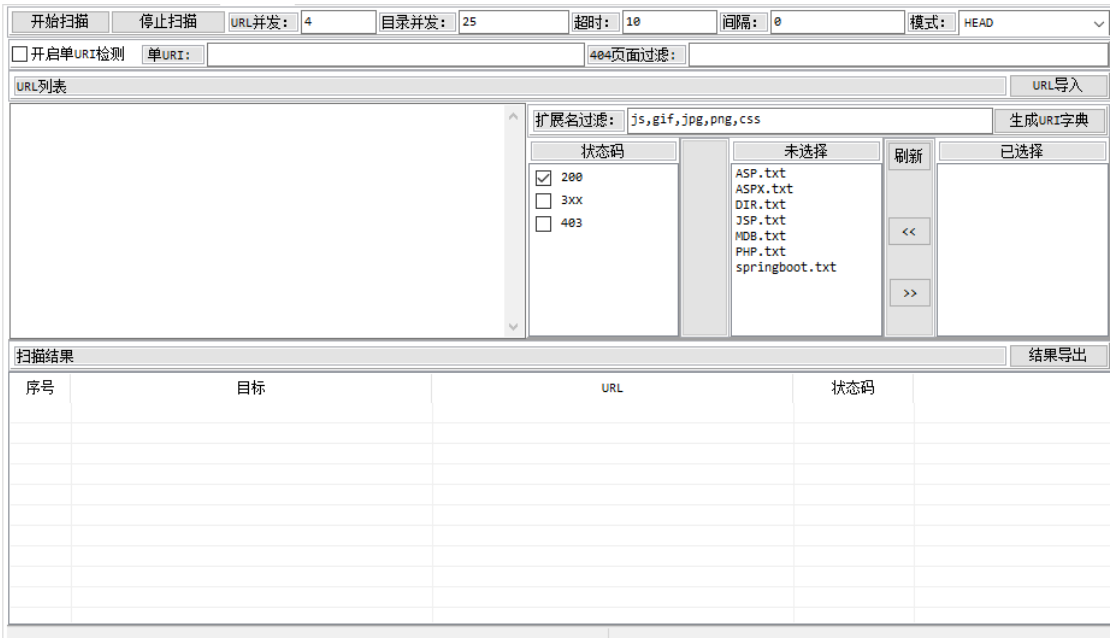
内置字典操作可通过” 设置-爆破字典设置”打开操作，界面如下



双击指定字典名称就可显示内容，修改完点击更新即可。

2.4 目录扫描

界面如下



该模块主要参考御剑模式做的，基于御剑增加了一些功能，如生成 URI 字典、仅扫描单 URI、自定义 404 页面特征等等。

下面一一介绍各功能。

2.4.1 设置说明

1．首行参数

这个和其他模块和 web 指纹扫描差不多，只是多了一个模式，HEAD/GET，扫描使用的 HTTP 方法不同。

2．单 URI 检测

当勾选”开启单 URI 检测”，并输入指定 URI，那么就不会读取 URI 字典，而是仅使用一个路径批量扫描，这个功能可用于批量检测某一 CMS 特征，或者是否存在漏洞路径。

404 页面过滤：输入正则表达式，主要用于某些网站错误页面也是返回状态码 200，在该处输入错误页面的正则表达式，则当正则匹配返回内容时，判定为 404，可有效过滤错误页面。注意：仅在 GET 模式下生效。

3．URL 输入及 URI 字典

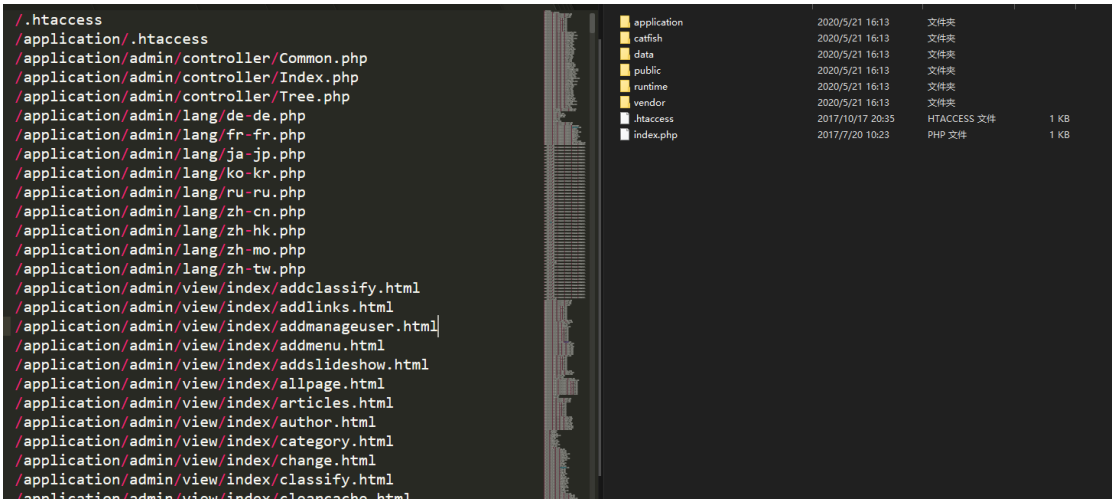
URL 列表，严格按照 http(s)://x.x.x.x/格式。

生成 URI 字典：这个功能用于特定 CMS 字典生成，比如判断目标可能为某个 CMS，需要扫描敏感路径，如果是开源的，下载源码后，可根据源码路径自动生成字典进行扫描。

扩展名过滤： 默认过滤 js,gif,jpg,png,css 这 5 个后缀不加入字典，清空则所有文

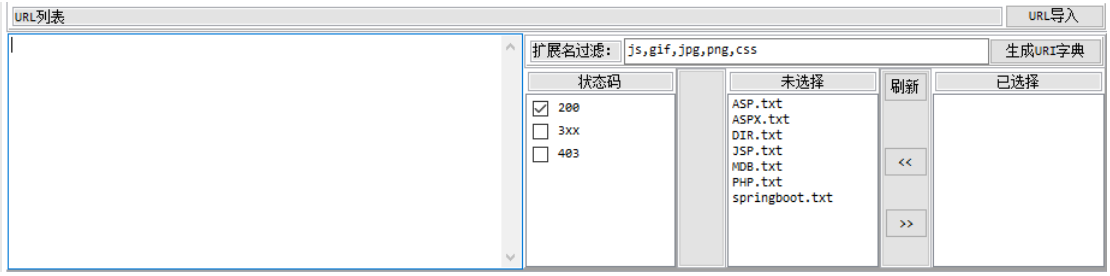
件都加入字典。

生成字典示例：



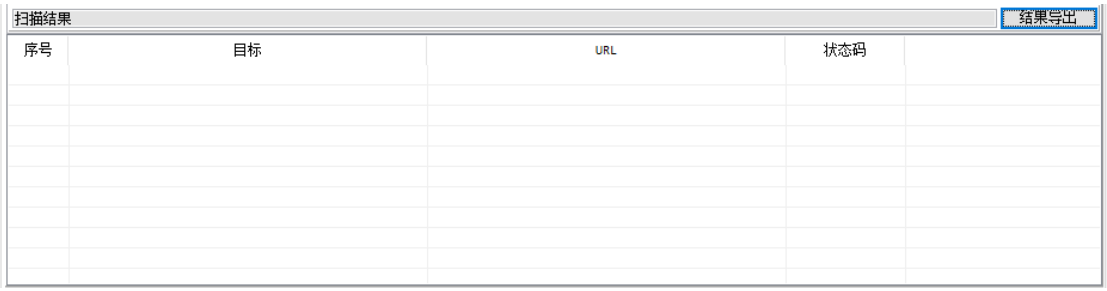
状态码：表示返回状态码有效需要打印显示的。

URI 字典列表：可单选、多选移动，刷新按钮可刷新当前字典列表信息。并且在列表栏右键即可打开字典路径进行调整。

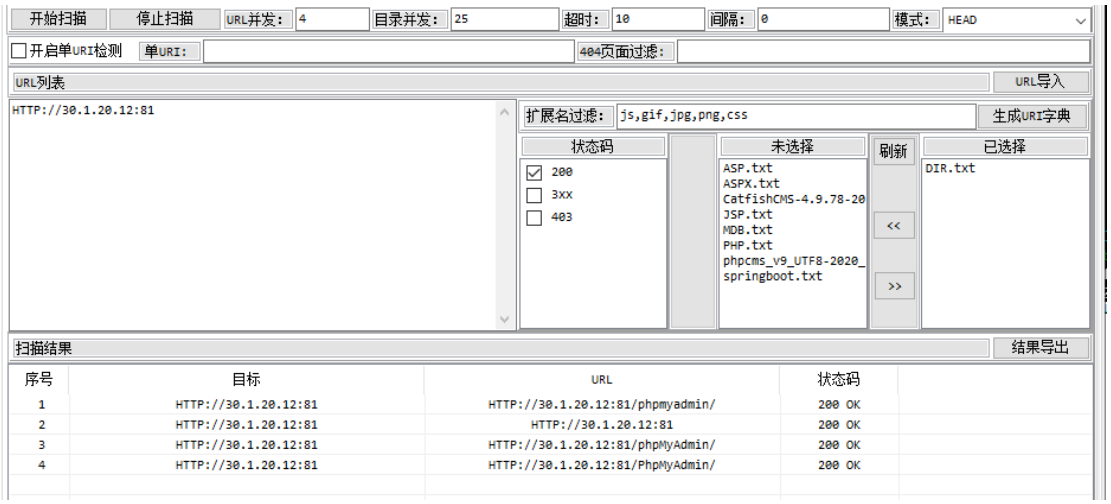


4．扫描结果

扫描结果很简单，就不多讲了。双击 URL 栏会自动打开默认浏览器访问。结果导出功能暂时未做。

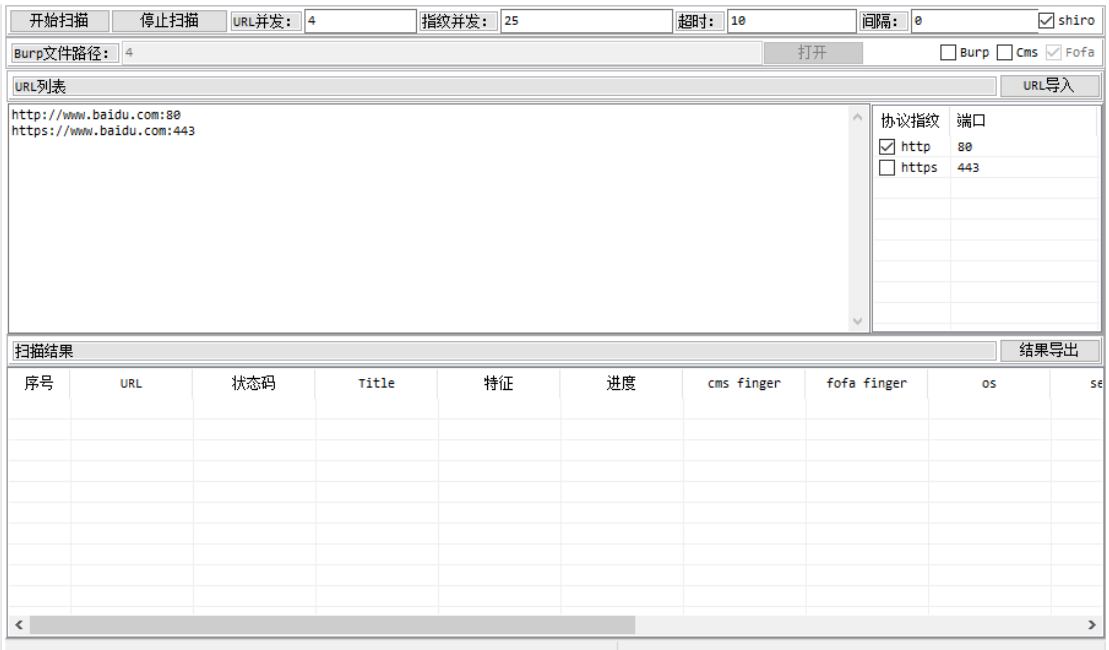


2.4.2 举例



2.5 web 指纹

界面如下



Web 指纹扫描，用于识别目标站点使用的框架及特征等。主要依赖 fofa+cms 指纹库，使用 sqlite 进行存储和读取，sqilte 文件为 cms_finger.db，fofa 和 CMS 指纹各 2000 条左右，具体可使用 navicat 等工具打开查看。

有三种解析方式：

Burpsuite 导出文件解析：burpsuite 代理记录的请求可导出成 xml 进行解析判断指纹。

Fofa 指纹扫描：只进行一次 URL 请求，根据返回结果的各个部分进行指纹识别，指纹规则与 fofa 搜索规则格式一致。

Fofa+传统指纹扫描：传统指纹会构成不同 URL 路径请求目标，每次请求返回结果还会经过 fofa 指纹识别。

CMS 指纹库

对象 cms @main (cms_finger) - 表

开始事务 文本 筛选 排序 导入 导出

finger_id	cms_name	path	match_pattern	options	hit
1	08cms	/images/admina/arrow.jp	4d31afa41252d32d8a9ae	md5	0
2	08cms	/images/admina/arrow.jp	6ad561345b55814902d01	md5	23
3	08cms	/images/admina/logo.pn	413946cd43e990aa55133	md5	4
4	08cms	/images/admina/logo.pn	db113c0f641da45947a37	md5	0
5	08cms	/images/admina/sitmap0.	71cc4f949f5a50008048e8	md5	22
6	08cms	/images/admina/sitmap0.	e0c4b6301b769d596d183	md5	0
7	1039_jxt	/css/stuselect/index.css	height: 35px; position: rel	md5	0
8	1039_jxt	/images/jxt_logo.gif	d9a4ebcfc4eec9120f8812	md5	1
9	1039家校通	/images/jxt_login_bg.gif	21224af1da24ba961ed4c	md5	0
10	1039家校通	/images/jxt_logo.gif	8adfb204fc17450fa124ccf	md5	0
11	3gmeeting	/images/an_on.jpg	f0c48c3ab92948f55bf328	md5	0
12	3gmeeting	/images/download.jsp	6ac63e4862841ebb76a19	md5	0
13	3gmeeting视讯系统	/images/download.jpg	816b4187721f32088960ef	md5	0
14	51Fax传真系统	/images/password.gif	ecc6bb79200836fd9c08cb	md5	0
15	51Fax传真系统	/images/user.gif	868773eab4863759e70b8	md5	0
16	53kf	/new/Client/Css/style.css	worker_info dt	keyword	0
17	53kf	/new/Client/Image/icon-o	fd8ee64400da8b5bafa0e	md5	0
18	53kf	/new/Client/Script/webCo	webCore.min.js	keyword	0
19	5UCMS	/admin/images/style.css	1f77c198658bcaf9f0df827	md5	22

Fofa 指纹库

对象 | cms @main (cms_finger) - 表 | fofa @main (cms_finger) - 表

开始事务 | 文本 | 筛选 | 排序 | 导入 | 导出

id	name	keys	type
113	DUclassified	body="assets/DUclassified.css" || title="DUclassified"	app
114	squid	header="squid"	app
115	kangle反向代理	header="kangle" || title="welcome use kangle"	app
116	Varnish	header="X-Varnish"	app
117	Aicache	header="X-Aicache"	app
118	SJSWPS_ OiWPS	header="Sun-Java-System-Web-Proxy-Server" || header="Oracle-iPlanet-Proxy-Server"	app
119	HAProxy_Report	body="Statistics Report for HAProxy"	app
120	PHP	header="X-Powered-By: PHP" || header="PHPSESSID"	lang
121	ASP.NET	header="X-Powered-By: ASP.NET"	lang
122	python	header="python" || header="Django"	lang
123	ruby	header="ruby" || header="WEBrick" || header="Phusion" || header="Mongrel" || header="X-Rack-Cache"	lang
124	jsp	header="jsp" || header="Servlet" || header="JBoss" || header="JSESSIONID"	lang
125	perl	header="perl"	lang
126	nodejs	header="X-Powered-By: Express" || header="pump.io" || header="node.js"	lang
127	ASP	header="X-Powered-By: ASP"	lang
128	ASP.NET Version	header="X-Aspnet-Version"	lang
129	JSP 2.x	header="JSP/2.2"	lang
130	lua	header="Server: lua"	lang
131	jquery	body="jquery"	script
132	bootstrap	body="bootstrap.css" || body="bootstrap.min.css"	script
133	d3	body="/d3.min.js" || body="/d3.v2.min.js" || body="/d3.js" || body="/d3.v2.js"	script
134	jquery-ui	body="jquery-ui"	script
135	yui	body="yui.js" || body="yui.min.js"	script
136	AlloyUI	body="cdn.alloyui.com"	script
137	React.js	body="/react.js" || body="React.createClass"	script

2.5.1 设置说明

1. 首行参数设置

URL 并发和指纹并发：URL 并发表示同时扫描的 URL 个数，指纹并发表示每个 URL 扫描时并发的 cms 指纹数;如上图所示,如果只有 1 个 URL 则同时发起的 HTTP 请求数为 25，如果超过 4 个 URL 扫描则同时发起的 HTTP 请求数为 100。

