

恶意代码扫描平台用户手册

一、系统概述

(一) 平台简介

本平台是基于Vue+MySQL的恶意代码在线扫描平台，我们在后端集成了Yara引擎与支持Sigma规则的Zircolite引擎，您可以自行选择基于恶意软件的静态扫描或基于Sysmon捕获日志的动态扫描。此外，您可以上传自定义的Yara规则与Sigma规则，通过选择某些特定规则，来进行有针对性的扫描工作。

(二) 主要特性

1. 用户在部署之后可以自行选择在网页进行测试或在本地进行测试，具体测试方法详见[详细功能指南](#)部分
2. 本平台可以支持规则单独上传，例如.yar .yara .yml；也可支持压缩包形式的上传，例如.zip形式上传。
3. 本平台可以支持恶意样本的批量处理，您可以通过上传压缩包来一次性扫描多个恶意程序。
4. 本平台可以提供简易的扫描报告，如果您选择前端扫描测试，可以直观看到匹配的规则结果；如果您选择后端扫描测试，我们可以在您指定的位置输出简易的扫描报告（我们使用的json文件形式，您可能需要安装相应的插件或程序，例如python以便能够看到详细的结果输出）
5. 本平台在后端已经完成了规则的预编译，可以一定程度上缓解您上传规则数量较多的问题，但请不要上传过量的规则，以防止出现卡顿的现象。

(三) 系统支持

本网站仅能够支持Windows系统，针对MacOS系统与Linux系统，我们并没有进行配置。

我们推荐使用Windows10及以后的操作系统，且已经完成了MySQL的安装与配置，建议您使用Edge浏览器进行访问。

二、快速入门

(一) 本地配置

在本地部署之后，您可以通过下面的命令实现环境的搭建：

```
1 cd vuln_backend
2 python -m src.run
3 cd ..
4 cd dist
5 python -m http.server 8081
```

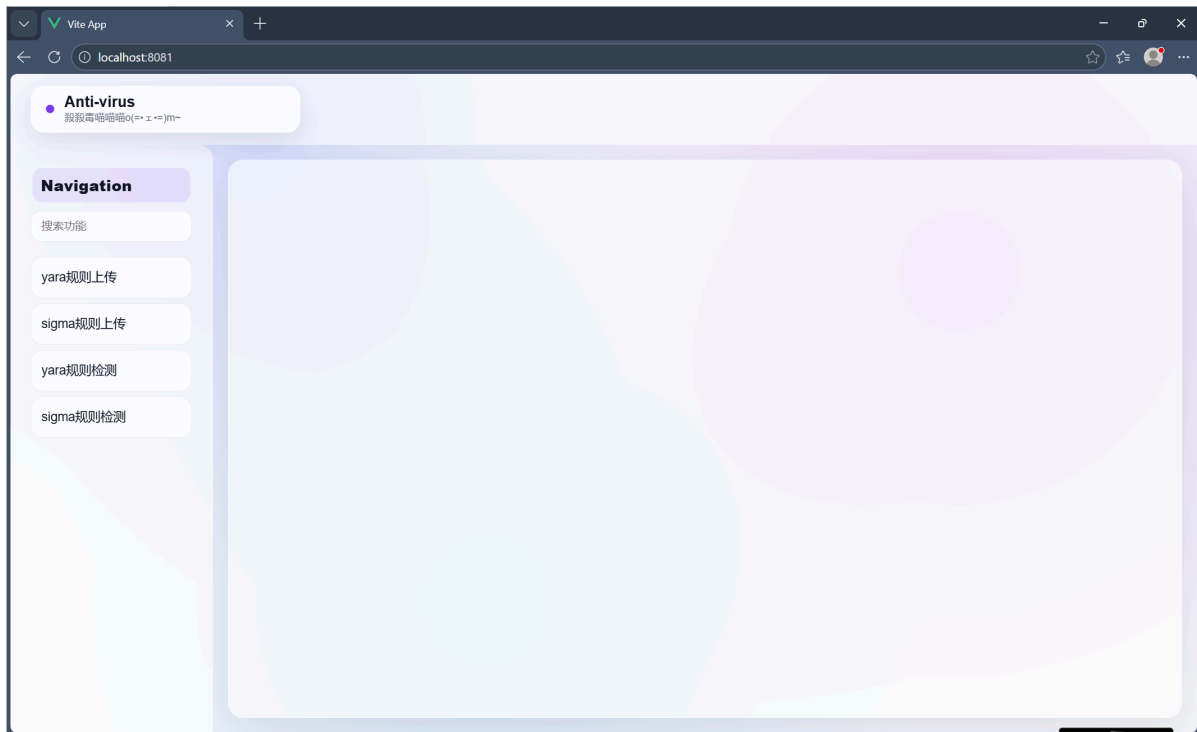
之后您就可以通过访问下面这个URL访问到我们的网页了：

```
1 http://127.0.0.1:8081/index.html
```

