Threat Hunting C2 Over DNS

> whoami

| Researcher @ Active CM

| Instructor @ AntiSyphon

| Building @ aionsec.ai

> follow

home | faanross.com

linkedin | faan rossouw

x | @faanross

github | @faanross

youtube | @faanross

Threat Hunting C2 Over DNS

Threat Hunting C2 Over DNS

“beyond the obvious”

what is it + why its awesome

Threat Hunting C2 Over DNS

“beyond the obvious”

what + how + why misused

Threat Hunting C2 Over DNS

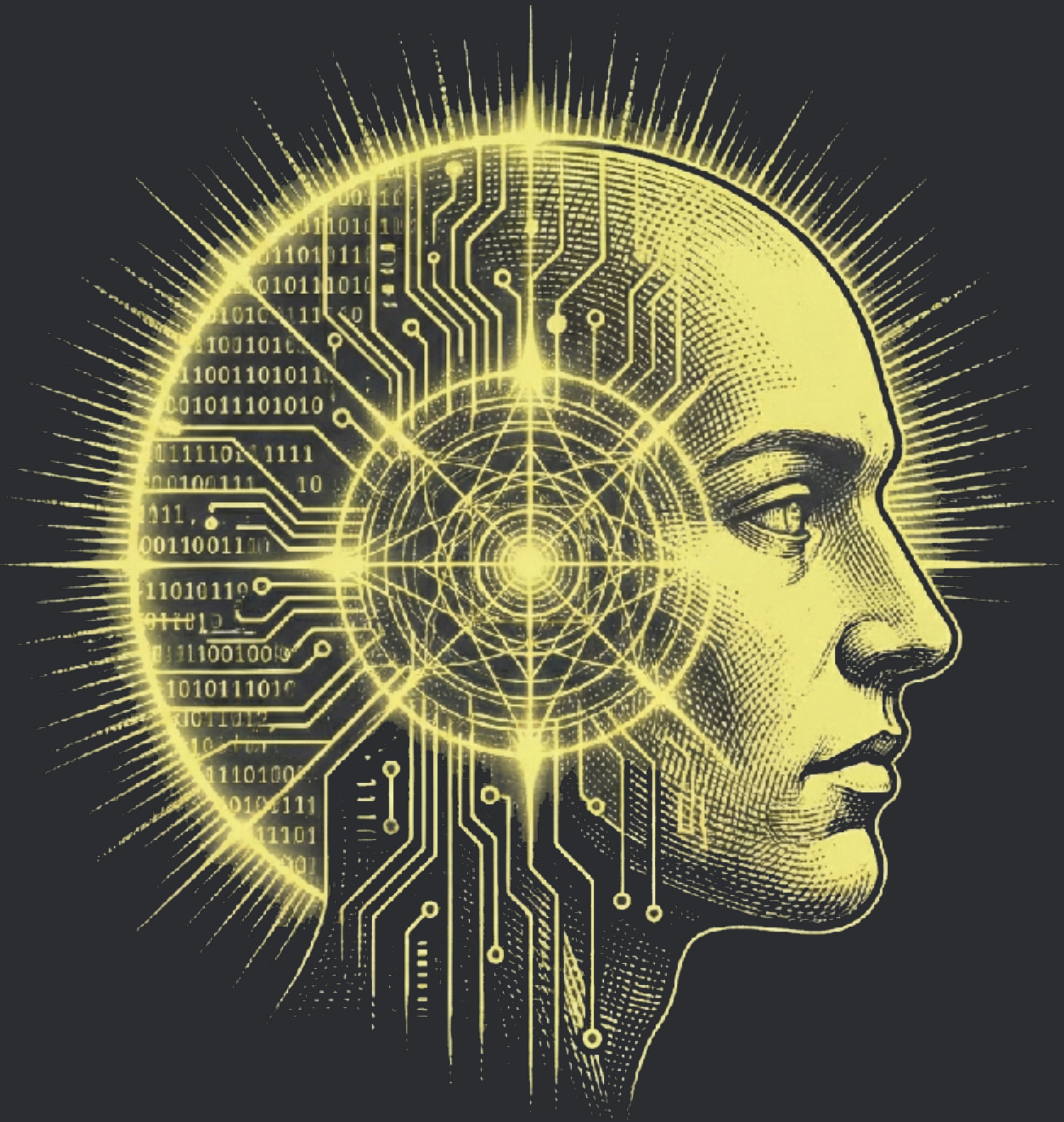
“beyond the obvious”

Threat Hunting C2 Over DNS

“beyond the obvious”

if know what to look for trivial

to find… except when its not



Threat Hunting

what it is +
why it’s awesome

defensive security posture

two things come to mind

PROTECTION
PROTECTION
AV
FIREWALLS
AV
AUTHENTICATION

stop them from coming in

deal with them once

discovered,or revealed

themselves (extortion)