间隔：主要用于 waf 场景，为每个并发请求设置时间间隔，防止过快被 WAF 检测到。

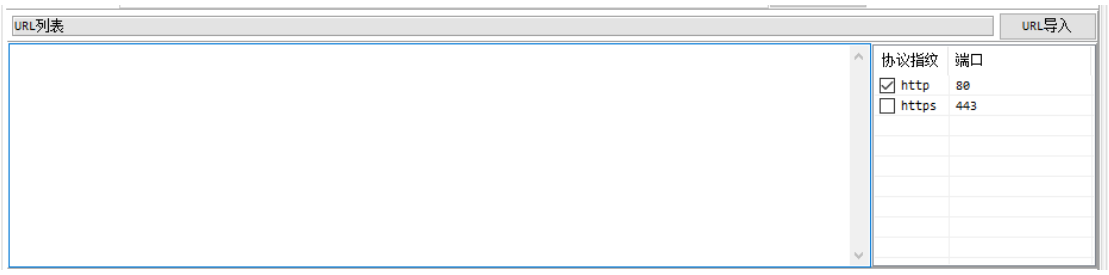
Shiro：该选项框默认勾选，因为 shiro 指纹识别需要设置”Cookie：RememberMe”，暂时不清楚是否会对部分网站造成影响，没做成内置模块。

2. 第二行设置

当勾选 burp 选项时，可设置 burp 的 xml 文件路径，可解析 burp 文件，但同时 cms 和 fofa 扫描无法使用。

Fofa 指纹扫描为默认勾选无法取消，可根据情况是否结合 cms 使用，如果 cms 不勾选，则每个 URL 只发送一次请求；否则每个 URL 会根据 cms 的指纹数发起相应数量的请求。

3．URL 设置



URL 列表写入方式有三种：

（1） 手动设置符合 URL 完整格式的字符串，每行一个 URL，如果开启 cms 指纹扫描，URL 不能设置为具体资源，只能是一个路径，因为后面需要拼接指纹库里的路径。

（2） 通过端口扫描识别为 http 或 https 的结果发送，写入该编辑框。

（3） 如果 txt 文本文件导入，也是一行一个 URL，这里需要注意的是，导入的数据可以没有 URL 前缀，即 http://或 https://，如果没有前缀，会根据右侧协议勾选情况自动加上前缀；

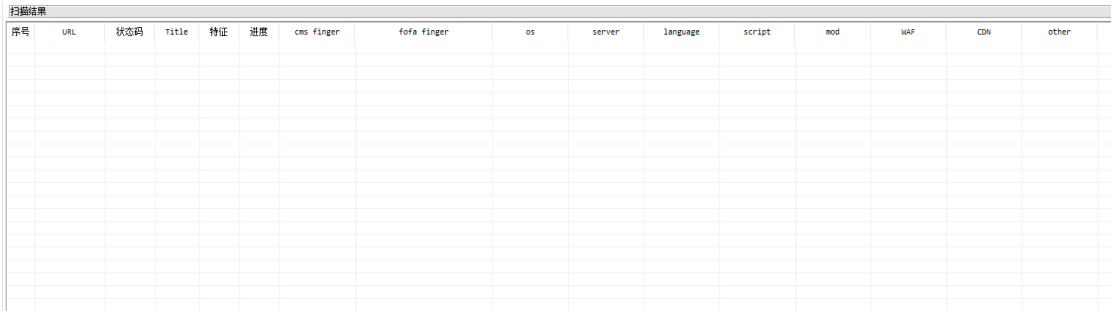
如只勾选 http，并设置端口为 80-81，则会将导入的 host 添加上 http 前缀，并设置端口为 80 和 81，https 也一样。

如果有前缀则不做改动。

示例如下：

4．结果显示

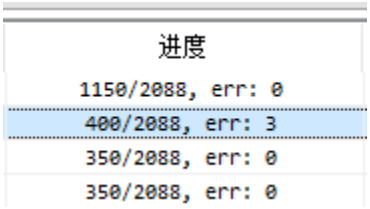
目前可显示扫描结果如下



URL： 不解释。

"状态码","Title", "特征"：这三个用于首次扫描结果解析，用于简单判断目前服务器状态，“特征”和端口扫描里的一样。

进度：用于显示每个 URL 在进行 cms 扫描时的进度，并且会提示请求超时等异常报错的次数；扫描结束会显示 scan over



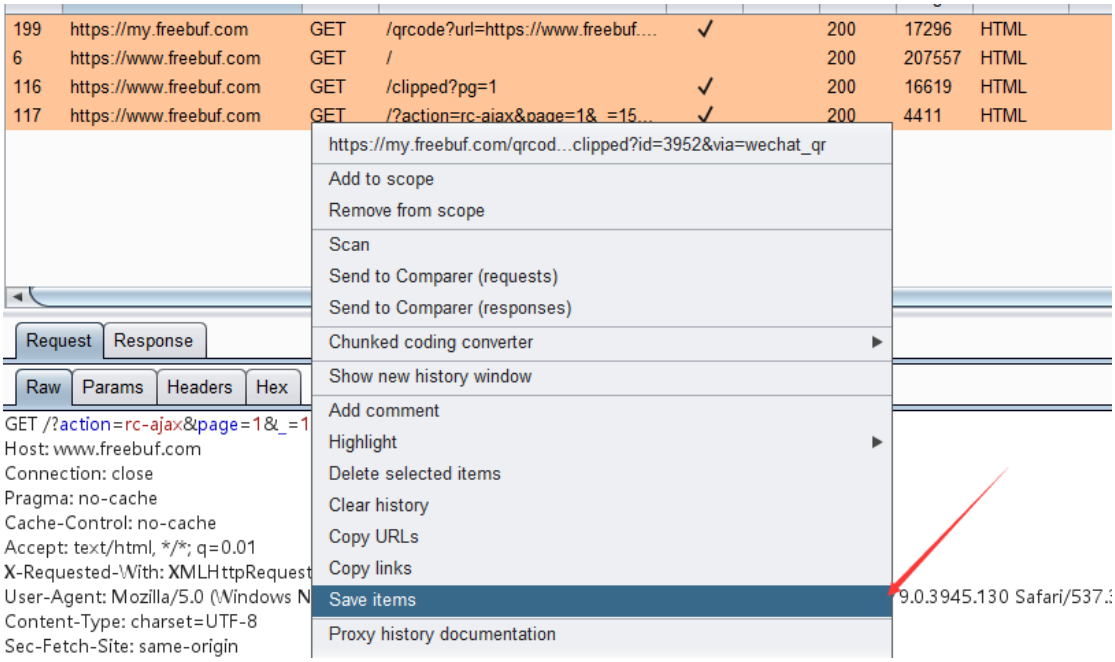
"cms finger"：该项为 cms 扫描结果显示，判断对端为哪种 cms 框架。

其他项：剩下的各项为 fofa 指纹识别结果，sqlite 对不同指纹规则进行分类了，标签含义和结合 fofa 规则库理解，未进行分类的还默认合并于“fofa finger”显示，所以后续还需要再优化。

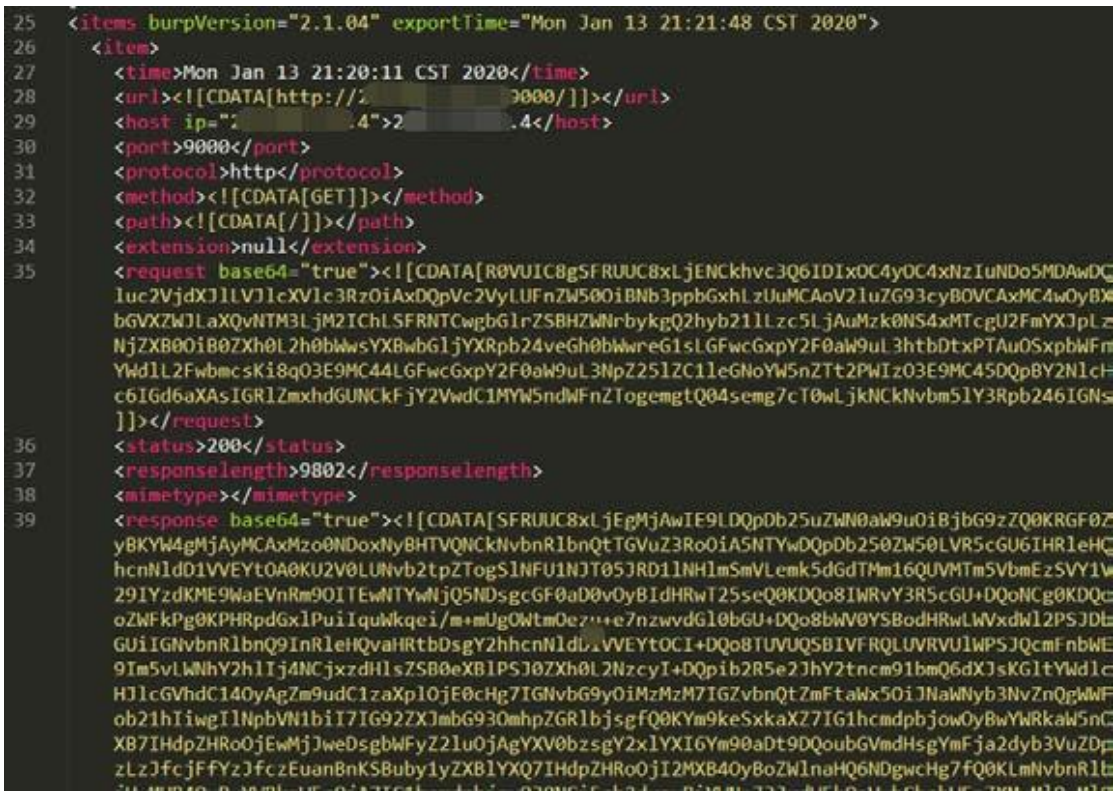
2.5.2 举例

2.5.2.1 解析 BurpSuite 导出文件

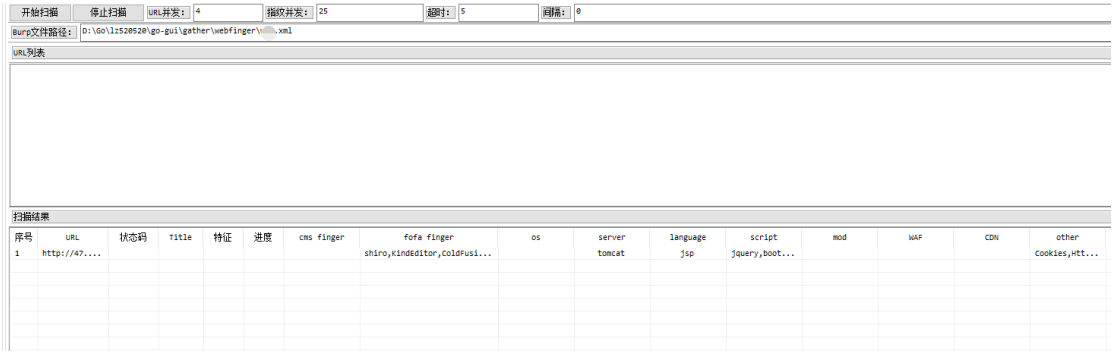
Burpsuite 右键-save items 保存指定域名信息即可，也可在 target 标签页导出。



大概格式如下

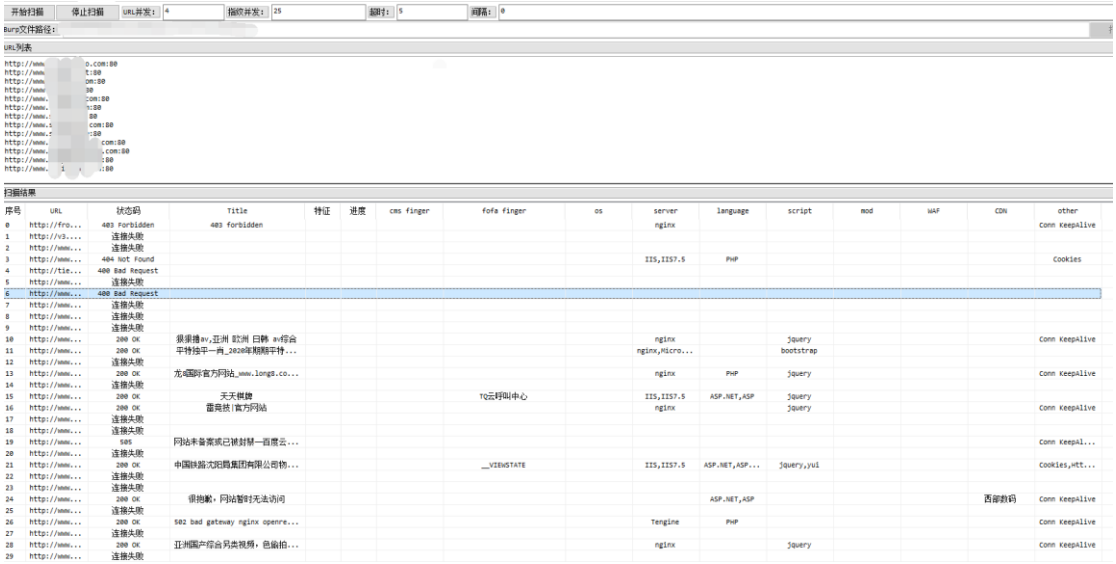


扫描结果如下

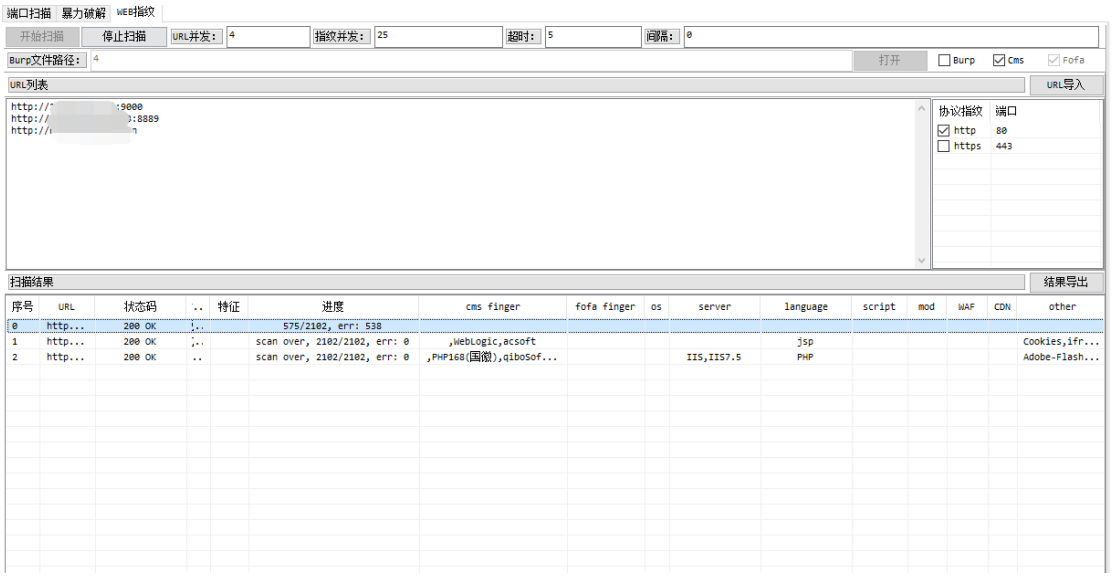


2.5.2.2 Fofa 指纹扫描

每次 URL 仅发起一次请求，结果如下显示



2.5.2.3 Fofa+cms 指纹扫描



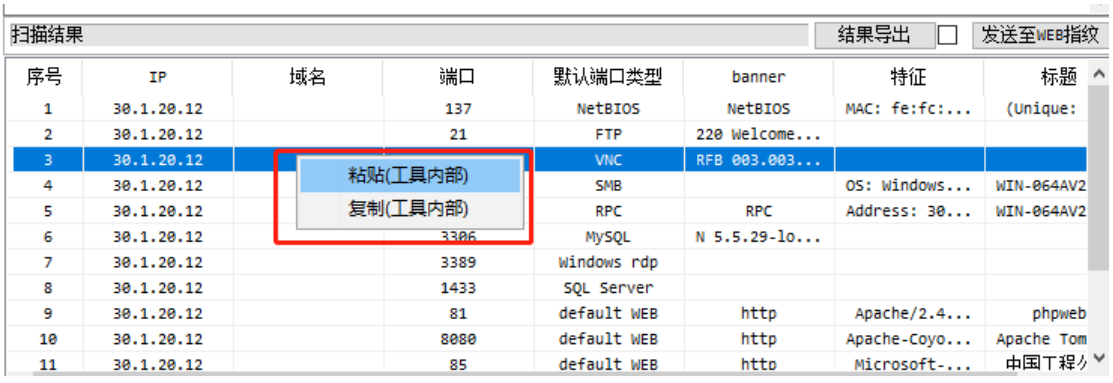
2.6 扩展功能

2.6.1 复制粘贴

信息收集各个模块均支持该功能。

右键结果栏，存在粘贴和复制功能，该功能主要用于工具之间的结果传递。

比如我当前工具挂在 vps 上扫描，扫描结束，我需要在本地工具也能查看结果，则可复制 vps 上的结果，然后粘贴到本地工具里。传统的 ctrl+C 复制是标准格式，无法粘贴。