如果您不想使用这个上面的IP+端口+制定文件的形势，您可以使用下面这个方法也可以快速访问到指定的页面：

1 | localhost:8081

如果您能够看到下面这个页面：



即可证明您已经完成了我们环境的配置了，接下来您就可以进行规则上传、文件扫描、规则匹配、检测报告的获取了。

（二）功能测试

在完成部署之后，您可以简单上传某个规则以及某个恶意样本进行功能性测试，我们将给您提供示例：

样本上传

您可以在Yara规则上传或Sigma规则上传页面点击**选择文件**，然后就可以打开本地文件，选择您想要上传的测试文件，如果上传之后出现下面这个结果：

YARA 规则导入

支持上传 .yar / .yara / .zip (zip 内只允许包含 yar/yara 文件)。

manual-upload

已选择: Lab03_01.yar (303 B)

已选择: Lab03_01.yar (303 B)

清空

上传成功 (单文件)。

后端响应

```
{
  "created_at": "2025-12-26T16:44:20",
  "file_sha256": "86f1e0eec27b20629a66a58e49d42e052c8c4db1621d06ff6f38667204612ffc6",
  "filename": "Lab03_01.yar",
  "kind": "single",
  "ok": true,
  "rule_names": [],
  "skipped_count": 1,
  "source_name": "manual-upload",
  "stored_count": 0
}
```

说明您已经成功上传了指定文件了，您可以在数据库表中找到对应的规则，如下所示：

[illegible]

也可导入Sigma文件

Navigation

- 搜索功能
- yara规则上传
- sigma规则上传
- yara规则检测
- sigma规则检测

Sigma 规则导入 (YAML)

支持上传 .yml 或 .zip (zip 内只允许包含 .yml 文件)。前缀会先校验: 扩展名、zip 内容、.yml 合法性; 校验通过后再上传到后端。

规则源名称:

选择文件: 选择文件 已选择: sigma2.zip (426.26 KB)

已选择: sigma2.zip (426.26 KB)

校验并上传 清空

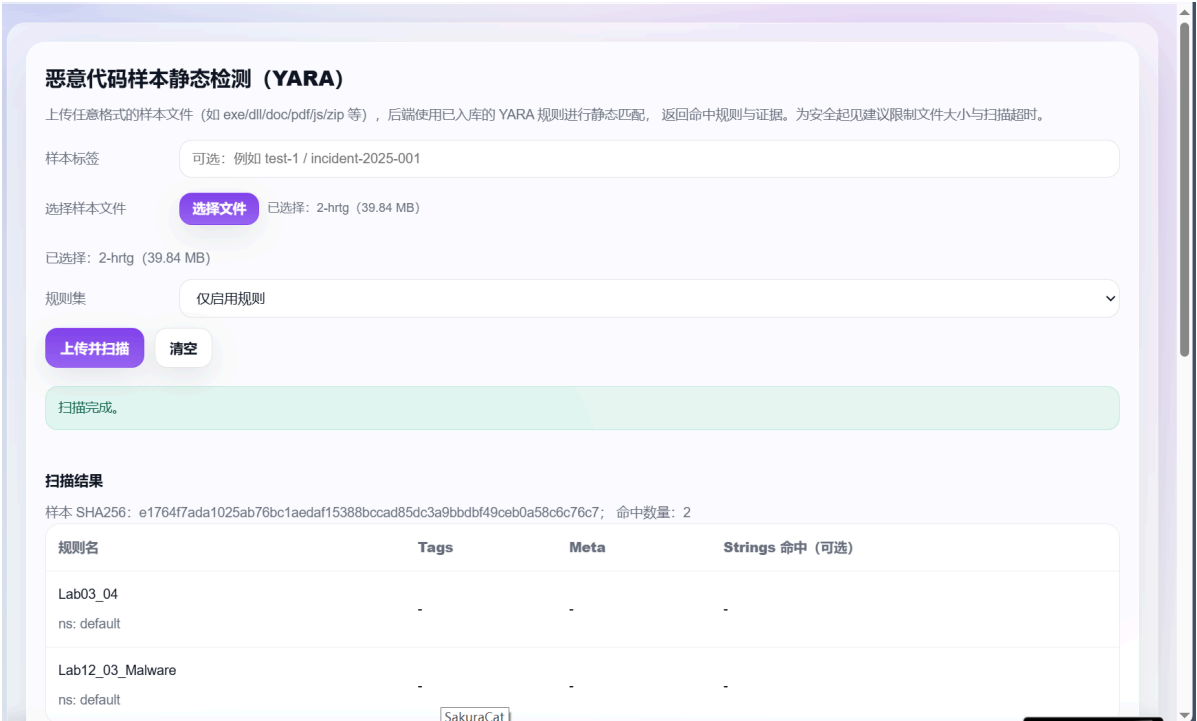
上传成功 (zip 规则包) .