PROTECTION

PROTECTION

FIREWALLS

AV

AUTHENTICATION

RESPONSE

RESPONSE

INCIDENT HANDLING

FORENSICS

PROTECTION

PROTECTION

FIREWALLS

AV

AUTHENTICATION

DWELL TIME: 6+ MONTHS AVERAGE

WAITING, WATCHING, LEARNING PATTERNS

RESPONSE

RESPONSE

INCIDENT HANDLING

FORENSICS

PROTECTION
PROTECTION
FIREWALLS
AV
AV
AUTHENTICATION
THREAT HUNTING MISSION
COLLAPSE THAT GAP
RESPONSE
RESPONSE
INCIDENT HANDLING
FORENSICS

PROTECTION

PROTECTION

FIREWALLS

AV

AUTHENTICATION

THREAT HUNTING MISSION

COLLAPSE THAT GAP

RESPONSE

RESPONSE

INCIDENT HANDLING

FORENSICS

“assumed compromise”

“assumed compromise”

- Pragmatism >>> Idealism
- No way we can keep 100% of attackers out
- TH: If someone is inside, how would we find them?

The goal of TH is…

The goal of TH is…

Finding threats!

The goal of TH is…

Finding threats!

Right…?

Not so fast…

Let’s turn to guidance

from one of our elders

David J. Bianco

- ”Pyramid of Pain guy”
- High Druid of TH
- FWs: sqrrl, PEAK

Ask most people:

What is goal of Threat Hunting?

Ask most people:

What is goal of Threat Hunting?

Finding threats. (duh)

Ask most people:

What is goal of Threat Hunting?

Finding threats that evaded existing detection.

That was his original definition (sqrrl)

But it has since evolved (PEAK)

What is goal of Threat Hunting?

What is goal of Threat Hunting?

"Improving overall security posture
through proactive searching.”

> “It's about making the organization fundamentally more secure through the hunting process itself.”

How does it do this?