3 漏洞扫描

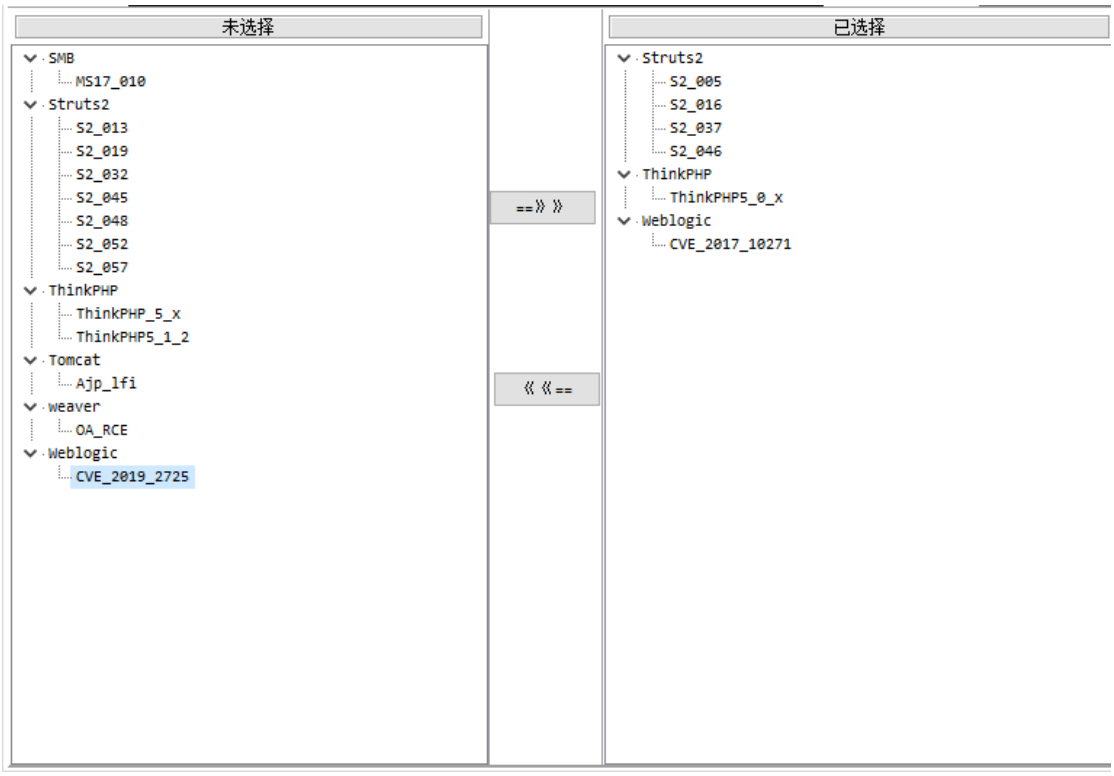
3.1 选项

选项界面如下，目前支持的漏洞扫描如左侧显示，根据不同框架分类了。

第一行 cookie 字段用于某些需要登录才能测试的站点。

可设置代理和 chunked 编码传输，但两者不可同时使用。

漏洞选项区，左侧为未选择，右侧为已选择，默认均在未选择，即不进行该漏洞扫描，双击某个漏洞名称可快捷进行左右移动;如果鼠标按住多选,需要点击中间的箭头进行多项移动。

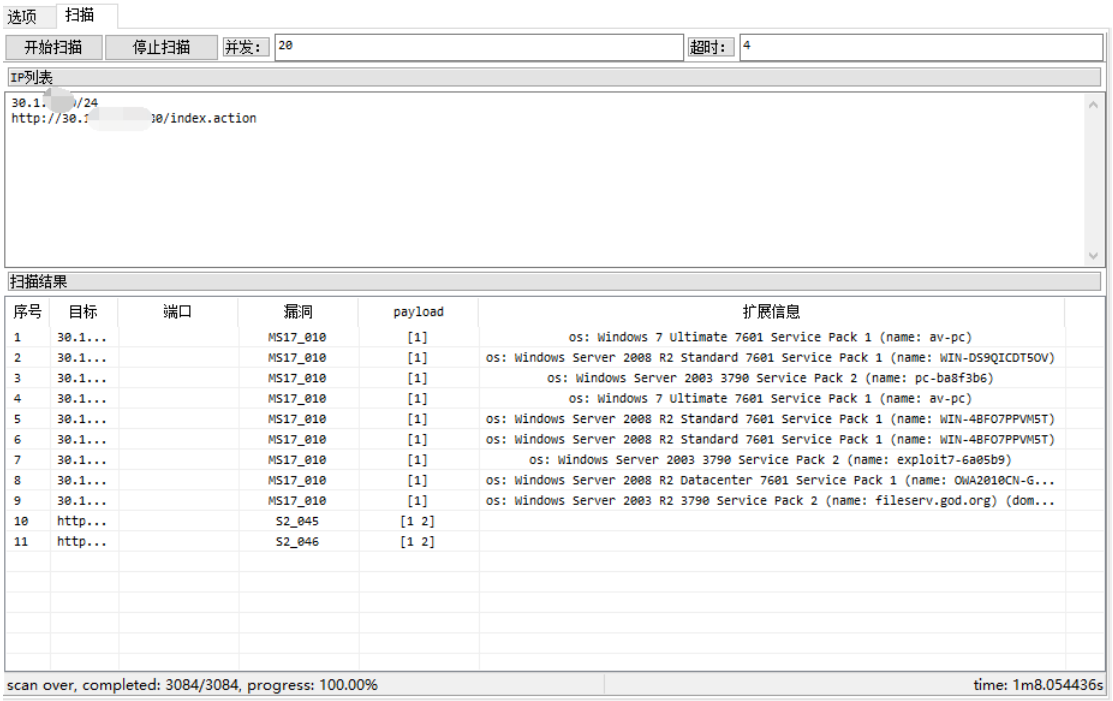


最右侧协议区用于修改协议对应端口，这块提前写好，目前只有 SMB/RDP/AJP 三个协议的漏洞。http/https 的端口在 URL 中就有，这里无需设置。

协议	端口
SMB	445
RDP	3389
AJP	8009

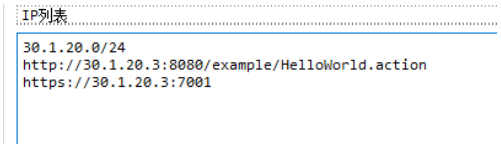
3.2 扫描

扫描页界面如下显示



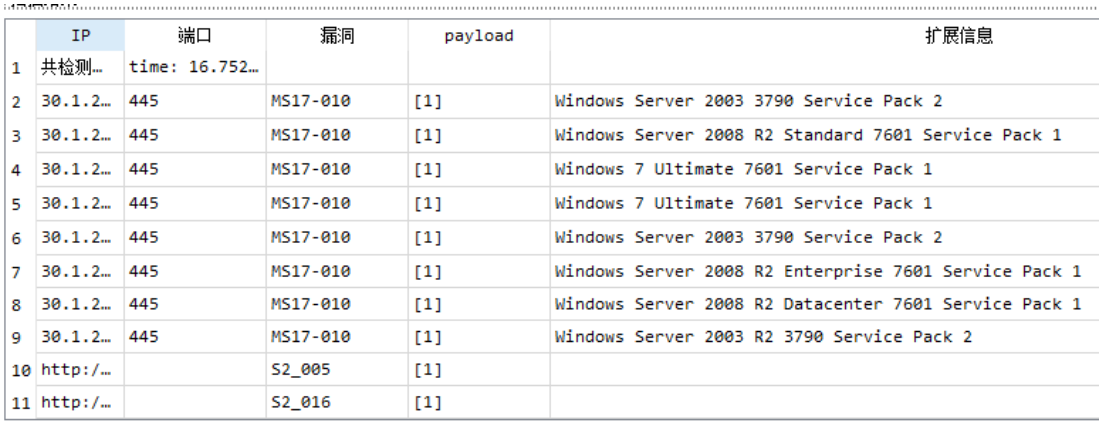
1. 第一行并发超时设置参考端口扫描。

2. IP 列表可设置 IP 和 URL，IP 设置参考端口扫描设置。URL 设置示例如下



3. 扫描结果分为 5 个内容，MS17-010 漏扫会同时识别目标操作系统、主机名和所在域。

   其中 payload 部分是和漏洞利用模块一一对应的，比如这里显示[1]表示为漏洞利用模块的 payload 1。



	IP	端口	漏洞	payload	扩展信息
1	共检测…	time: 16.752…			
2	30.1.2…	445	MS17-010	[1]	Windows Server 2003 3790 Service Pack 2
3	30.1.2…	445	MS17-010	[1]	Windows Server 2008 R2 Standard 7601 Service Pack 1
4	30.1.2…	445	MS17-010	[1]	Windows 7 Ultimate 7601 Service Pack 1
5	30.1.2…	445	MS17-010	[1]	Windows 7 Ultimate 7601 Service Pack 1
6	30.1.2…	445	MS17-010	[1]	Windows Server 2003 3790 Service Pack 2
7	30.1.2…	445	MS17-010	[1]	Windows Server 2008 R2 Enterprise 7601 Service Pack 1
8	30.1.2…	445	MS17-010	[1]	Windows Server 2008 R2 Datacenter 7601 Service Pack 1
9	30.1.2…	445	MS17-010	[1]	Windows Server 2003 R2 3790 Service Pack 2
10	http:/…		S2_005	[1]	
11	http:/…		S2_016	[1]	

下面 payload 有 1,2 即可使用漏洞利用模块的 thinkphp 的 payload 1,2 两个。

	IP	端口	漏洞	payload
1	共检测到 2 …	time: 3.6938…		
2	http://30.…		ThinkPHP5_0_x	[1, 2]
3	http://30.…		ThinkPHP_5_x	[1]

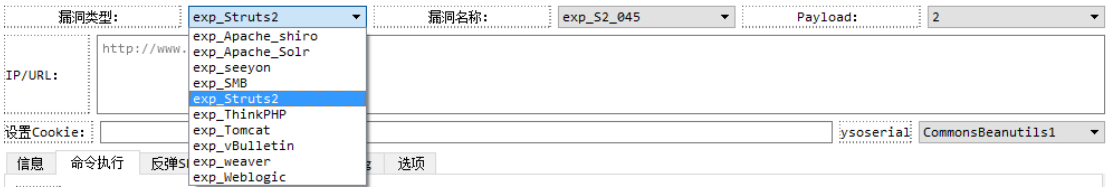
4 漏洞利用

漏洞利用界面如下所示

模块大致分为漏洞设置、信息、命令执行、反弹 shell、文件上传、dnslog、选项这七部分，下面一一介绍。

4.1 漏洞设置

1. 第一行设置漏洞 payload，分别选择漏洞类型、漏洞名称、payload，如 struts2->exp_S2_045->2

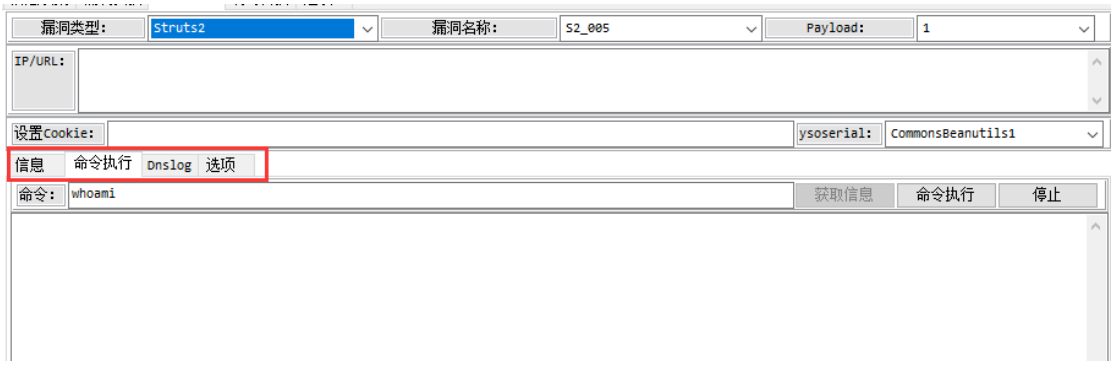


根据选择的 payload 的值，命令执行、反弹 shell、文件上传会根据是否有对应的 payload

开启。

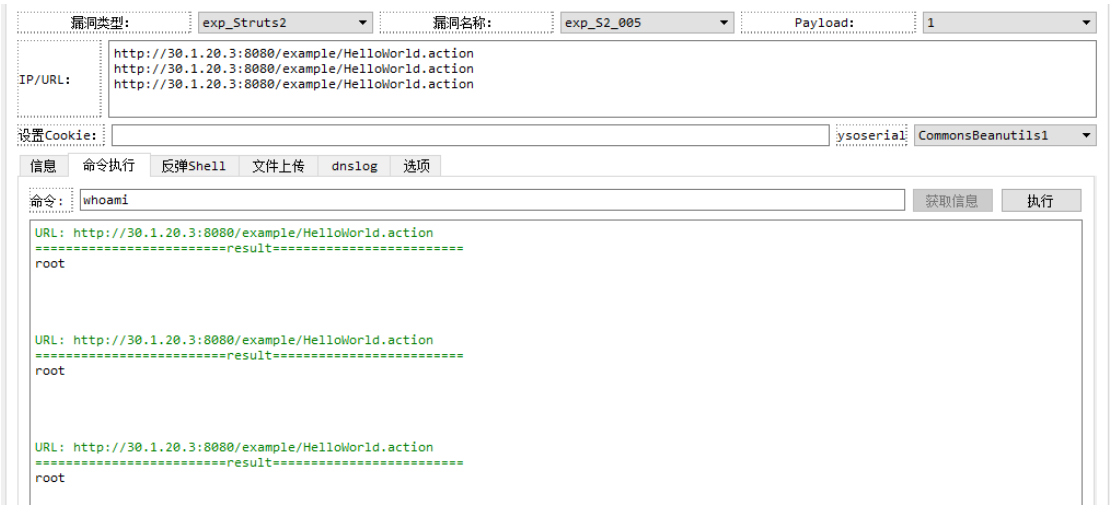
示例：

我当前选择的 payload 仅支持命令执行，则文件上传和反弹 shell 的标签页隐藏不显示。



且根据不同的漏洞名称，注意是漏洞名称会显示不同的信息，用于指导该漏洞的利用。

2. URL 栏用于设置 URL 或 IP，可设置多行用于单漏洞批量攻击，但 IP 只支持单 IP，不支持网段范围之类的；根据漏洞情况可设置 cookie。效果如下



3. 这里额外增加的 ysoserial payload，用于反序列化攻击，像 weblogic 反序列化、shiro 反序列化需要调用 ysoserial。合入工具后不需要 java 环境也可使用。

目前支持的 payload 有以下，后续还会根据漏洞情况增加新的。

```
ysoserialList = [
'CommonsBeanutils1',
'CommonsCollections1',
'CommonsCollections2',
'CommonsCollections3',
'CommonsCollections4',
'CommonsCollections5',
'CommonsCollections6',
'CommonsCollections7',
'CommonsCollections10',
'URLDNS',
'JRMPClient',
]
```

Ysoserial 是一个反序列化漏洞利用 payload 的生成工具。使用 java -jar ysoserial.jar 可查看 payload，具体工具原理可网上查阅资料。

```
$ java -jar ysoserial.jar
Y SO SERIAL?
Usage: java -jar ysoserial-[version]-all.jar [payload] '[command]'
  Available payload types:
一月 11, 2020 9:16:28 下午 org.reflections.Reflections scan
信息: Reflections took 294 ms to scan 1 urls, producing 18 keys and 146 values
     Payload             Authors                               Dependencies

     -------             -------                               ------------

     BeanShell1          @pwntester, @cschneider4711           bsh:2.0b5

     C3P0                @mbechler                             c3p0:0.9.5.2, mchange-commons-java:0.2.11

     Clojure             @JackOfMostTrades                     clojure:1.8.0

     CommonsBeanutils1   @frohoff                              commons-beanutils:1.9.2, commons-collections:3.1, commons-logging:1.2

     CommonsCollections1 @frohoff                              commons-collections:3.1

     CommonsCollections2 @frohoff                              commons-collections4:4.0

     CommonsCollections3 @frohoff                              commons-collections:3.1

     CommonsCollections4 @frohoff                              commons-collections4:4.0

     CommonsCollections5 @matthias_kaiser, @jasinner           commons-collections:3.1
```

4.2 信息

信息模块主要用于介绍当前漏洞，并指导如何进行漏洞利用。包括发布时间、影响范围、漏洞编号、漏洞概述、漏洞利用以及漏洞载荷。

要注意信息内容是根据漏洞名称切换的，所以相同漏洞选择不同 payload 是不会切换的，即漏洞载荷部分只是做简单载荷展示和漏洞的触发点，不会和 payload 对应上。