Zip 内容预览 (校验通过后会展示)

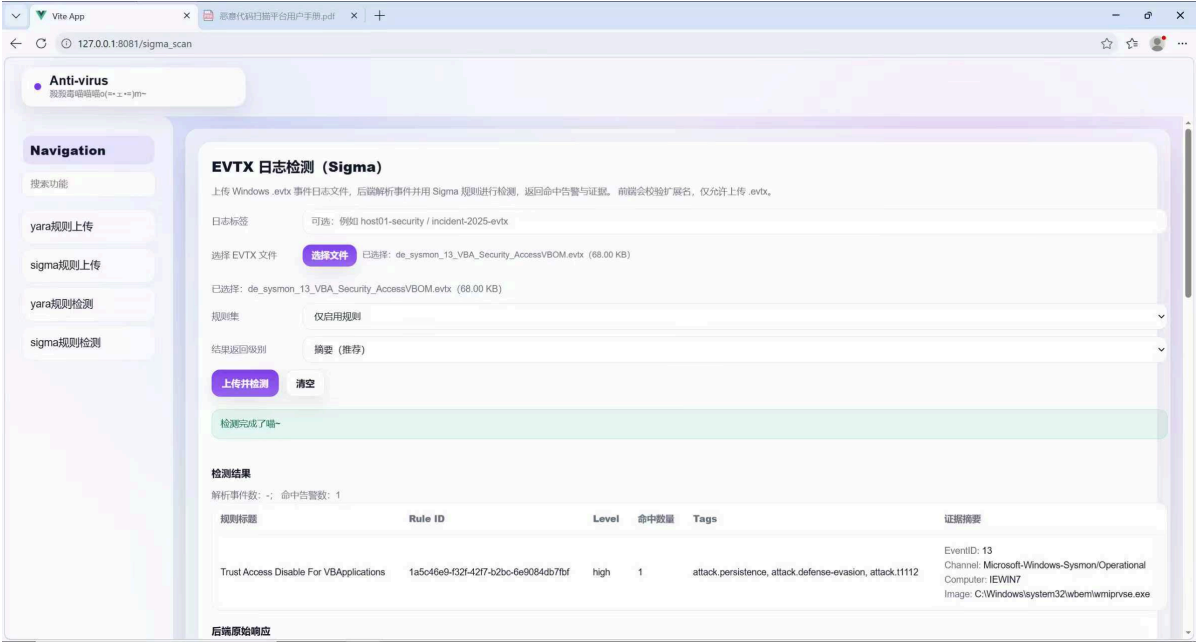
文件名	大小	yml 校验
posh_ps_invoke_obfuscation_clip.yml	767 B	OK
posh_ps_invoke_obfuscation_obfuscated_lex.yml	1.26 KB	OK
posh_ps_invoke_obfuscation_stdin.yml	749 B	OK
posh_ps_invoke_obfuscation_var.yml	787 B	OK
posh_ps_invoke_obfuscation_via_compress.yml	942 B	OK
posh_ps_invoke_obfuscation_via_rundll.yml	813 B	OK

恶意样本测试

您可以选择**Yara规则检测模块**，点击选择文件，然后选择一个测试的恶意样本，上传后点击测试，如果看到下面的结果，说明您的环境已经配置完毕了，基础的测试功能也已经实现了。



也可以选择**Sigma规则检测模块**



三、详细功能指南

如果您已经完成了上面所说的环境配置与基本功能测试，接下来我们就可以进行详细功能的展示了。

(一) 批量规则上传与检测

批量上传

与普通规则上传一致，您只需要将您需要上传的规则变为压缩包，就能完成一次性的成批上传了。需要注意的是，处于安全考虑，我们不允许其他样式后缀名规则的上传，因此在您的压缩包中，如果存在我们指定的格式（.yaml .yara .yar）之外的后缀名，会产生错误提示。为了能够正常使用，我们建议在批量上传时检查一下文件的后缀名：

已选择: Yara.zip (6.29 KB)

校验并上传

清空

上传成功 (zip 规则包)。

Zip 内容预览 (校验通过后会展示)