Goal: Improve Overall Security Posture

Goal: Improve Overall Security Posture

PEAK defines 5 Core Metrics

Goal: Improve Overall Security Posture

Goal: Improve Overall Security Posture

1. Incidents Discovered

Actual threats found

Goal: Improve Overall Security Posture

2. New Detections Created

Analytics/rules produced from hunts

Goal: Improve Overall Security Posture

3. Visibility Gaps Identified

Missing telemetry or blind spots discovered

Goal: Improve Overall Security Posture

4. Vulnerabilities/Misconfigurations Found

Security weaknesses identified

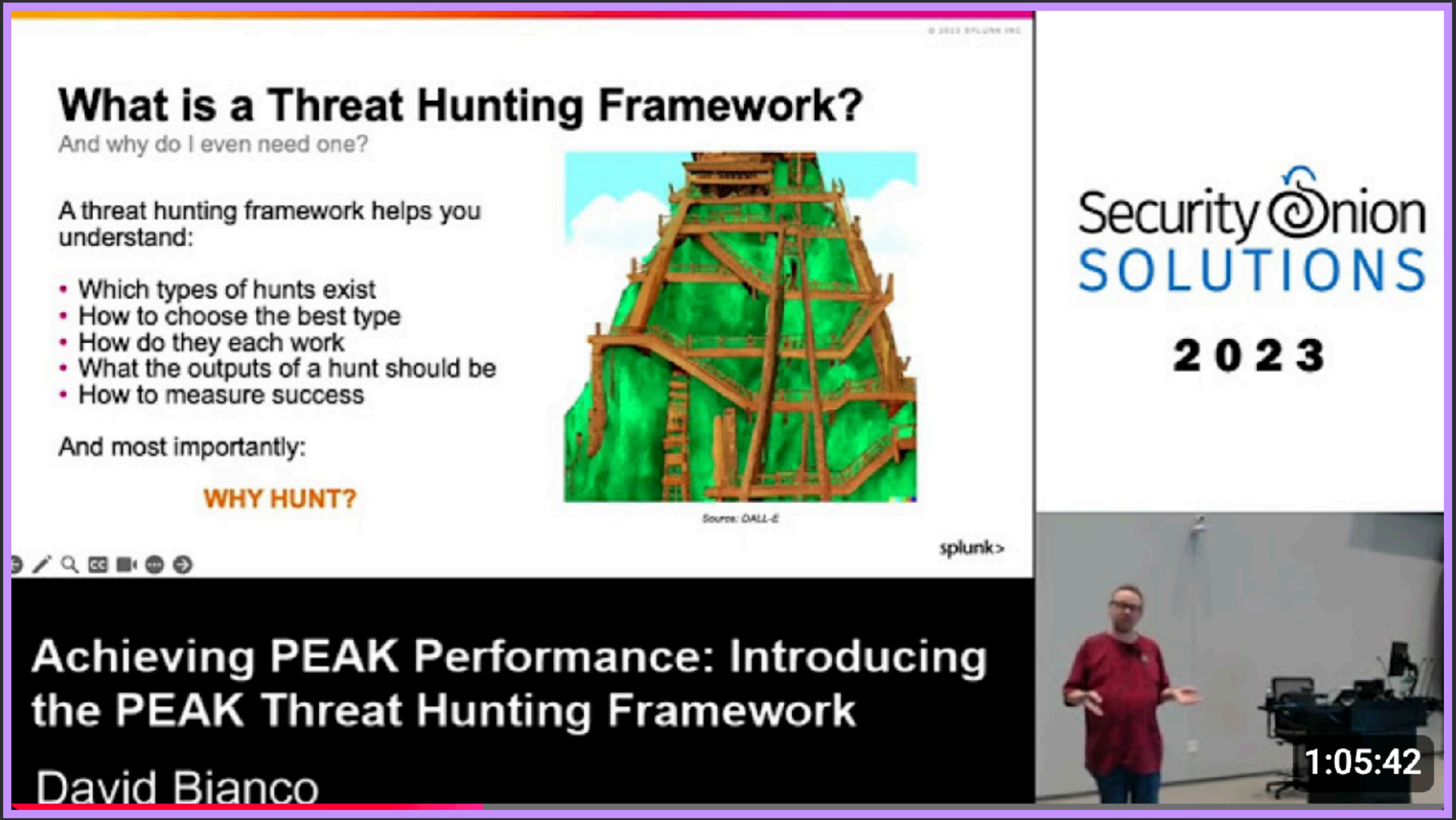
Goal: Improve Overall Security Posture

5. Techniques Hunted

Coverage across ATT&CK or similar framework

Hunt outputs feed back into the system to strengthen it (detections, documentation, future hypotheses)

A hunt that finds no incidents but produces solid documentation and new detections is still a successful hunt



What is a Threat Hunting Framework?

And why do I even need one?

A threat hunting framework helps you understand:

- Which types of hunts exist
- How to choose the best type
- How do they each work
- What the outputs of a hunt should be
- How to measure success

And most importantly:

WHY HUNT?