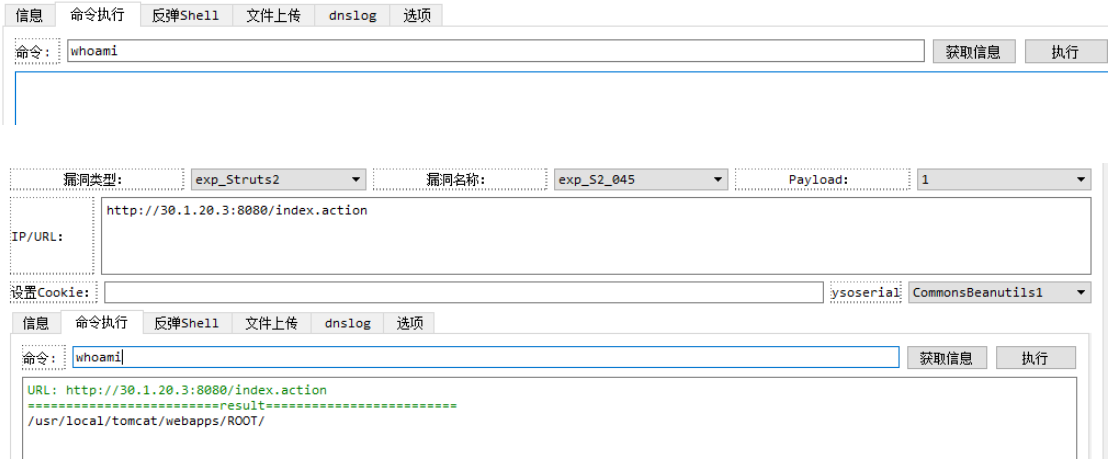
PS：在使用漏洞利用前，尽量先查看信息页说明，因为不同漏洞会有不同注意事项，测试总结都会放在该页。

4.3 命令执行

命令执行页有两个功能

获取信息：根据不同漏洞效果不一样，比如 struts2 用于获取绝对路径，shiro 用于漏洞 key 遍历验证，可参考信息说明。

执行：可以点击执行按钮，或者输入命令后直接回车执行。



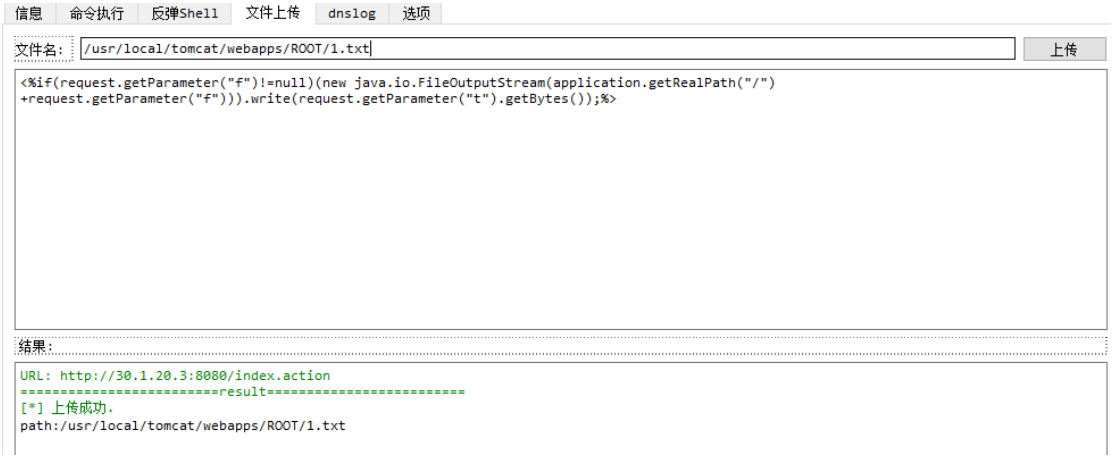
4.4 反弹 shell

主要用于 NC 反弹，目前仅支持 MS17-010 的反弹（已移除），其他 RCE 的反弹 shell 可通过命令执行页来实现。

设置 vps 的 nc 监听 ip 端口即可进行利用。

4.5 文件上传

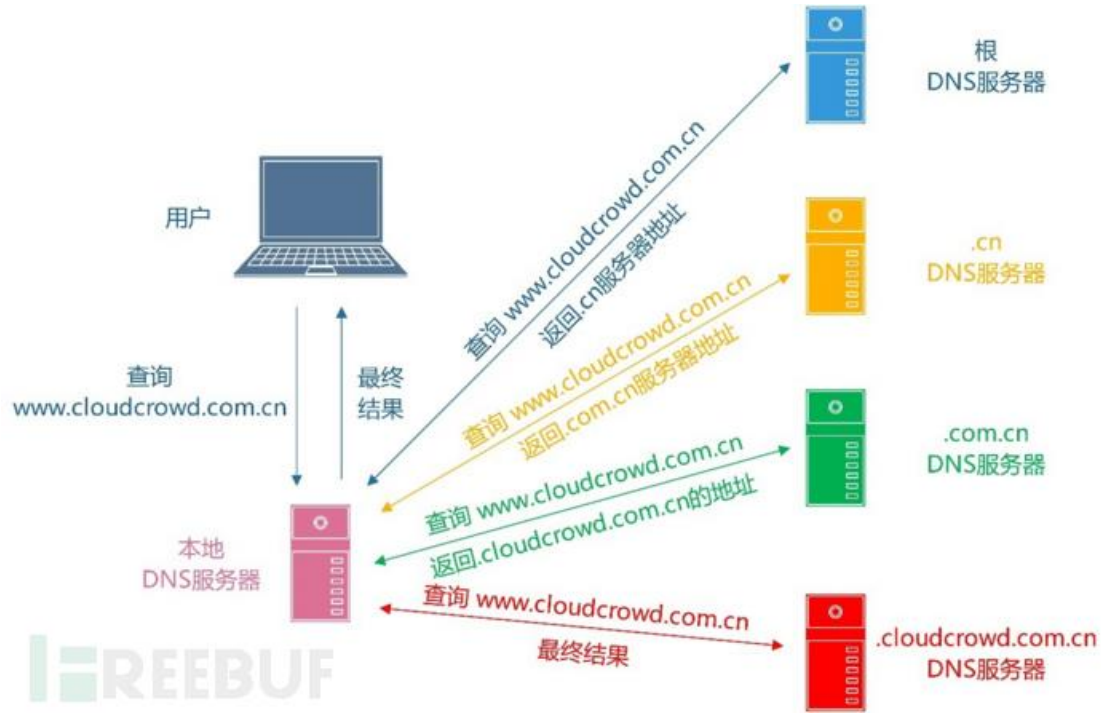
设置上传路径，绝对路径或者相对路径，可自行设置内容。点击上传即可，如下图所示。



4.6 Dnslog

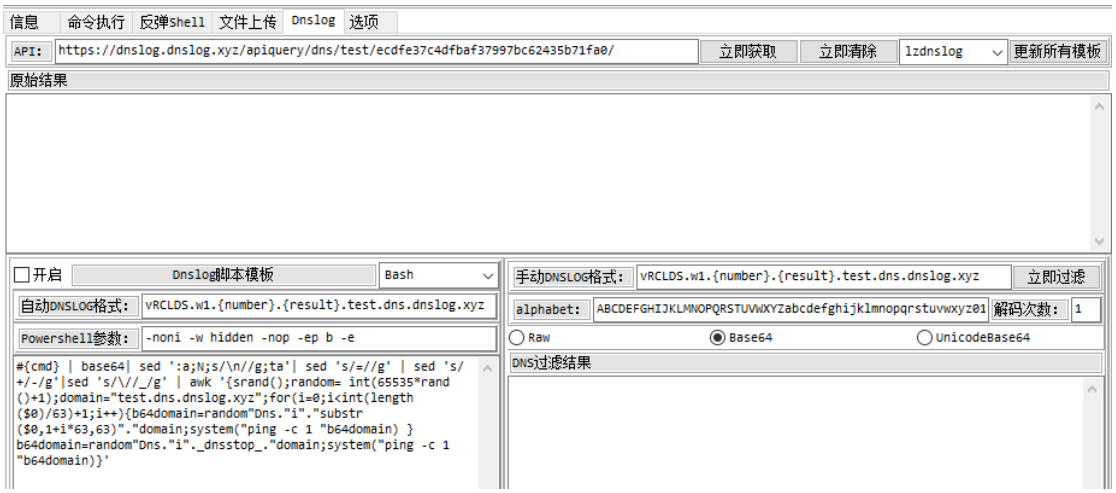
在渗透测试中，SQL 盲注、命令盲注等漏洞是较难利用的，由于无回显，这类漏洞即使存在也显得有些鸡肋。针对此类问题，我们可以使用 DNSLOG 来进行突破。DNSLOG 是一种回显机制，使用者可以通过 DNS 解析日志来读取漏洞的回显。

DNS 的解析是递归与迭代相结合的，下面给出了当我们访问 www.cloudcrowd.com.cn 时，DNS 的解析过程示意图。



其中，红色部分是可控的。我们只需要搭建一个红色部分的 DNS 服务器，并将要盲打或盲注的回显，放到自己域名的二级甚至三级域名上去请求，就可以通过 DNS 解析日志来获取到它们。

该模块如下所示，需要搭建 DNSLOG 平台，通过 API 去调用，目前支持对接我修改的 DNSLOG 平台和 ceye。



填写好查询 API，注意需要点击更新所有模板。

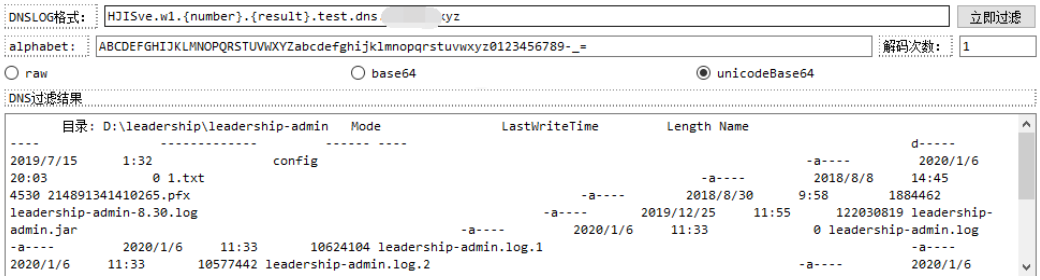
4.6.1 Dnslog 手动回显

填写好查询 API，点击立即获取即可，如下图所示，即可看到回显的 dnslog

并且平台设置了清除 API，会自动替换 apiquery 为 apidel，无需修改 API，点击立即清除即可。

由于考虑 RCE 执行结果可能过长或者执行结果有特殊字符无法通过域名传输，编写了脚本搭配编码分段传输 DNSLOG，回显的 DNSLOG 需要拼接+解码才能正常显示，如上图显示则是 base64 编码后分段传输结果。

预先设置好 dnslog 格式，如下



用{}括起来的为变量参数，{number}表示该位置提取的为编号，{result}表示为分段传输的数据。如果无{number}那么只会解码单个域名的{result}，如果有{number}会将所有匹配的域名进行拼接，然后再解码。

设置解码次数为 1，解码类型为 unicodeBase64，其他两种 raw 表示原生无解码字符

串，base64 就是普通解码。powershell 编码比较特殊而已。还有 base64 的 alphanet 可能也需要修改，因为 base64 编码后可能出现”+/”，这两个符号不能进行域名解析，所以编码的时候会做转换成如“-_“

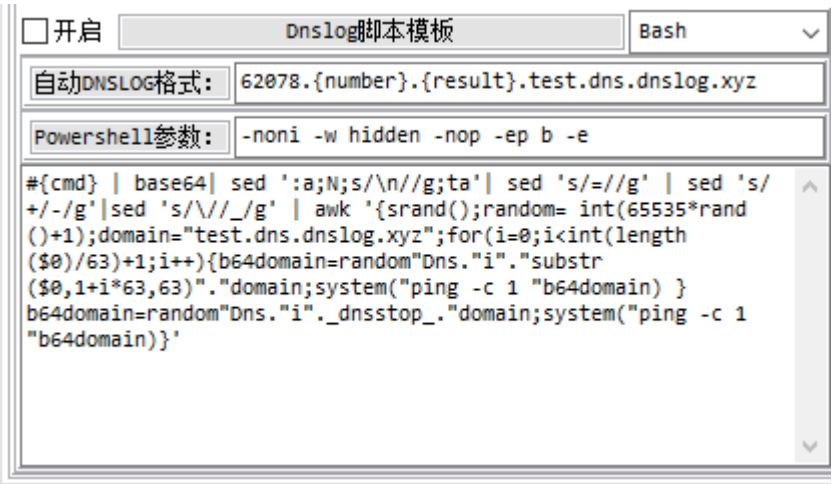
最后点击解码就可以看到上图的内容，可以看到当前目录和目录下文件，就可以进行下一步写 shell 操作了。

4.6.2 Dnslog 自动回显

整个模块左侧为自动回显模块，支持 bash/python/perl/powershell 回显。

使用方法步骤：

1. 勾选”开启”
2. 选择需要使用的 dnslog 脚本类型，这个处决于目标系统支持，选择不同模板，会自动刷新右侧的 alphabet 等参数。
3. 自动 DNSLOG 格式和手动 DNSLOG 格式差不多，通过之前的更新模板按钮即可刷新。
4. Powershell 参数功能适用于 windows 的 powershell，因为有些参数会被杀软拦截，实测只需-e 即可。
5. 最后的 dnslog 模板内容无需调整，是设定好的，只需关注 domain 参数和格式里的后缀是否一致即可。
6. 接下来在 exp 的命令执行模块输入命令执行等待回显即可，目前主要用于 shiro 等 java 反序列化漏洞回显。



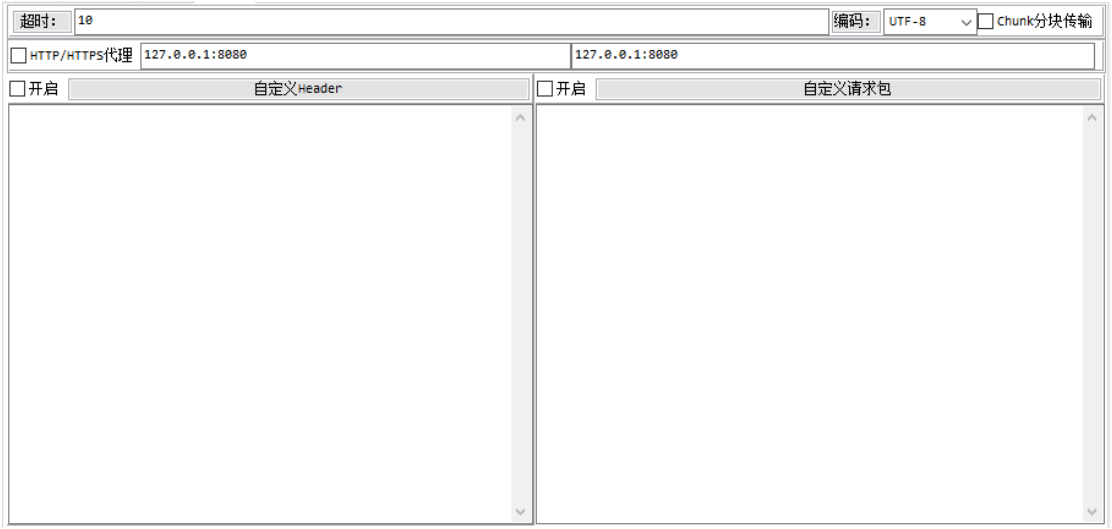
原理：命令执行内容会替换脚本模板的#{cmd}，然后用指定格式 base64 编码，发送，之后程序会间隔发送请求至 dnslog 平台，判断是否返回“_dnsstop_”日志，有则根据手动回显逻辑进行拼接解码并打印在命令执行结果栏。

具体 DNSLOG 的使用方法，可参考以下链接。

https://sec.lz520520.cn:4430/2020/01/337/

4.7 选项

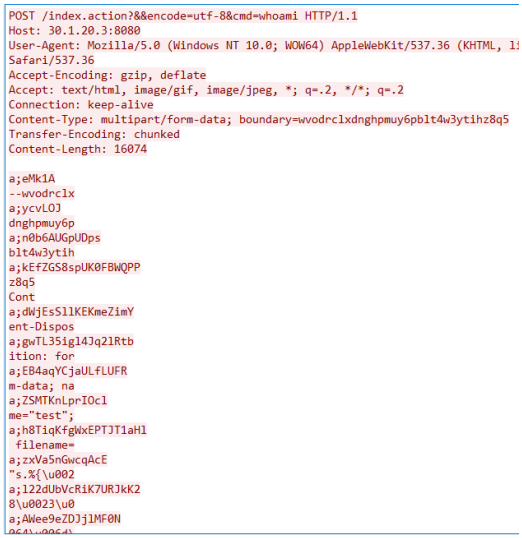
目前支持的选项如下图所示



超时：漏洞利用超时设置，单位为秒

编码：目前支持 UTF-8 和 GBK，目标平台编码不同可能导致回显结果乱码，需要使用正确的编码。

Chunk 分块传输：用于将 http 或 https 的 POST 请求的 body 部分进行 chunk 编码，用于某些 WAF 绕过，特殊情况使用的时候开启就好。效果如下图所示