文件名	大小
Yara/Yara/Lab03_01.yar	303 B
Yara/Yara/Lab03_02.yar	291 B
Yara/Yara/Lab03_03.yar	229 B
Yara/Yara/Lab03_04.yar	335 B
Yara/Yara/Lab07_01.yar	296 B
Yara/Yara/Lab07_02.yar	330 B
Yara/Yara/Lab07_03.yar	310 B
Yara/Yara/Lab11_01.yar	1.30 KB
Yara/Yara/Lab11_02.yar	375 B
Yara/Yara/Lab11_03.yar	410 B

批量检测

与普通上传一致，您只需要将您需要上传的恶意程序变为压缩包，就能完成一次性的成批上传检测了。

恶意代码样本静态检测 (YARA)

上传任意格式的样本文件（如 exe/dll/doc/pdfs/zip 等），后端使用已入库的 YARA 规则进行静态匹配，返回命中规则与证据。为安全起见建议限制文件大小与扫描超时。

样本标签

可选: 例如 test-1 / incident-2025-001

选择样本文件

选择文件

已选择: Chapter_3L.zip (37.58 KB)

已选择: Chapter_3L.zip (37.58 KB)

规则集

仅启用规则

上传并扫描

清空

扫描完成。

扫描结果

样本 SHA256: ef609561158dda293de8b6375f6cba18a5e52e929f3b636f1dbf79fe0c61aa6a; 命中数量: 1

规则名	Tags	Meta	Strings 命中 (可选)
Lab03_03	-	-	-
ns: default	-	-	-

后端原始响应

(二) 后端扫描并输出日志

如果您不满足于仅能分辨出样本是否存在恶意行为，还想要得到相应的扫描日志，我们推荐您绕过前端直接使用后端进行测试，在这里，您不仅可以扫描单个文件，还可以扫描整个文件夹。您可以使用下面的指令来进行操作：

```
1 python yara_folder_scan_client.py "您的待扫描程序目录" --api
  "http://127.0.0.1:3000/scansamplewithYara" --rule-set enabled
```

这样您就会在您的扫描目录下面找到后缀为**您的待扫描程序目录_testres的文件夹**，这里就是对应的扫描结果，您可以查看相应的扫描报告。我们已经提前进行过后端的扫描测试了，下面是一个扫描的结果，可供您进行参考：

```
1 {
2     "ok": true,
3     "code": null,
4     "message": null,
5     "http_status": 200,
6     "sample_filename": "5-BossDaMajor",
7     "sample_sha256_from_server":
8     "730a41a7656f606a22e9f0d68782612d6e00ab8cfe1260160b9e0b00bc2e442a",
9     "rule_set": "enabled",
10    "hit_rule_count": 1,
11    "hit_rule_names": [
12        "Lab03_04"
13    ],
14    "matches": [
15        {
16            "meta": {},
17            "namespace": "default",
18            "rule": "Lab03_04",
19            "strings": [],
20            "tags": []
21        }
22    ],
23    "engine_stderr_tail": "",
24    "engine_stdout_tail": "",
25    "rel_path": "5-BossDaMajor",
26    "abs_path": "C:\\\\Users\\\\Malware_test_yzx\\Desktop\\恶意代码教学样本-在沙盒中修
27    改文件名\\恶意代码教学样本-在沙盒中修改文件名\\5-BossDaMajor",
28    "size": 2014208,
29    "local_sha256":
30    "730a41a7656f606a22e9f0d68782612d6e00ab8cfe1260160b9e0b00bc2e442a",
31    "timestamp": "2025-12-22 08:25:58"
32 }
```

这里是一个文件的扫描结果，您可以找到您需要的内容，包括**匹配到的规则、文件目录、文件大小、扫描时间**等信息。

```
1 {
2     "root_dir": "C:\\\\Users\\\\Malware_test_yzx\\Desktop\\恶意代码教学样本-在沙盒中修
3     改文件名\\恶意代码教学样本-在沙盒中修改文件名",
4     "api": "http://127.0.0.1:3000/scansamplewithYara",
```

```

4     "rule_set": "enabled",
5     "started_at": "2025-12-22 08:25:56",
6     "total_files_seen": 4,
7     "total_scanned": 4,
8     "total_ok": 4,
9     "total_hit_files": 3,
10    "total_skipped_too_large": 0,
11    "total_errors": 0,
12    "hit_rule_counter": {
13        "Lab03_04": 2,
14        "Lab12_03_Malware": 2
15    },
16    "finished_at": "2025-12-22 08:25:58"
17 }

```

如果您扫描了一整个文件夹，那么还可以找到一个文件夹的扫描结果。这里我们会向您展示**这个文件夹所在的目录，文件夹中文件的数量，扫描到的恶意文件，匹配到的对应规则，以及扫描完成的时间**。在这里，您可以**计算扫描准确率、漏报率**等信息。