Source: DALL-E

Achieving PEAK Performance: Introducing the PEAK Threat Hunting Framework

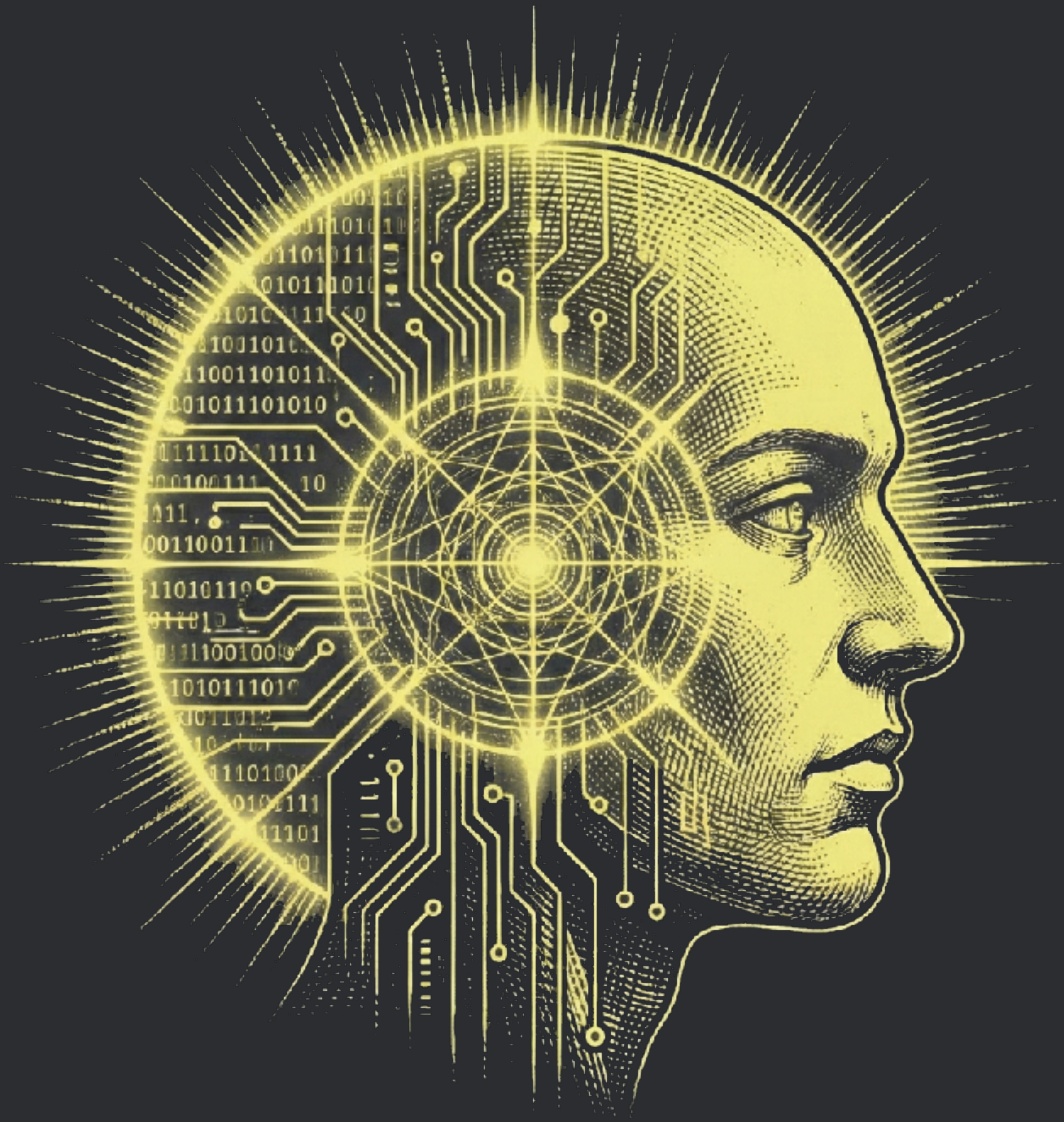
David Bianco

Threat Hunting C2 Over DNS

“beyond the obvious”

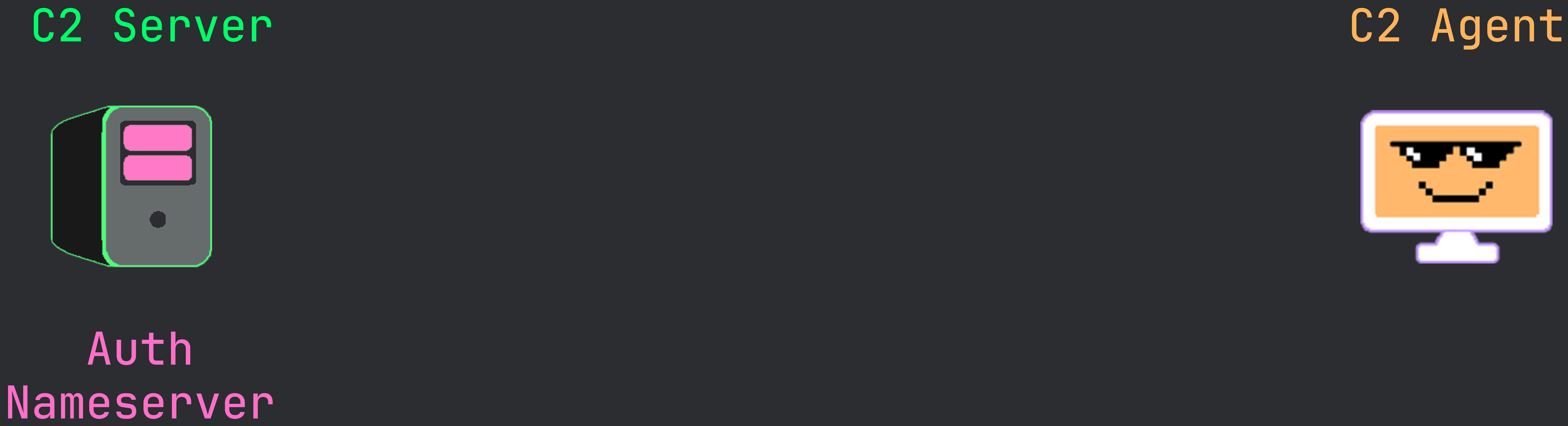
Threat Hunting C2 Over DNS

“beyond the obvious”