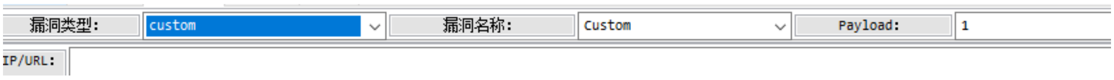
HTTP/HTTPS 代理：设置代理地址勾选即可代理传输，多用于 burp 代理抓包调整 payload 等等。

PS：HTTP 代理和 chunk 编码不能同时使用，会出现超时问题

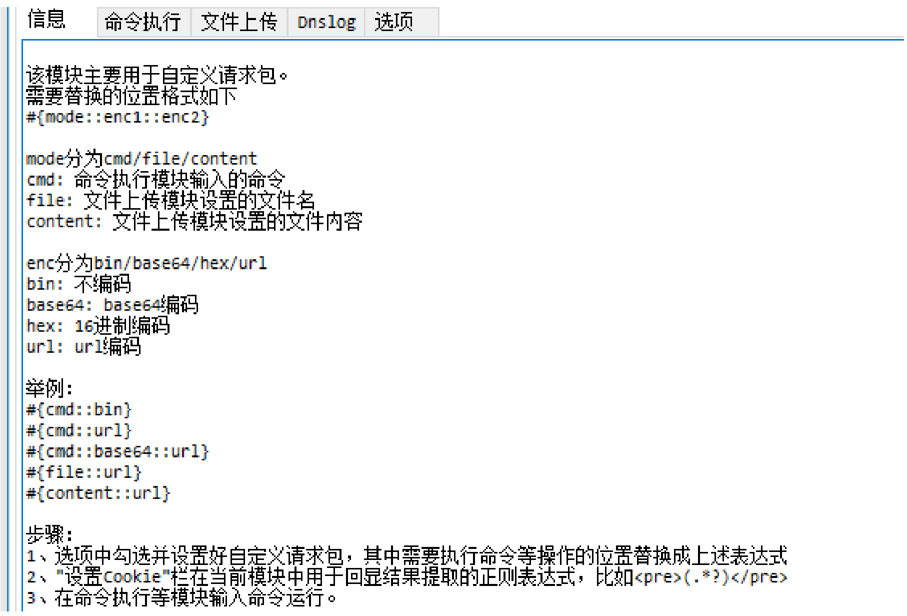
自定义 header：针对某些目标系统,可能 header 字段需要设置多个(不仅仅需要 cookie)，exp 才能执行，勾选开启，设置如下 header 即可。

自定义请求包：该设置用于工具未集成的 exp，可快速设置 payload 测试效果。

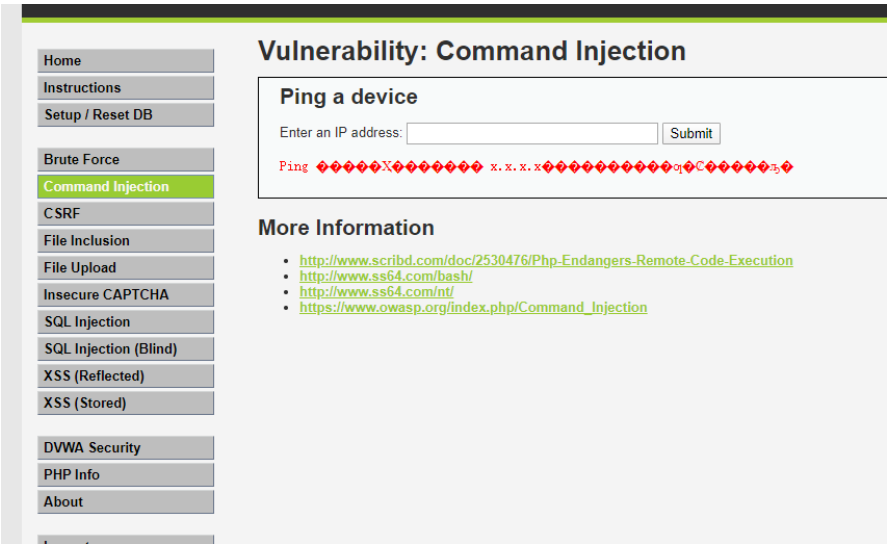
使用该模块之前需设置漏洞类型为 custom。



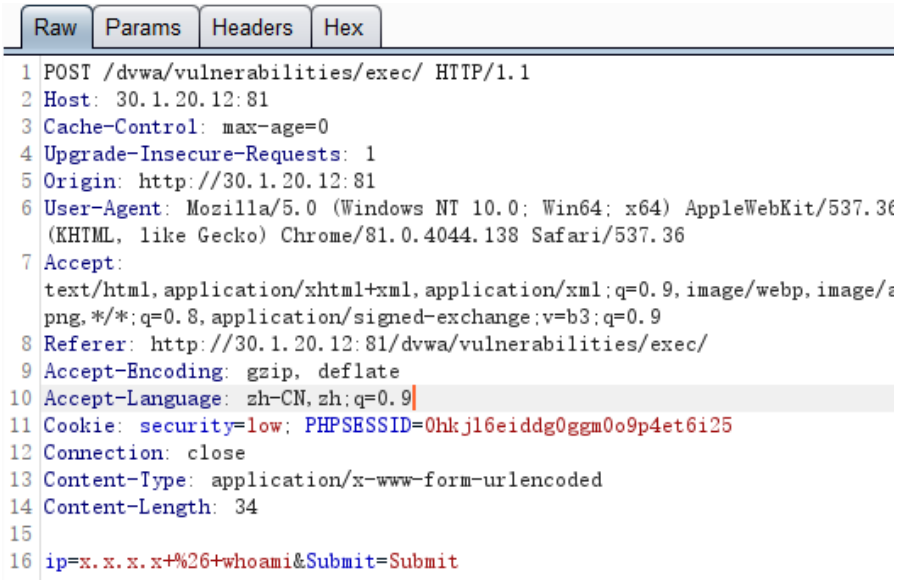
然后使用说明参考信息 tab



这里我举个例子，如 dvwa 的命令执行。



使用 burp 等工具获取完整请求包。

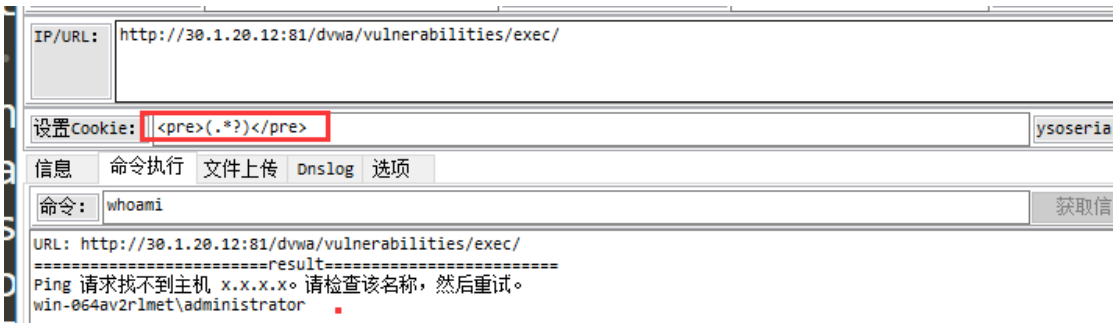


把命令执行部分(whoami)替换成#{cmd::url}，表示参数为命令并进行 URL 编码；复制到自定义请求包窗口，并勾选开启。

然后输入 URL，并在命令执行模块输入命令即可

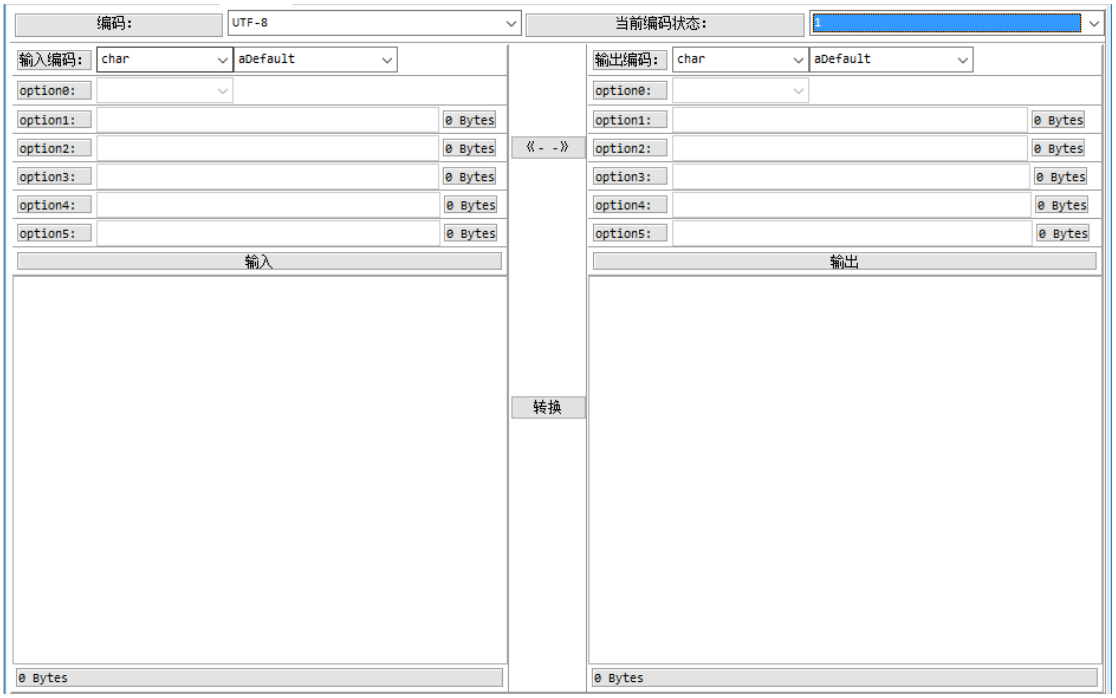
如果结果内容太多需要过滤，可以在”设置 cookie”栏输入正则表达式过滤即可，这个可以

使用 burp intruder 模块自带的 grep extract 功能提取正则表达式。效果如下



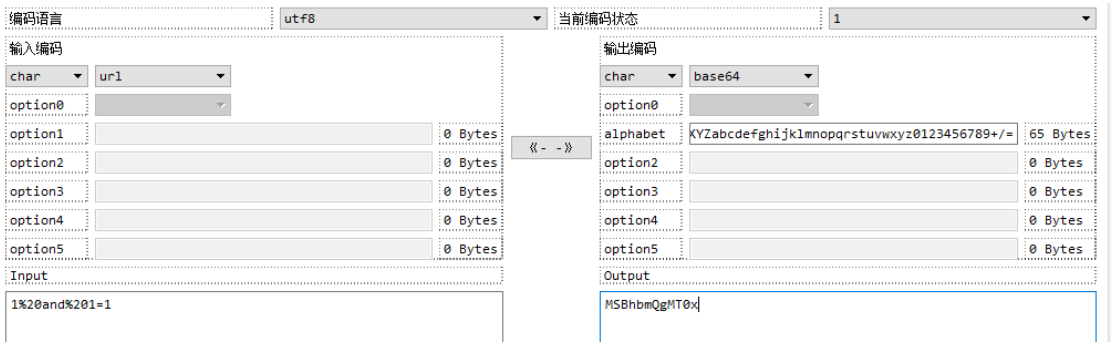
5 编码转换

编码转换界面如下



使用逻辑为，输入编码表示 input 部分对应的编码，输出编码表示 output 部分对应的编码。

举例如左边是 URL 编码字符串，要将明文转换成 base64 编码，那左边输入编码应该设置成 URL 编码，右边输出编码则应该设置成 base64 编码。



可以这么理解，输入设置编码，会将输入先解码，然后再转换成输出编码，如上图先将 URL 编码解码后再转换成 base64 编码。如果某部分为明文，则只需要设置 default 即可。