(三) 扫描测试

我们首先扫描课上的恶意程序，我们在上传之前就已经完成了测试了，您可以自行检查，便不再赘述了。

在测试我们程序的准确率与漏报率时，我们使用了VirusShare的病毒样本，由于样本数量过多（65536个），我们的虚拟机内存不够了，因此我们就只解压了一部分（大概两万多个样本），然后我们进行跳过前端的测试，进行后端测试，来查看检测报告、准确率和漏报率。我们发现大概每测试一个样本都要消耗1秒左右的时间，因此在虚拟机里测试两万多个样本需要耗时太久，因此我们进行了三组测试，一组是测试490个病毒样本，一组是测试2000个病毒样本，还有一组测试了1880个病毒样本。

我们先测试了490个病毒文件，发现扫描结果如下：

```

1  {
2     "root_dir": "C:\\Users\\Malware_test_yzx\\Desktop\\temp",
3     "api": "http://127.0.0.1:3000/scansamplewithYara",
4     "rule_set": "enabled",
5     "started_at": "2025-12-26 19:28:04",
6     "total_files_seen": 490,
7     "total_scanned": 490,
8     "total_ok": 490,
9     "total_hit_files": 200,
10    "total_skipped_too_large": 0,
11    "total_errors": 0,
12    "hit_rule_counter": {
13        "Lab03_04": 163,
14        "Lab07_02_Malware": 12,
15        "lab12exe": 20,
16        "SUSP_NET_NAME_ConfuserEx": 1,
17        "MAL_Malware_Imphash_Mar23_1": 1,
18        "PowerShell_Susp_Parameter_Combo": 1,
19        "Lab12_03_Malware": 17,
20        "Lab03_02": 8,
21        "SUSP_Imphash_Mar23_2": 7,
22        "Lab12_04_Malware": 9,
23        "SUSP_PE_Discord_Attachment_Oct21_1": 3,

```

```

24     "RAT_Sakula": 1,
25     "ProcessInjector_Gen": 1,
26     "lab11dll": 2,
27     "SUSP_XORed_Mozilla_Oct19": 1,
28     "SUSP_XORed_URL_In_EXE": 1
29 },
30 "finished_at": "2025-12-26 19:37:41"
31 }

```

这里我们可以看到我们的正确率是40.8%，漏报率是59.2%。

然后我们又测试了两千个样例，结果如下：

```

1  {
2    "root_dir": "C:\\Users\\Malware_test_yzx\\Desktop\\temp",
3    "api": "http://127.0.0.1:3000/scansamplewithYara",
4    "rule_set": "enabled",
5    "started_at": "2025-12-26 19:57:15",
6    "total_files_seen": 2000,
7    "total_scanned": 2000,
8    "total_ok": 2000,
9    "total_hit_files": 792,
10   "total_skipped_too_large": 0,
11   "total_errors": 0,
12   "hit_rule_counter": {
13     "SUSP_Imphash_Mar23_2": 43,
14     "Lab03_04": 648,
15     "Lab12_03_Malware": 65,
16     "lab12exe": 58,
17     "lab11dll": 12,
18     "Lab07_02_Malware": 37,
19     "Lab03_02": 24,
20     "ProcessInjector_Gen": 2,
21     "Ping_Command_in_EXE": 2,
22     "Lab12_04_Malware": 19,
23     "SUSP_PS1_JAB_Pattern_Jun22_1": 2,
24     "MAL_Unknown_PWDumper_Apr18_3": 1,
25     "Lab12_01_Malware": 4,
26     "SUSP_certificate_payload": 1,
27     "SUSP_ELF_LNX_UPX_Compressed_File": 2,
28     "SUSP_XORed_MS DOS_Stub_Message": 2,
29     "Nanocore_RAT_Feb18_1": 1,
30     "Nanocore_RAT_Gen_2": 1,
31     "SUSP_PE_Discord_Attachment_Oct21_1": 8,
32     "IMPLANT_4_v3_AlternativeRule": 4,
33     "SUSP_ENV_Folder_Root_File_Jan23_1": 1,
34     "WEBSHELL_Cookie_Post_Obfuscation": 1,
35     "SUSP_NET_Msil_Suspicious_Use_StrReverse": 1,
36     "MAL_Malware_Imphash_Mar23_1": 3,
37     "Hunting_Rule_ShikataGaNai": 1,
38     "SUSP_PDB_Path_Keywords": 2,
39     "WaterBug_wipbot_2013_dll": 1,
40     "MAL_Neshta_Generic": 3,
41     "APT_Malware_PutterPanda_WUAUCLT": 1,
42     "Suspicious_PowerShell_WebDownload_1": 1,