C2 over DNS

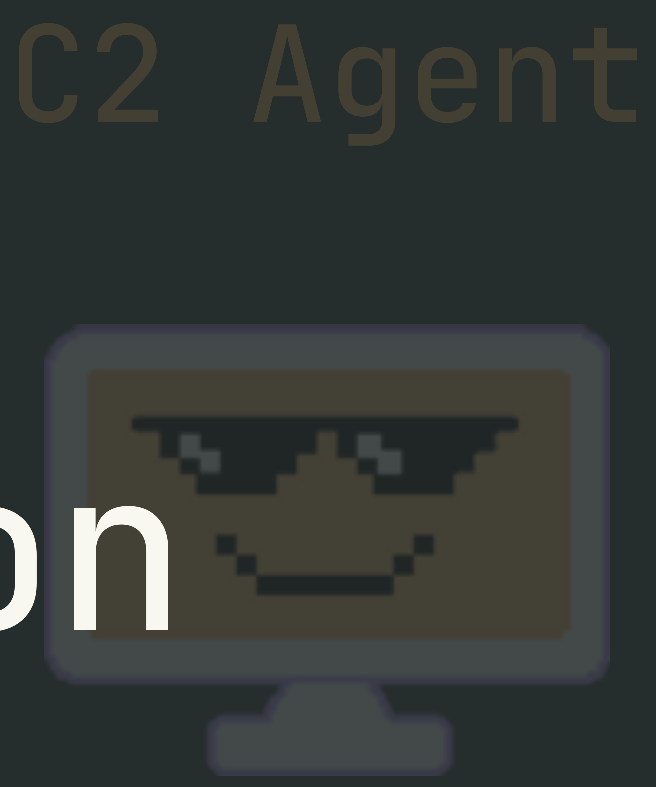
The Domain Name System is fundamentally a distributed, hierarchical database that translates human-readable domain names into machine-usable IP addresses



C2 Server

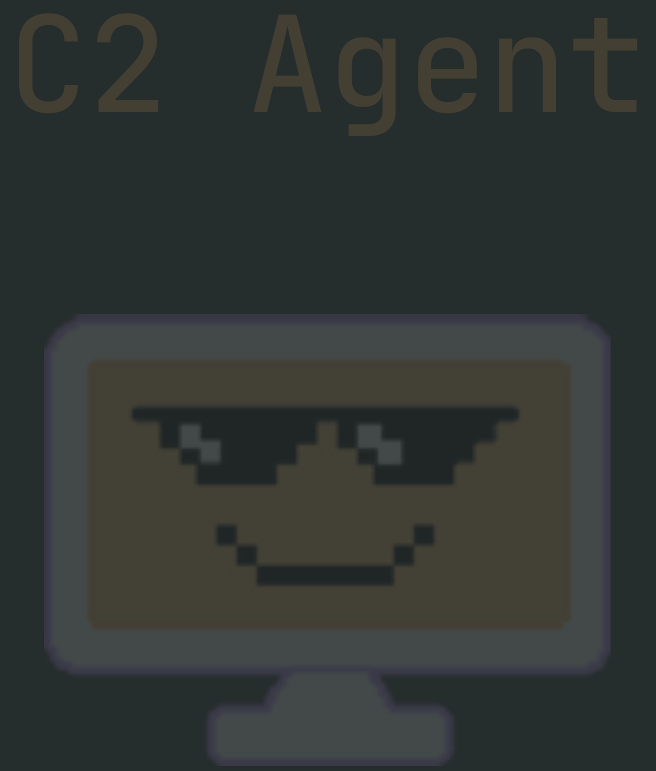
Auth Nameserver

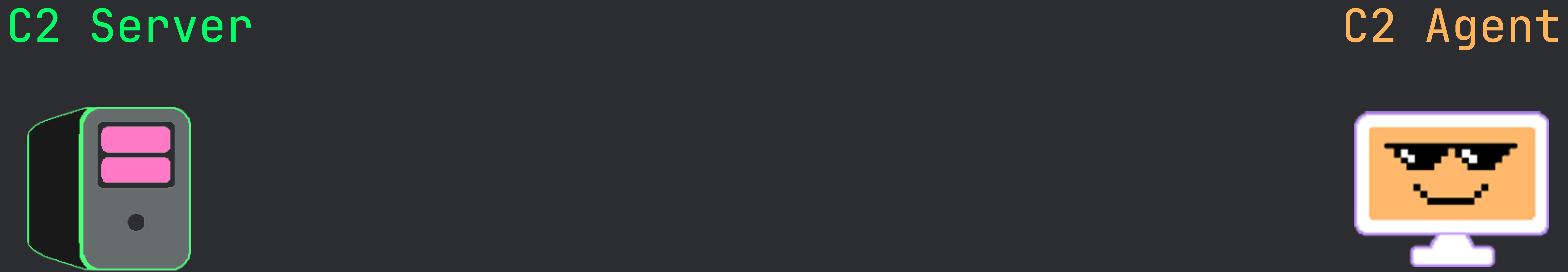
C2 Agent



important, here we imply there is a direct connection between C2 agent and server…