5.1 整体设置

5.1.1 当前编码状态

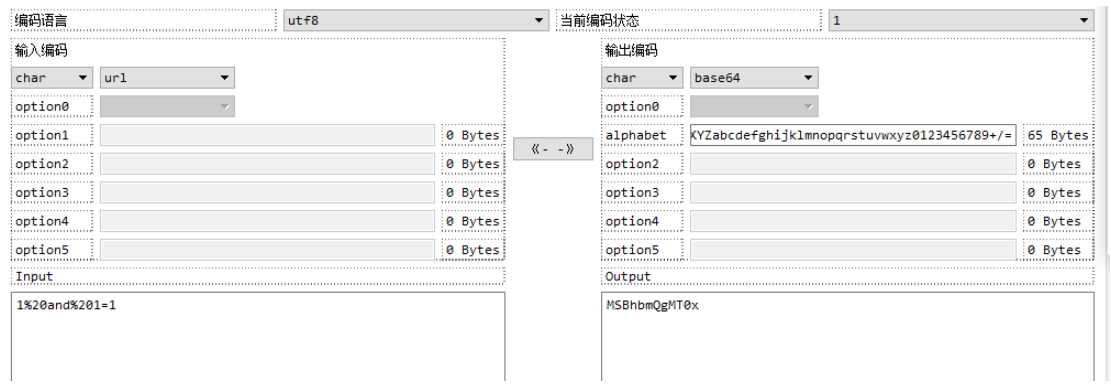
先解释第一行

编码语言：用于不同中文字符串编解码情况。

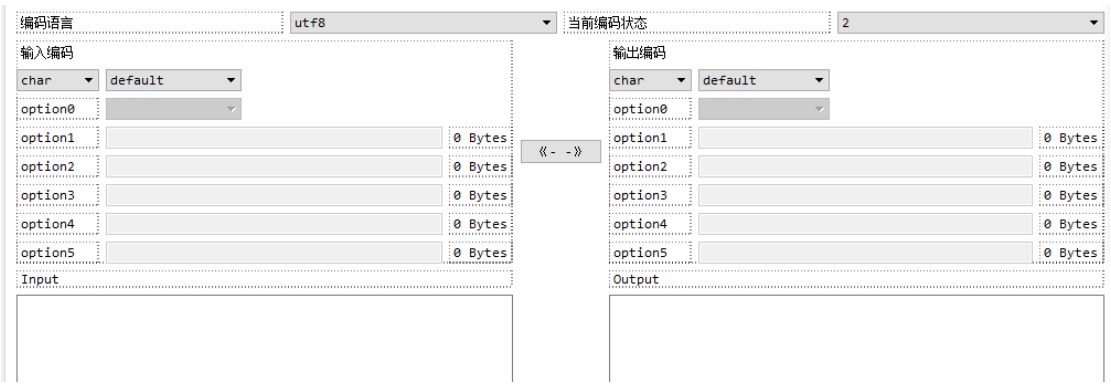
当前编码状态：保存不同编码状态，用于不同编码设置之前快速切换。

示例：

目前当前编码状态为 1，设置的输入编码为 URL，输出编码为 BASE64。



修改成 2 后，是一个全新的编码状态，可设置其他编码，如果想使用 1 中的编码，直接选择 1 快速切换回去，无需重新设置输入输出编码。

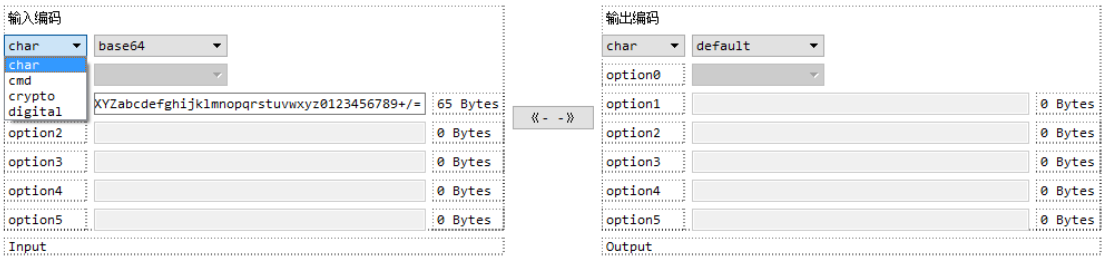


该功能主要方便在不同编码状态中切换，更快捷操作，保存的编码状态包括输入输出编码类型，option 字段参数值，并且输入输出内容也会进行保存。

使用 alt+1、alt+2、alt+3 快捷键，可快速切换状态。

但鼠标焦点（有光标闪烁即可）得在输入或者输出框里，快捷键才生效。

5.1.2 编码设置



输入输出编码均有主编码类型、子编码类型。

主编码类型：char/cmd，编码的整体分类，分别对应字符编码、命令编码、加密算法、进制转换。

子编码类型：

Char:default/ascii/base64/html/reverse/Unicode/UnicodeBase64/url/MDB/UTF7

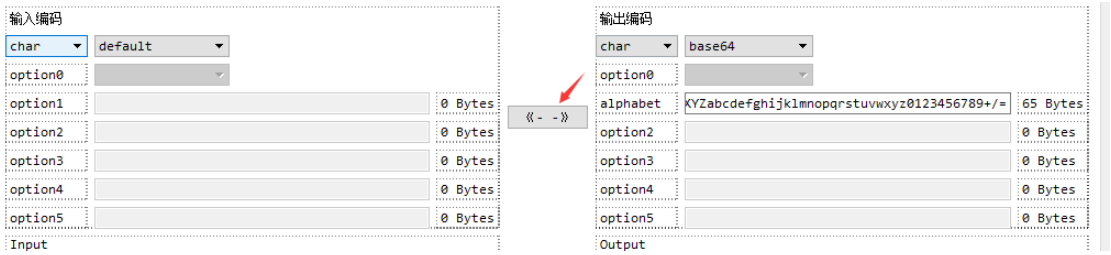
Cmd: javaRuntimeExec/normal

Crypto: AES/MD5

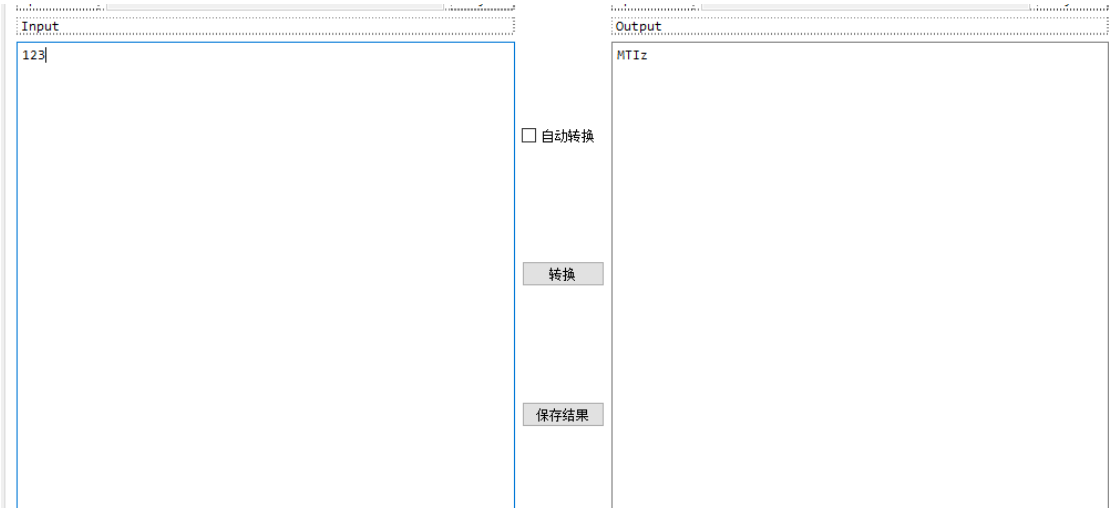
Php: chr

这些编码下面会详细介绍

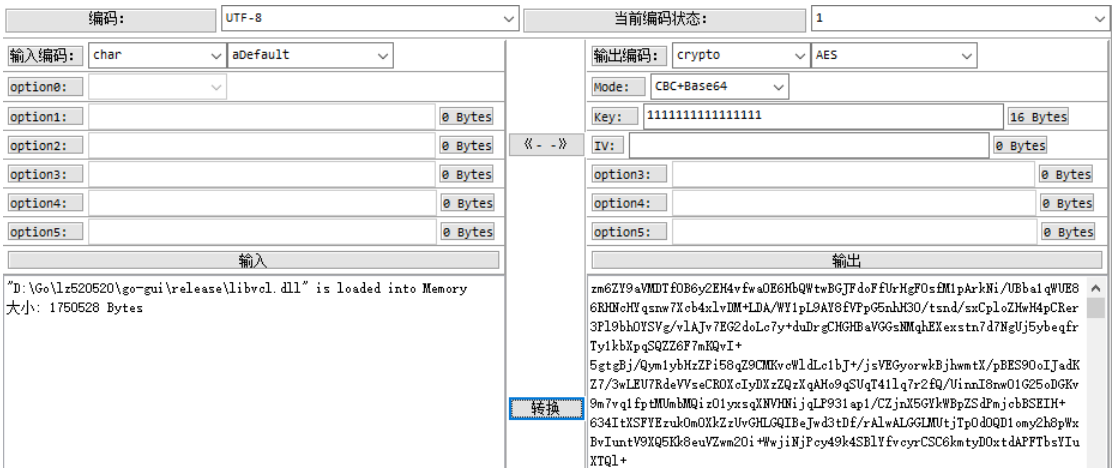
中间按钮用于快速切换输入输出编码，将输入编码切换到输出编码，输出编码切换到输入编码。



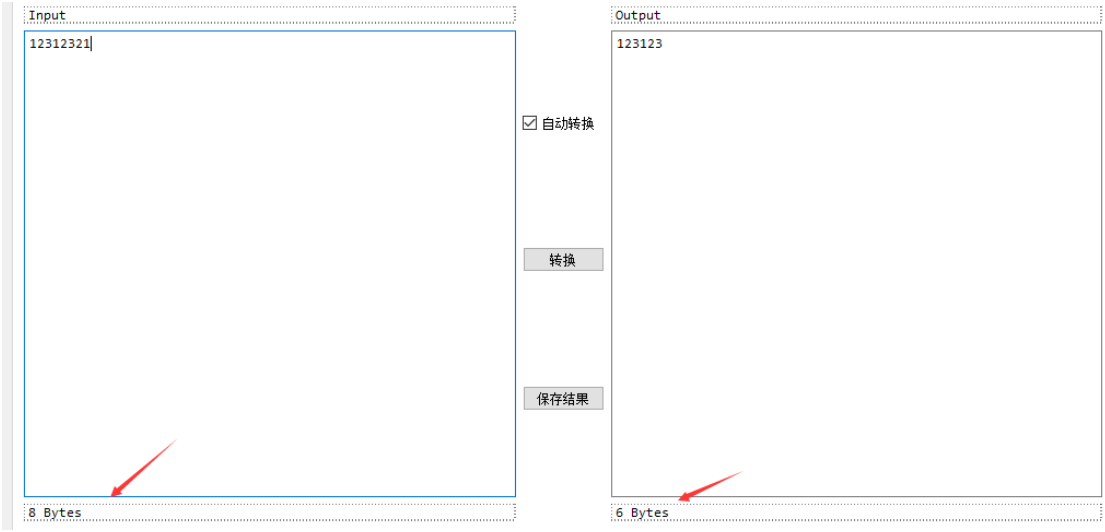
5.1.3 输入输出内容



输入数据点击“转换”按钮可进行编码转换到输出栏，也可使用 ctrl+enter 进行快速转换，但鼠标焦点（有光标闪烁即可）得在输入或者输出框里，快捷键才生效。



新版本支持文件拖拽，但鼠标焦点也需要先放在输入窗口，拖拽加载文件到内存里，其他编码设置没有区别，如果想恢复到手动输入，只需要把输入框清空，即可清除内存加载的文件。



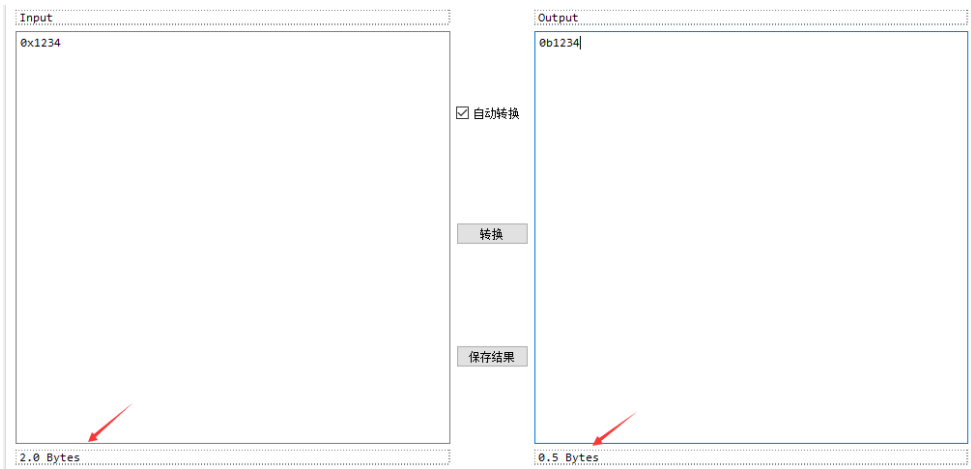
最下方会实时显示当前输入输出框字符串长度。

不同字符串开头长度计算不一样。

0x：0x 开头表示 16 进制，如 0x1234 长度为 2 bytes

0b：表示二进制，如 0b1234 长度为 0.5 Bytes

0o：表示八进制，如 0o123 长度为 1Bytes



5.2 编码使用详解

5.2.1 Char

5.2.1.1 Default

不做任何编码解码转换

5.2.1.2 Ascii

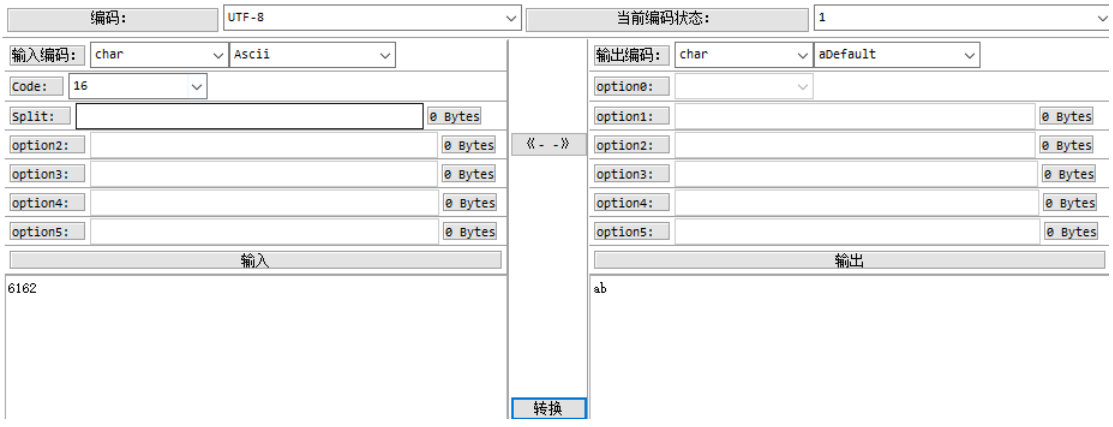
Code 字段可设置 2/8/10/16，表示内容框对应的进制，默认为 16 进制。

设置 16，6162 或 0x6162 解码完为 ab

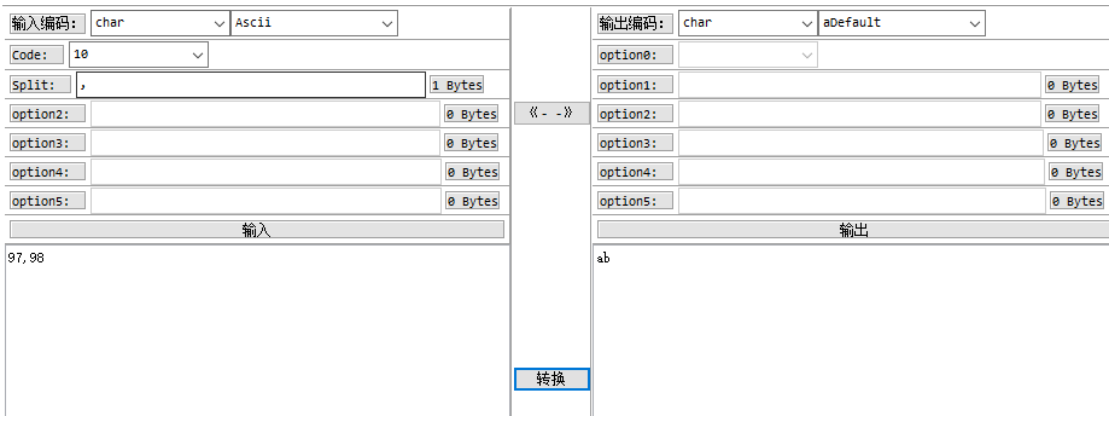
设置 2，01100001 或 0b01100001 解码完为 a。

设置 10，97 解码完为 a。

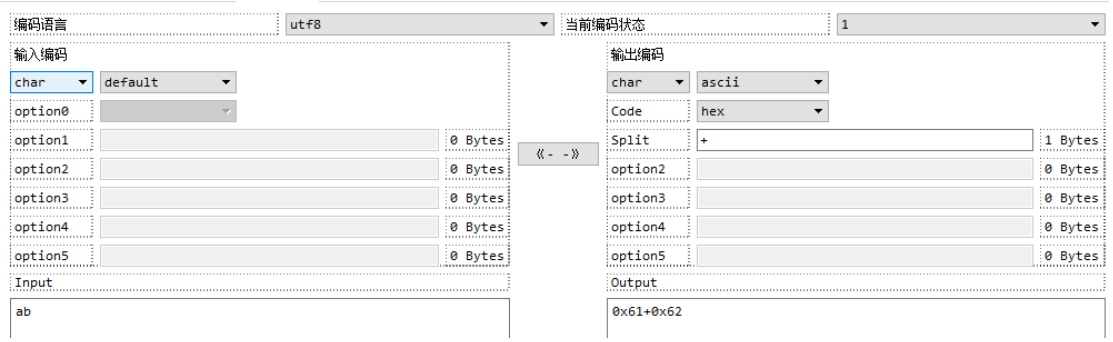
这里可能有人对 ascii 编码和解码的概念弄混，英文字母 a 的 ascii 编码后为 0x61，一般编码后都为进制数；ascii 编码为 0x61，解码完为 a。