```



```

43     "SocGholish_JS_22_02_2022": 1,
44     "MAL_ARM_LNX_Mirai_Mar13_2022": 1,
45     "SUSP_XORed_Mozilla_Oct19": 1
46 },
47 "finished_at": "2025-12-26 20:34:15"
48 }

```

这里我们可以看到我们的正确率是39.6%，漏报率是60.4%。

这主要是因为我们的Yara规则太少导致的（我们现在大概只有800个Yara规则左右）。

然后我们又搜索了两个github库中的Yara规则，主要是这两个库：

<https://github.com/100DaysofYARA/2024> 和 <https://github.com/Yara-Rules/rules>。通过添加完这些Yara规则，我们的数据库现在大概有1700个Yara规则作用，然后我们又进行了1880个恶意样本的测试（虚拟机内存不够，我们仅解压了部分恶意程序），结果如下：

```

1  {
2      "root_dir": "C:\\Users\\Malware_test_yzx\\Desktop\\temp",
3      "api": "http://127.0.0.1:3000/scansamplewithYara",
4      "rule_set": "enabled",
5      "started_at": "2026-01-09 11:21:15",
6      "total_files_seen": 1880,
7      "total_scanned": 1880,
8      "total_ok": 787,
9      "total_hit_files": 787,
10     "total_skipped_too_large": 0,
11     "total_errors": 7,
12     "hit_rule_counter": {
13         "ct_size_10kb_100kb": 787,
14         "file_pdf_header": 517,
15         "invalid_trailer_structure": 518,
16         "multiple_versions": 439,
17         "pdf_comment_present": 357,
18         "pdf_comment_values": 517,
19         "pdf_print_version": 517,
20         "ct_size_1kb_10kb": 320,
21         "spyeye_plugins": 165,
22         "Hunting_resources_noimps": 355,
23         "INFO_MPRESS_PACKER": 702,
24         "ct_size_100kb_1000kb": 451,
25         "file_pe_header": 702,
26         "head_mz": 702,
27         "head_mz_d_med_100kb_1mb": 280,
28         "head_pe_signed": 654,
29         "DontDoThatNoReally": 327,
30         "Lab03_04": 618,
31         "head_mz_c_med_10kb_100kb": 110,
32         "head_mz_a_small_lt_5kb": 3,
33         "QR_HTML_obfuscation": 10,
34         "invalid_xref_numbers": 35,
35         "INFO_PE_DeviceIOControl_API": 27,
36         "Lab07_02_Malware": 38,
37         "Lab13_01_EXE": 10,
38         "Lab13_02_EXE": 92,
39         "TTP_RegOpenKeyExA_HKEY_CURRENT_USER_tight": 8,