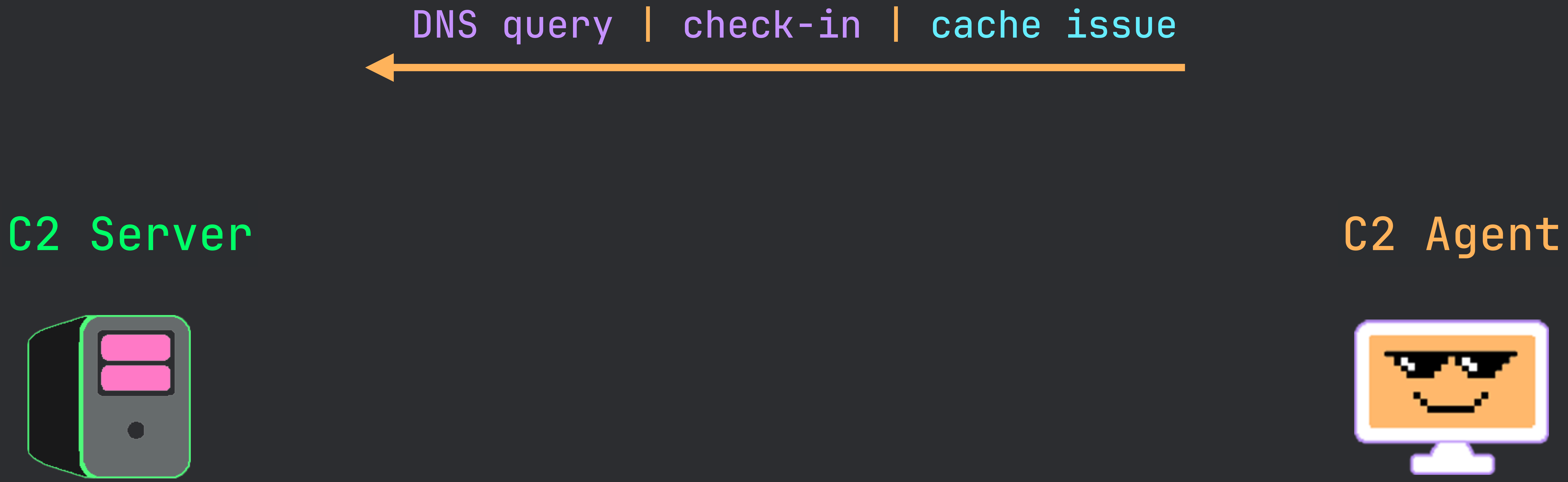
most often, the C2 agent is communicating directly with the local DNS resolver