Split 字段主要用于一些字符串有分割符，用来去除分隔符拼接。如下图，“97,98”被“，”隔开，需要设置分割符为“,”,才能正常解码。

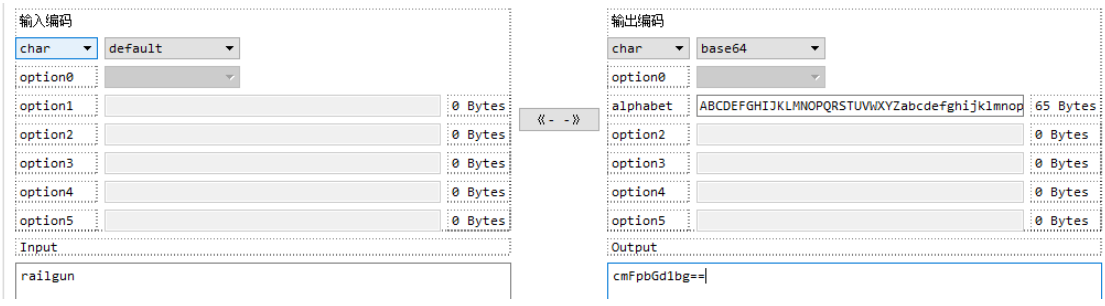


编码也是一样的操作

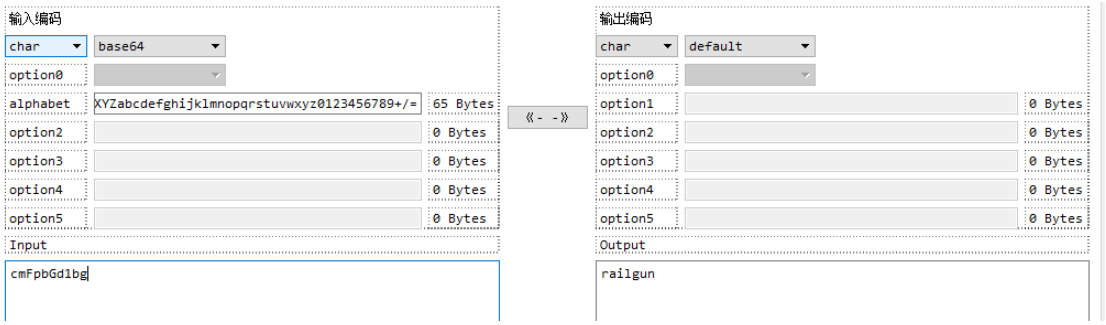


5.2.1.3 Base64

下图为编码，alphabet 可自定义。

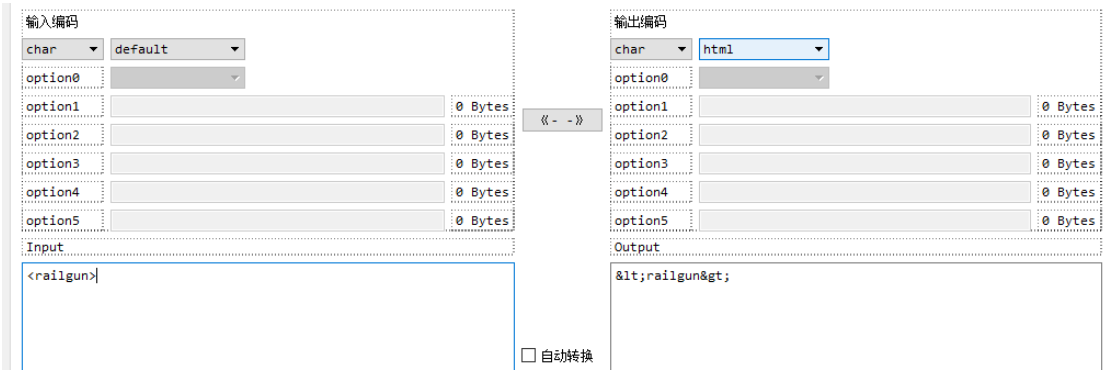


解码时可忽略“=”，也可正常解码

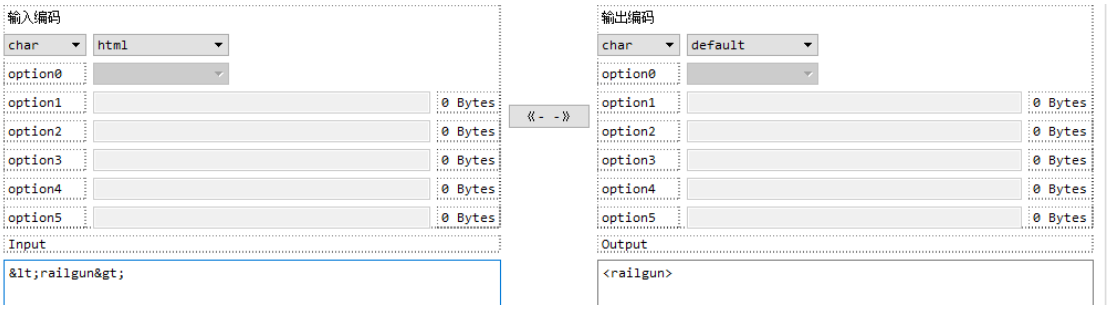


5.2.1.4 Html

编码

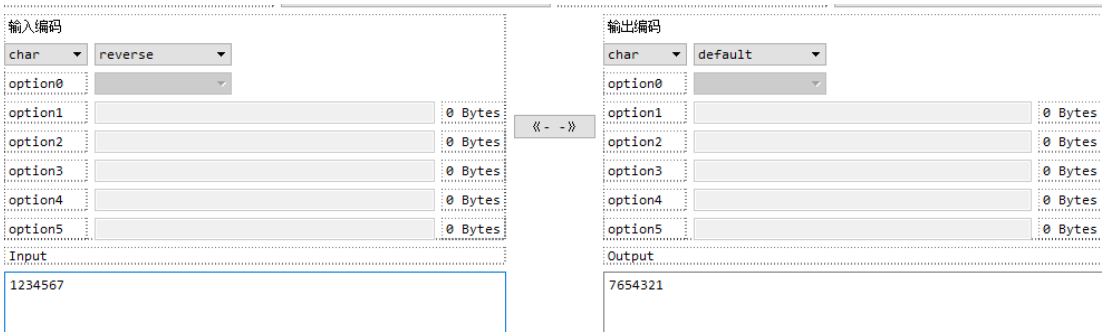


解码



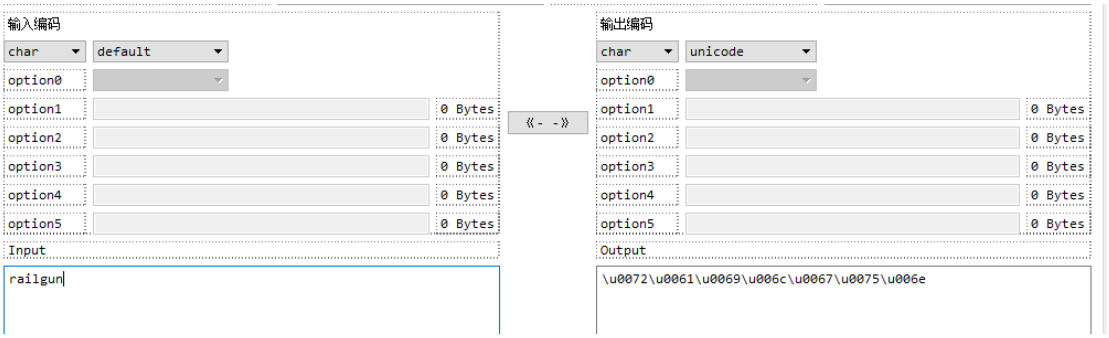
5.2.1.5 Reverse

Reverse 为逆序转换，设置成输入输出无差别。

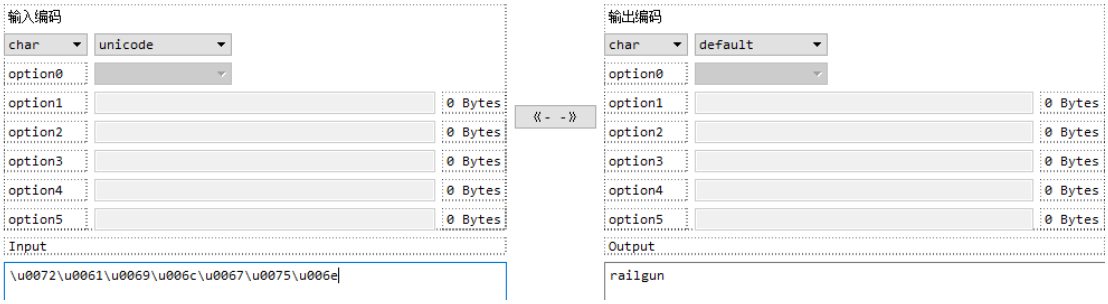


5.2.1.6 Unicode

Unicode 编码转换，如 struts2 payload 可能用到

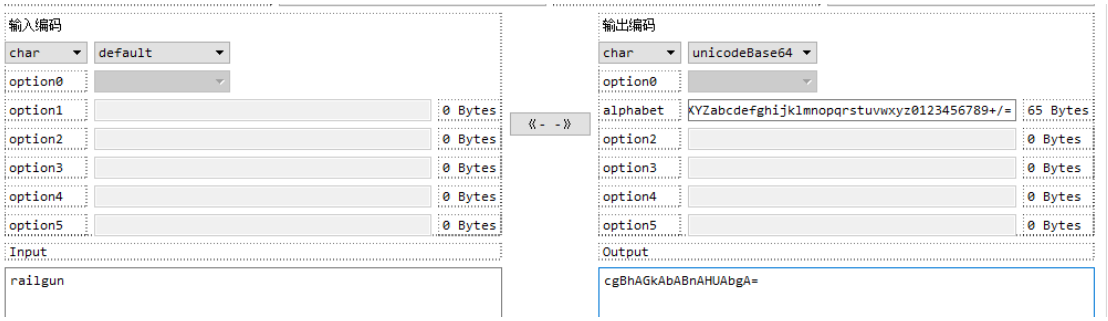


解码

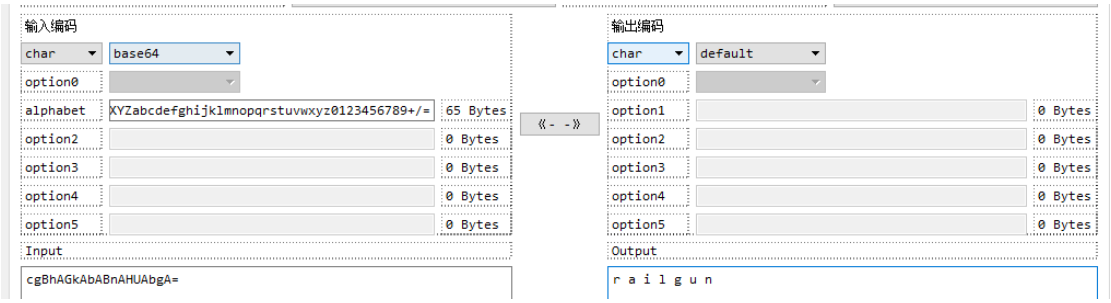


5.2.1.7 unicodeBase64

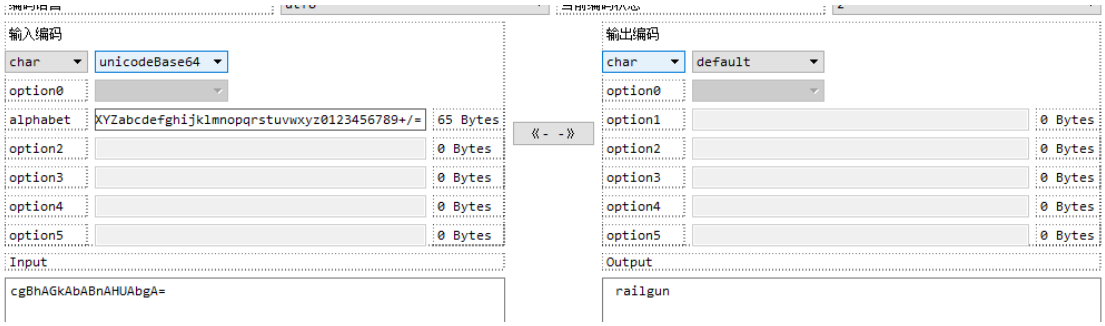
UnicodeBase64 表示先 unicode 编码再进行 base64 编码，这个 base64 编码主要用在 powershell 命令编码上，因为 powershell 字符集默认使用 unicode 编码。



下面编码内容使用常规 base64 解码，会出现不连续的情况，中间间隔的不是空格而是\x00

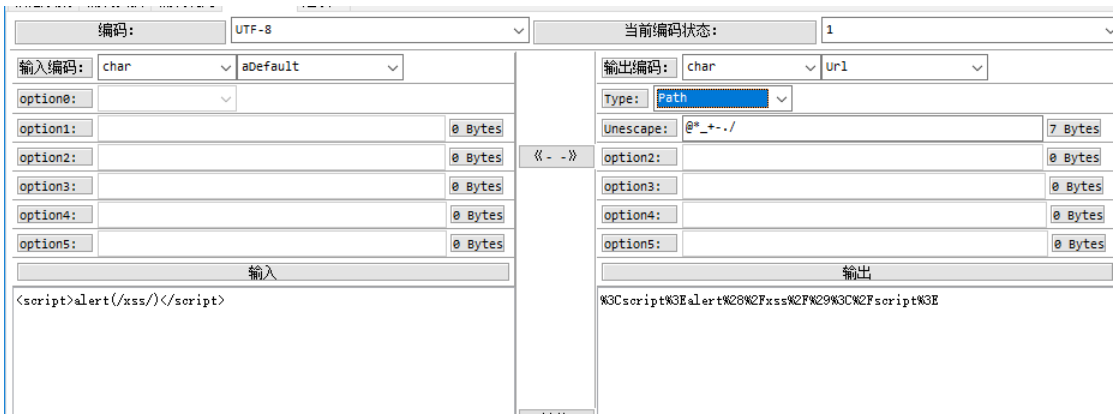


使用 unicodebase64 才能正常解码，在 powershell 病毒样本分析的时候可能用到。



5.2.1.8 url

上图为 URL 编码，Type 分为 Path，Query，Custom，分别用于 URI、POST 的数据、自定义需要编码字符；



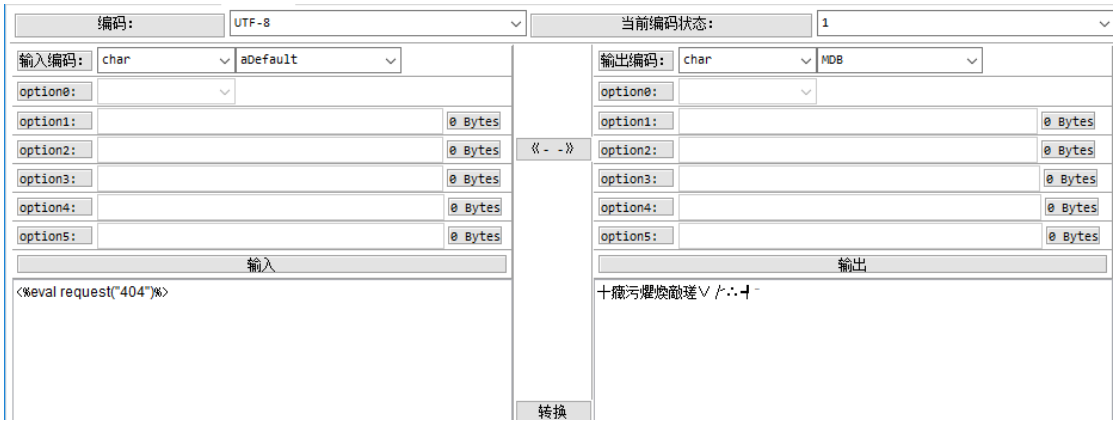
Unescape 字段只有在 Type 为 Custom 时才生效，用于设置某些字段不编码，默认值为“@*_+-./”，如“@*_+-./:”表示默认的加上冒号不编码。

解码

5.2.1.9 MDB

MDB 编码主要用于 access 一句话木马编解码。实际原理为 ANSI 编码后再使用 UNICODE 解码，ANSI 根据系统编码可能为 GBK 或 UTF-8。