```

```
40 "file_pe_signed": 46,
41 "head_pe_unsigned": 46,
42 "lab12exe": 56,
43 "pe_unsigned_uncommon_product_name": 37,
44 "HKTL_NET_NAME_ConfuserEx": 1,
45 "SUSP_NET_NAME_ConfuserEx": 1,
46 "MAL_Malware_Imphash_Mar23_1": 6,
47 "GEN_PowerShell": 1,
48 "PowerShell_Susp_Parameter_Combo": 1,
49 "SUSP_Obfuscated_Powershell_xor": 10,
50 "ct_size_0_1kb": 9,
51 "powershell": 13,
52 "INFO_UPX_PACKER": 51,
53 "UPX": 45,
54 "suspicious_packer_section": 92,
55 "Rand_Constants": 198,
56 "ct_size_1mb_10mb": 286,
57 "head_mz_e_med_1mb_10mb": 286,
58 "TTP_contains_BTC_address": 125,
59 "Lab12_03_Malware": 61,
60 "maldoc_find_kernel32_base_method_1": 48,
61 "Lab03_02": 21,
62 "INFO_PE_Port_Slash_Combo": 13,
63 "INFO_PE_WebServer_References_Apache": 20,
64 "ct_size_10mb_100mb": 15,
65 "head_mz_f_large_gt_10mb": 15,
66 "maldoc_indirect_function_call_3": 1,
67 "INFO_VMPROTECT_PACKER": 16,
68 "HUNT_nopsled_8": 113,
69 "maldoc_getEIP_method_1": 31,
70 "Algorithm_AESSBoxes": 17,
71 "RAT_Sakula": 3,
72 "sakula_v1_2": 2,
73 "MSCryptoApi": 9,
74 "suspicious_creator": 18,
75 "deflate_copyright": 7,
76 "lab11dll": 7,
77 "HUNT_nopsled_16": 34,
78 "HUNT_nopsled_32": 10,
79 "SUSP_PE_RSRCs_Name_Strings_64_and_32_Refs": 1,
80 "SUSP_PE_Unusual_Imported_Library_Names": 4,
81 "SUSP_Imphash_Mar23_2": 27,
82 "Hunting_exploit_logs": 25,
83 "Lab12_04_Malware": 21,
84 "with_urls": 12,
85 "without_attachments": 14,
86 "without_images": 17,
87 "file_elf_header": 7,
88 "head_elf": 7,
89 "INFO_ANDROID_APK_FILE": 1,
90 "JavaDropper": 5,
91 "file_zip": 6,
92 "head_pkzip": 6,
93 "SUSP_MinimalImports_LoadLibrary_and_GetProcAddress": 31,
94 "INFO_PE_Contains_HTML_Page": 4,
95 "SharedStrings": 1,
```

```
96 "INFO_PE_WSAREcv_API": 6,
97 "SUSP_Obfuscated_Powershell_Casing_Anomaly": 7,
98 "ttp_lib_openssl_no_version_str_unsigned": 5,
99 "ttp_toolmark_fileextensions_array_2": 4,
100 "TTP_contains_onion_address": 5,
101 "INFO_EPL_BUILD": 5,
102 "INFO_Macho_LOObin_sysctl": 2,
103 "file_macho_header": 2,
104 "head_macho": 2,
105 "SUSP_PE_Discord_Attachment_Oct21_1": 5,
106 "head_html": 5,
107 "TTP_VirtualAlloc_RWX_loose_1": 11,
108 "Hunting_exploit_logs_2": 15,
109 "spyeye": 10,
110 "without_urls": 7,
111 "SUSP_References_Likely_Traffic_Listening_Network_Interface": 12,
112 "PoetRat_Python": 1,
113 "SUSP_References_Likely_Traffic_Capture_eth0": 4,
114 "with_images": 2,
115 "LimaAlfa": 1,
116 "salsa20": 1,
117 "BlackShades_4": 1,
118 "INFO_UPACK_PACKER": 3,
119 "sakula_v1_3": 1,
120 "ProcessInjector_Gen": 5,
121 "INFO_HTTP_HTTPS_XOR": 4,
122 "XOR_hunt": 5,
123 "blackhole_basic": 5,
124 "TTP_RegOpenKeyExA_HKEY_LOCAL_MACHINE_tight": 3,
125 "SUSP_LoadLibraryA_mutation_hex_enc_str": 1,
126 "header_evasion": 1,
127 "suspicious_launch_action": 1,
128 "INFO_ASPACK_PACKER": 10,
129 "function_through_object": 3,
130 "Contains_VBA_macro_code": 5,
131 "office_document_vba": 5,
132 "ct_size_10kb_1mb": 2,
133 "SUSP_Obfuscated_Mozilla_xor": 3,
134 "SUSP_XORed_Mozilla_Oct19": 3,
135 "with_sqlite": 1,
136 "head_mz_b_small_5kb_10kb": 6,
137 "Cobalt_functions": 1,
138 "SUSP_XORed_URL_In_EXE": 2,
139 "ttp_pe_size_of_code_gt_filesize": 10,
140 "CAP_HookExKeylogger": 1,
141 "ntdll_fallchill": 2,
142 "BinaryAndroidManifestXml": 1,
143 "SUSP_kernel32_mutation_reverse": 2,
144 "Suspicious_Powershell_WebDownload_1": 2,
145 "SUSP_References_Likely_Traffic_Capture_802_11": 2,
146 "INFO_KKRUNCHY_PACKER": 1,
147 "Lab12_01_Malware": 1,
148 "SUSP_LoadLibraryA_mutation_xor": 2,
149 "possible_exploit": 2,
150 "TTP_RegOpenKeyExA_HKEY_LOCAL_MACHINE_loose": 2,
151 "TTP_VirtualAlloc_RWX_tight_1": 1,
```

```

152     "MachO_File_pyinstaller": 3,
153     "PE_File_pyinstaller": 2,
154     "IMPLANT_4_v3_AlternativeRule": 2,
155     "SUSP_NullSoftInst_Combo_Oct20_1": 1,
156     "INFO_ELF_Contains_iptables": 1,
157     "SUSP_ELF_LNX_UPX_Compressed_File": 1,
158     "SUSP_DLL_All_LowerCase_Exports": 2,
159     "hex_script": 1,
160     "Hijacked_Chrome_Extensions_2024_Names_IDs": 2,
161     "Remcos": 1,
162     "INFO_Macho_Execution_Bin_sh": 1,
163     "MacControl": 1,
164     "MacControlCode": 1,
165     "MacControlStrings": 1,
166     "MyWScript_CompiledScript": 1,
167     "TTP_PowerShell_Download_command": 1,
168     "MAL_ADVOCBFUSCATOR_STRINGS_1": 1,
169     "INFO_LNK_Command_set": 2,
170     "ttp_toolmark_physicaldrive_unsigned": 1,
171     "ProgramLanguage_Golang": 1,
172     "SUSP_Imphash_Mar23_3": 1,
173     "Install": 1,
174     "InstallCode": 1,
175     "INFO_XOR_DOS_HEADER": 1,
176     "SUSP_XORed_MSDOS_Stub_Message": 1,
177     "TTP_RegOpenKeyExA_HKEY_CLASSES_ROOT_tight": 1,
178     "TTP_RegOpenKeyExA_HKEY_CURRENT_USER_loose": 1,
179     "GenerateTLSClientHelloPacket_Test": 1,
180     "Hunting_Rule_ShikataGaNaI": 1,
181     "HUNT_QBIT_STEALER_DEV": 1,
182     "warp": 1,
183     "warpStrings": 1,
184     "ntdll_reverse": 1
185 },
186 "finished_at": "2026-01-09 11:41:44"
187 }