C2 Server

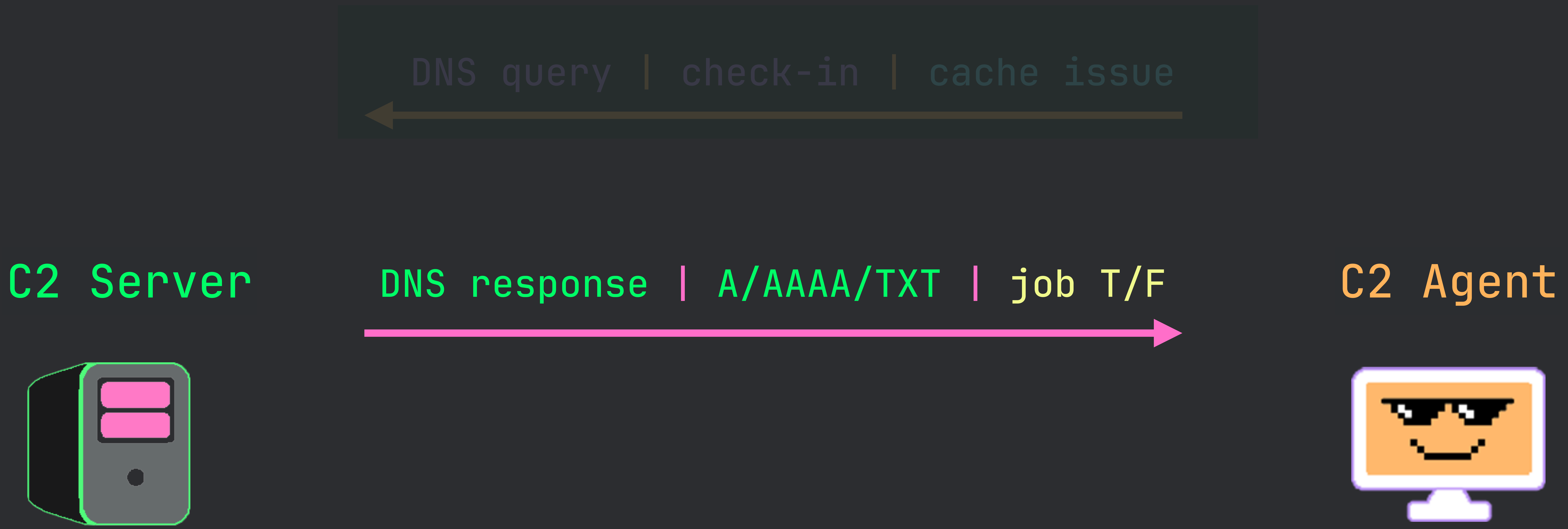
C2 Agent



DNS query | check-in | cache issue

C2 Server

C2 Agent

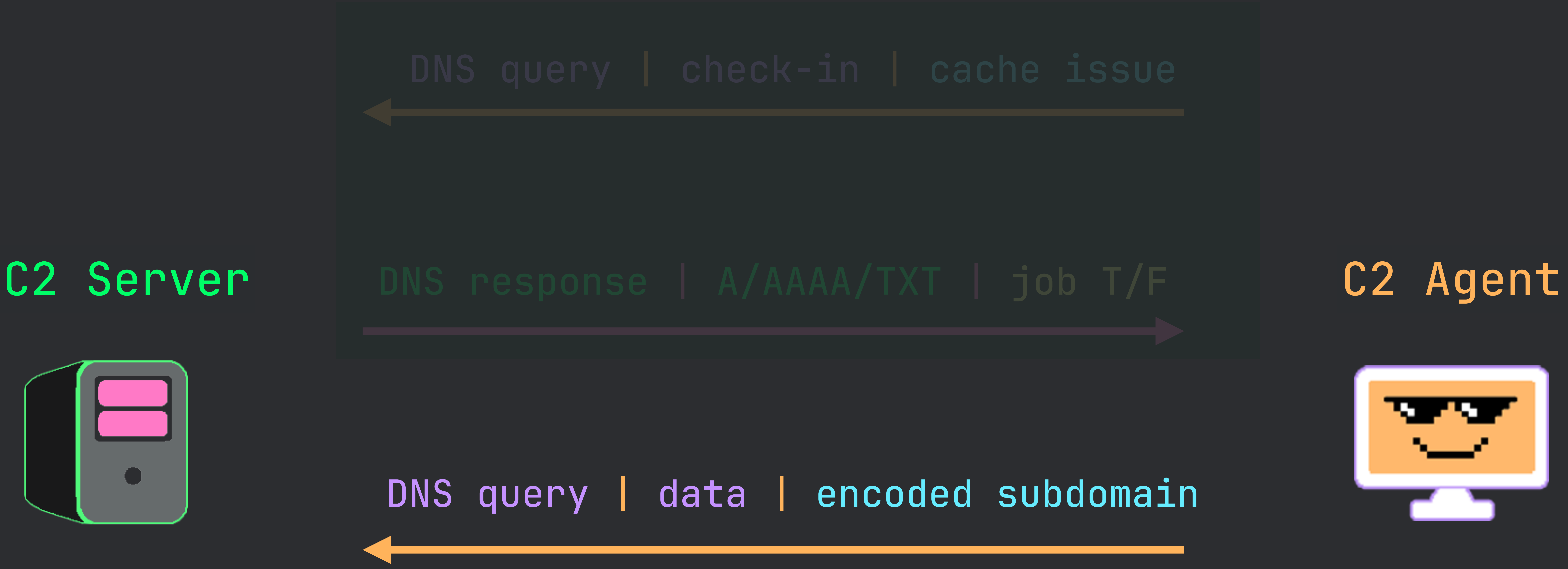


DNS query | check-in | cache issue

C2 Server

DNS response | A/AAAA/TXT | job T/F

C2 Agent