编码，需要注意的是输入字节应该为偶数，因为 unicode 每个符号为 2 字节

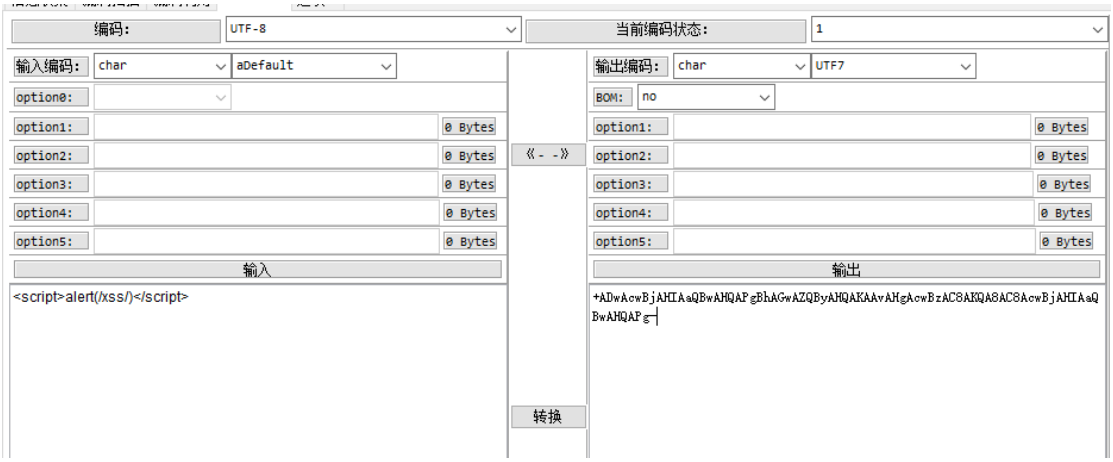


解码

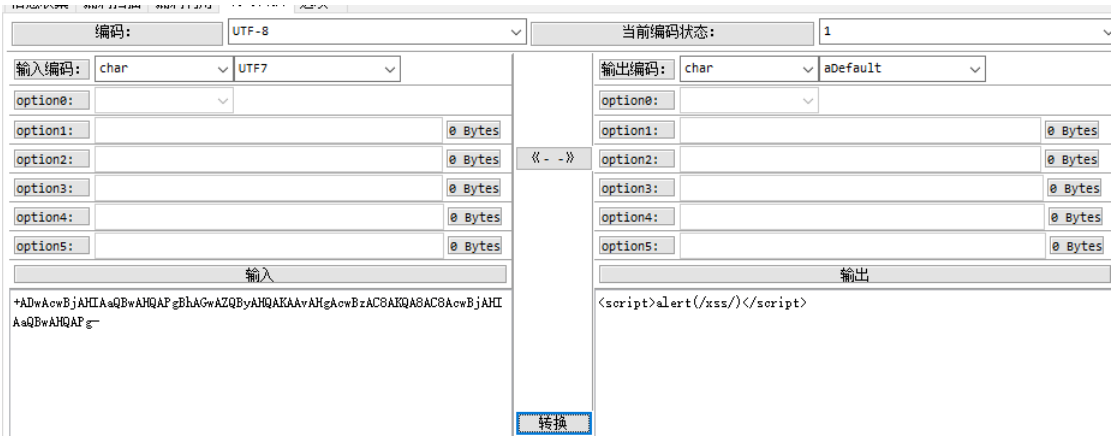
5.2.1.10 UTF7

UTF7 编码可用于如 XSS 绕过

编码，可选择是否设置 BOM 头。

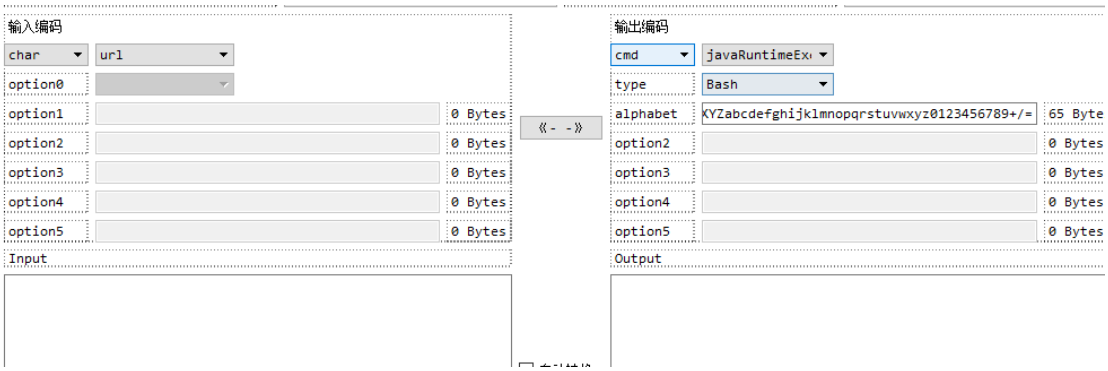


解码



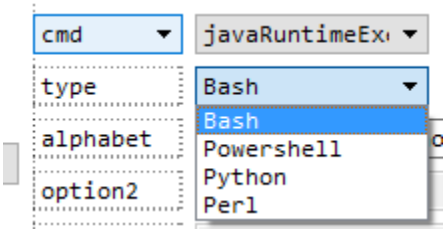
5.2.2 Cmd

5.2.2.1 javaRuntimeExec

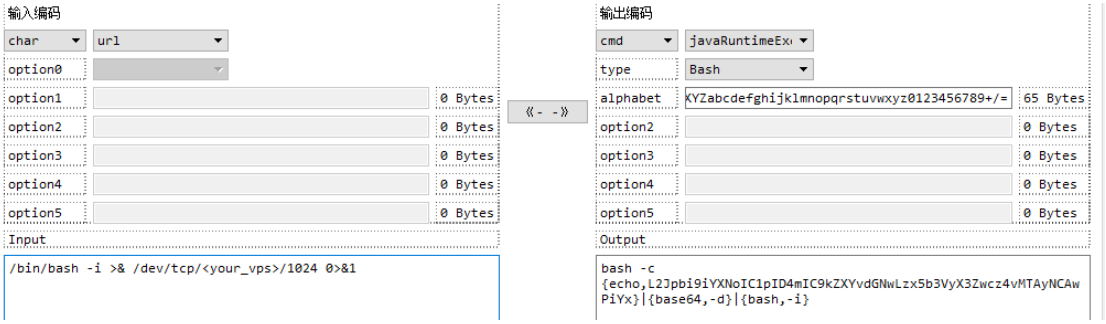


该类型只有编码，无解码，用于 `java.lang.Runtime.Exec` 方法执行命令时，不被特殊字符影响导致无法执行命令，所以进行编码。

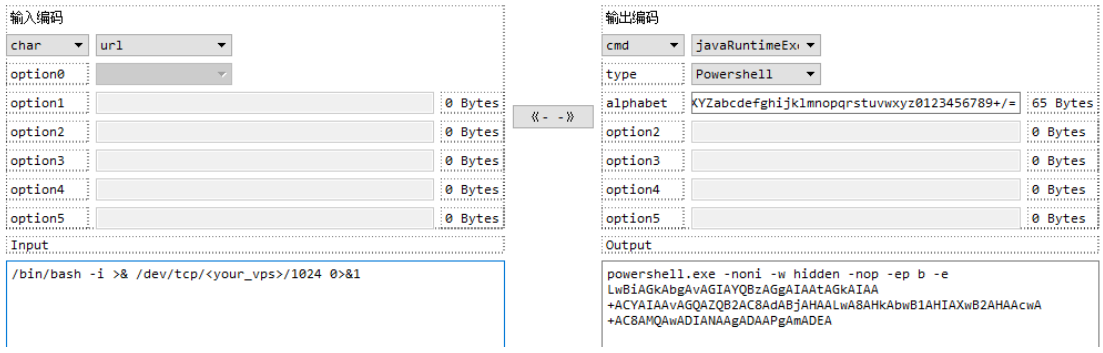
实际上将命令进行 base64 编码，支持以下四种脚本的编码。



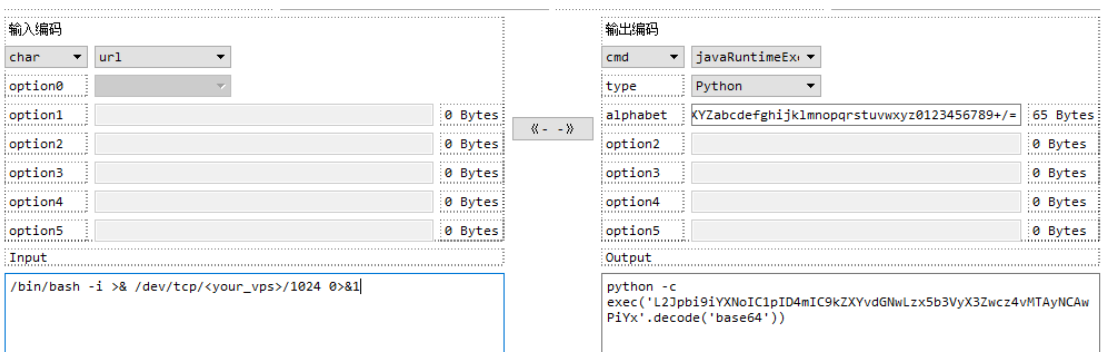
Bash：



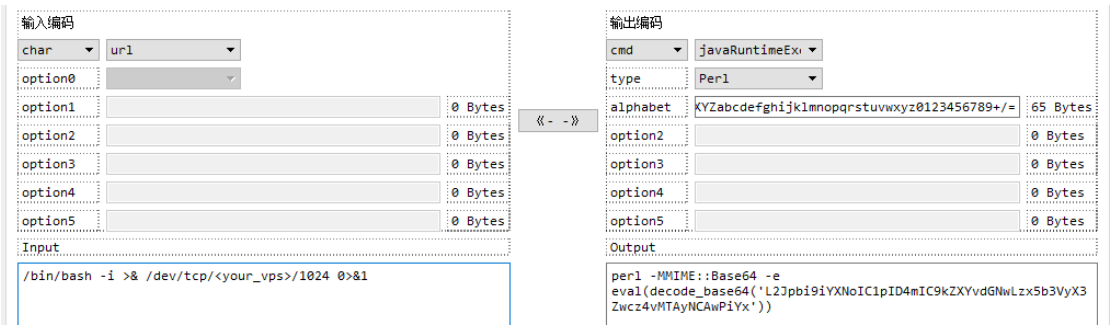
Powershell：



Python：



Perl:



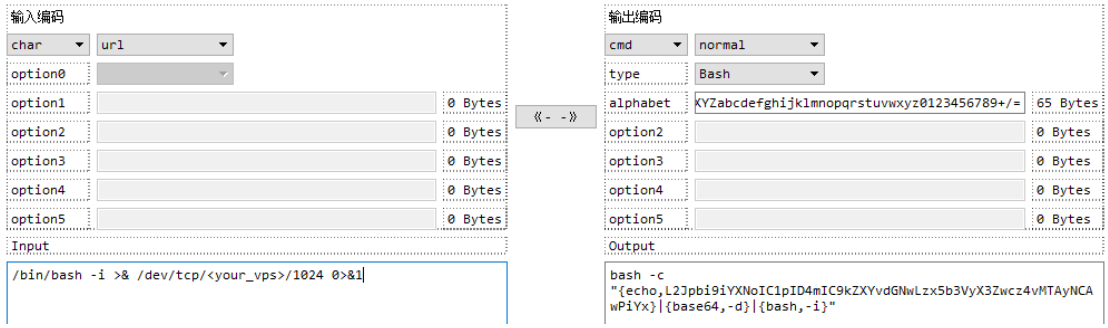
5.2.2.2 normal

normal 其实和 javaRuntimeExec 大致相同，除了 powershell 一样以外，bash/python/perl 在执行内容部分都使用双引号，仔细观察会发现 javaRuntimeExec 里都无双引号，因为 exec 方法会自动分割字符串，如果添加双引号可能会执行不成功。而在常规的命令执行里，如果无双引号也会执行不成功。

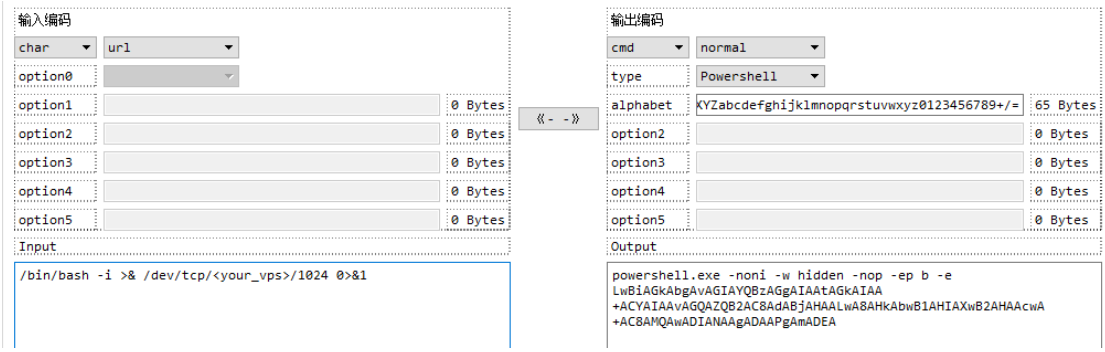
即 exec 方法里命令无需双引号，常规命令里需要双引号。

示例:

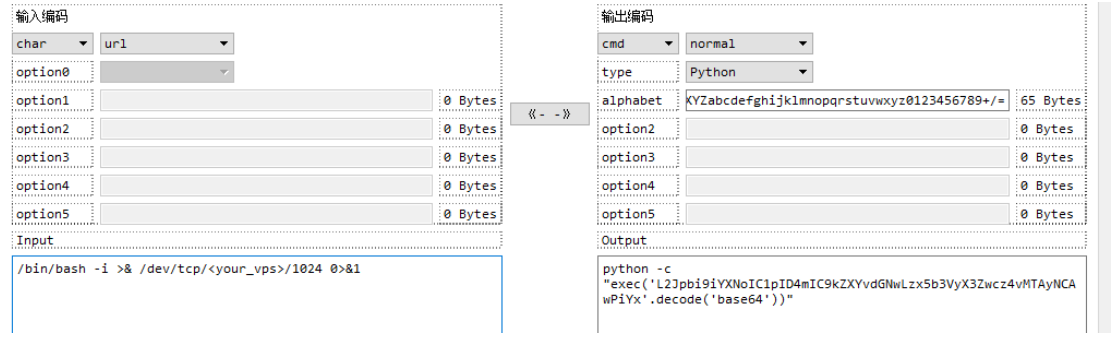
Bash



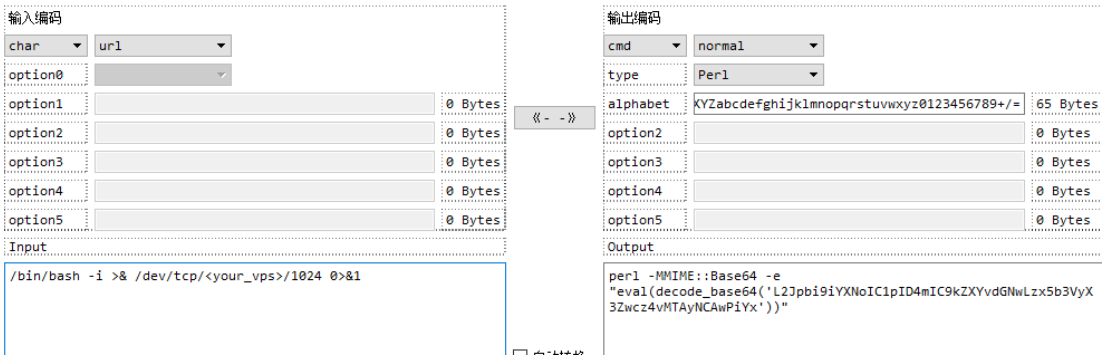
Powershell



Python



Perl



5.2.3 Crypto

5.2.3.1 AES

目前 AES 只设置了 CBC 模式，加密结果可选择 Base64 或 HEX，由于原始字符串不可读，所以需要再编码下。

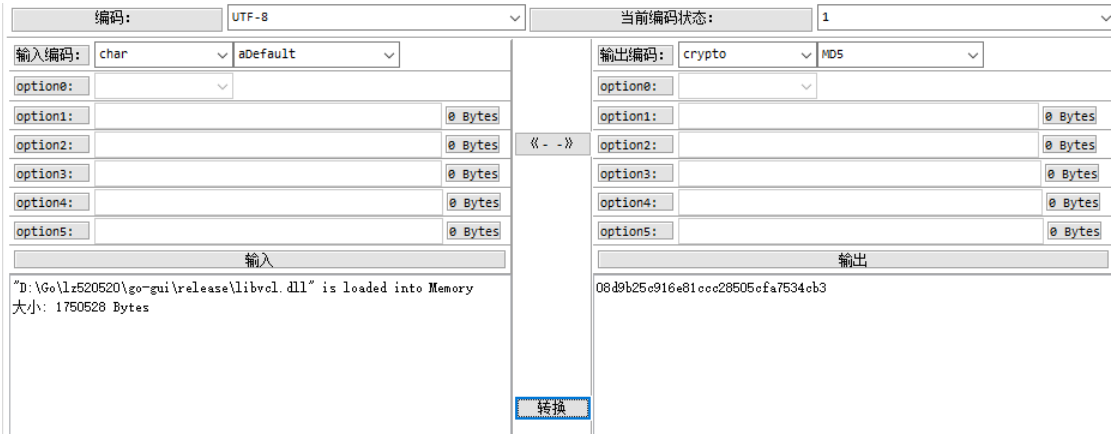
Key：加密密钥，长度必须为 16/24/32。

IV：初始化向量，长度为 16，如果不输入，会提取密钥前 16 位当做 IV。

下图会加密截图，解密配置类似。

5.2.3.2 MD5

只支持加密（不支持解密，没有做彩虹表）

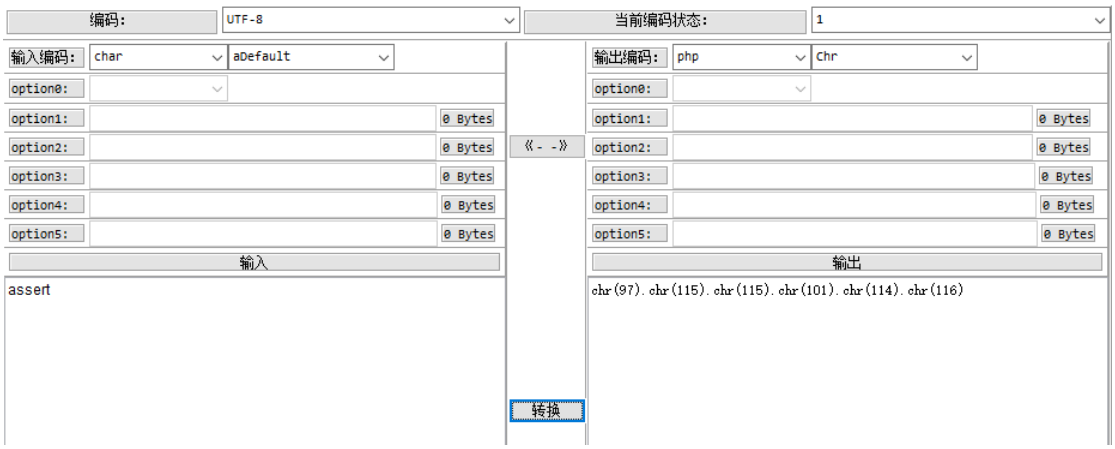


5.2.4 PHP

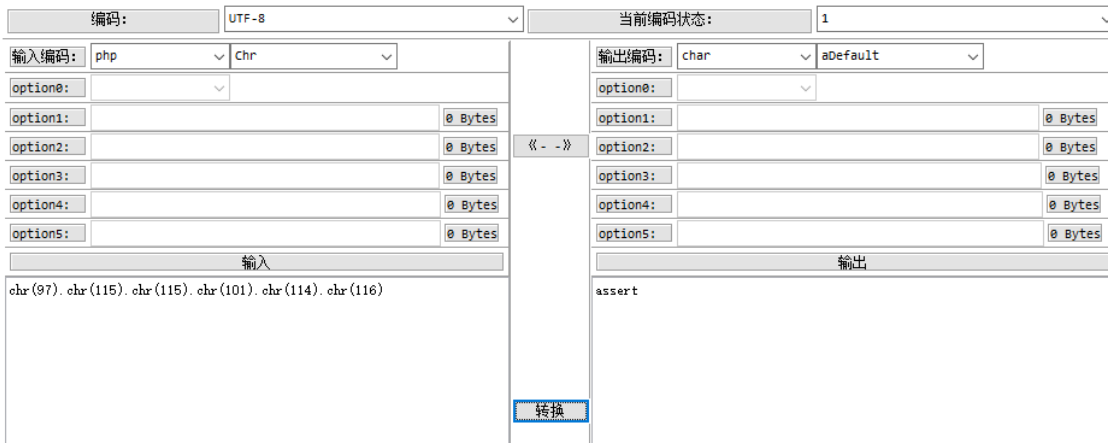
5.2.4.1 Chr

这个是之前做免杀测试顺手加的，用于 php 字符串变量编码混淆，将字符串转换成 ascii 值显示。

下图为编码



解码



6 BUG 反馈

留言 https://sec.lz520520.cn:4430/或 https://github.com/lz520520/railgun

熟人请私聊我