```

这里我们可以看到我们的正确率是41.8%，漏报率是58.1%。

如果想要提升正确率，我们需要搜索更多的Yara规则，但由于时间原因，我们现在只能做到这个程度了。后续我们还可以继续进行实现，来补充Yara规则，升级版本。但这就是之后时间更加充分的时候的工作了，我们现在的yara工作就先到这里了。

对于Sigma规则，我们书写了课内实验规则，以及借鉴了网上规则（见提交），通过检测课内恶意代码实验日志以及网上<https://github.com/sbousseaden/EVTX-ATTACK-SAMPLES>的evtx日志测试，由于不好批量测，我们进行了人工大量随机测试，由于规则库的完善，基本不存在扫描不出来的情况

四、支持文件格式

- 对于**Yara规则**和**Sigma规则**的上传，我们支持使用**.yara .yar .yml**的未压缩形式，以及**.zip**的压缩包形式，您可以自行选择需要进行上传。
- 对于**日志文件**的上传，我们支持日志的**普通形式**的形式进行上传。
- 对于**恶意程序**的上传，我们支持使用**单个样本**的形式与**压缩包**的形式进行上传。

五、部署与使用

您可以查看我们项目中的README.md进行项目的部署配置，详细的过程我们再次便不再赘述了，如果还存在部署配置的问题，请联系我们。

六、常见问题解答

（一）数据库配置问题

您在本地克隆了我们的项目之后，需要对数据库配置进行修改，下面我们会详细给您介绍需要修改哪些部分：

vuln_backend/src/config.py

```
1 class Config:
2     SECRET_KEY = os.environ.get('SECRET_KEY') or 'your-secret-key'
3     #连接串 下面这一行要根据您的mysql密码来填写: SQLALCHEMY_DATABASE_URI =
4     'mysql+pymysql://root:您的数据库密码@localhost:3306/nvd_database'
5     SQLALCHEMY_DATABASE_URI =
6     SQLALCHEMY_TRACK_MODIFICATIONS = False
7     PORT = 3000
```

这个文件是我们的数据库配置文件，在这里需要配置连接串，您需要根据您的数据库配置进行自行的修改，我们在注释中已经给出了一个示例说明。

vuln_backend/src/apps/services/CnvdDataInfoImpl.py

```
1 def delVulnCnvd(vuln_id):
2     db = pymysql.connect(host="localhost", user="root", password="您的数据库密
3     码", database="nvd_database")
4     cursor = db.cursor()
5
6 def updateVulnCnvd(id, data):
7     db = pymysql.connect(host="localhost", user="root", password="你的数据库密
8     码", database="nvd_database")
9     cursor=db.cursor()
```

这个文件中您也需要将您的数据库密码进行输入，我们需要连接您部署在本地的数据库，这样才能够复现我们的功能。

在完成这两个文件中的配置之后，数据库问题一般就得到解决了，如果存在问题，请联系我们。

（二）匹配精度过高的问题

我们的数据库中有个收集的规则是 ct_size_gt0，这个规则直接匹配文件大小大于0的样本，这就导致了許多误报，针对扫描结果，我们建议您忽略这个样本的扫描结果，直接看第二多的匹配规则，那里才是真正扫描到的样本数。我们对规则搜集检查不严格的行为对您表示歉意。

七、联系我们

下面是我们的邮箱，您可以根据邮箱来联系我们：

2312796@mail.nankai.edu.cn

2313781@mail.nankai.edu.cn

2313508@mail.nankai.edu.cn

2312323@mail.nankai.edu.cn