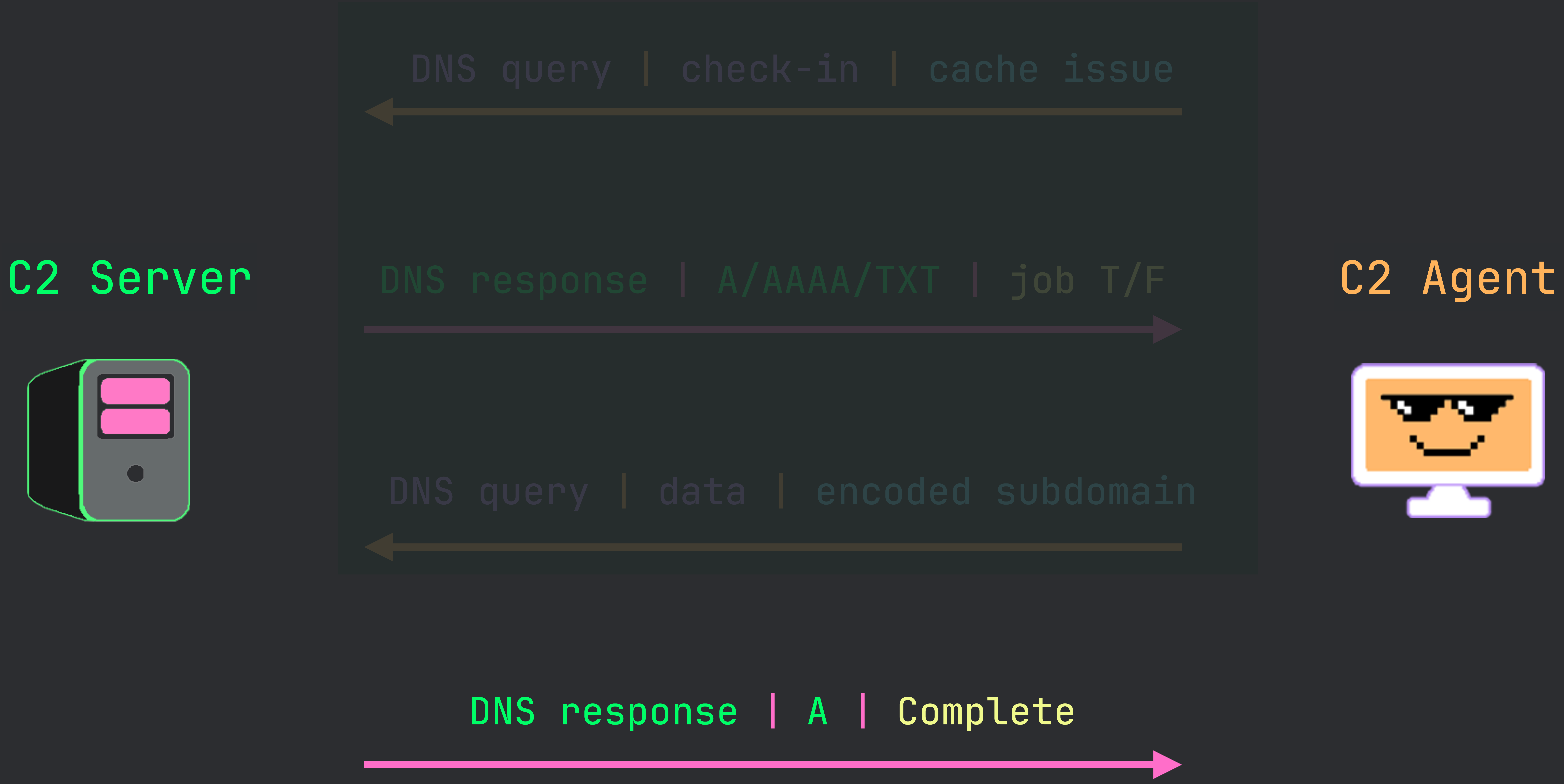
DNS query | check-in | cache issue

C2 Server

DNS response | A/AAAA/TXT | job T/F

C2 Agent

DNS query | data | encoded subdomain



C2 Server

C2 Agent

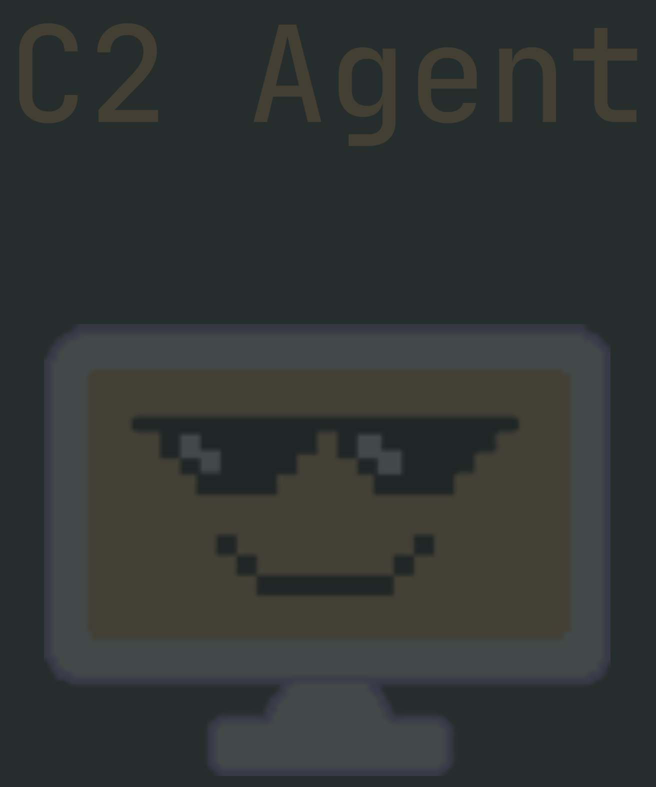
DNS query | check-in | cache issue

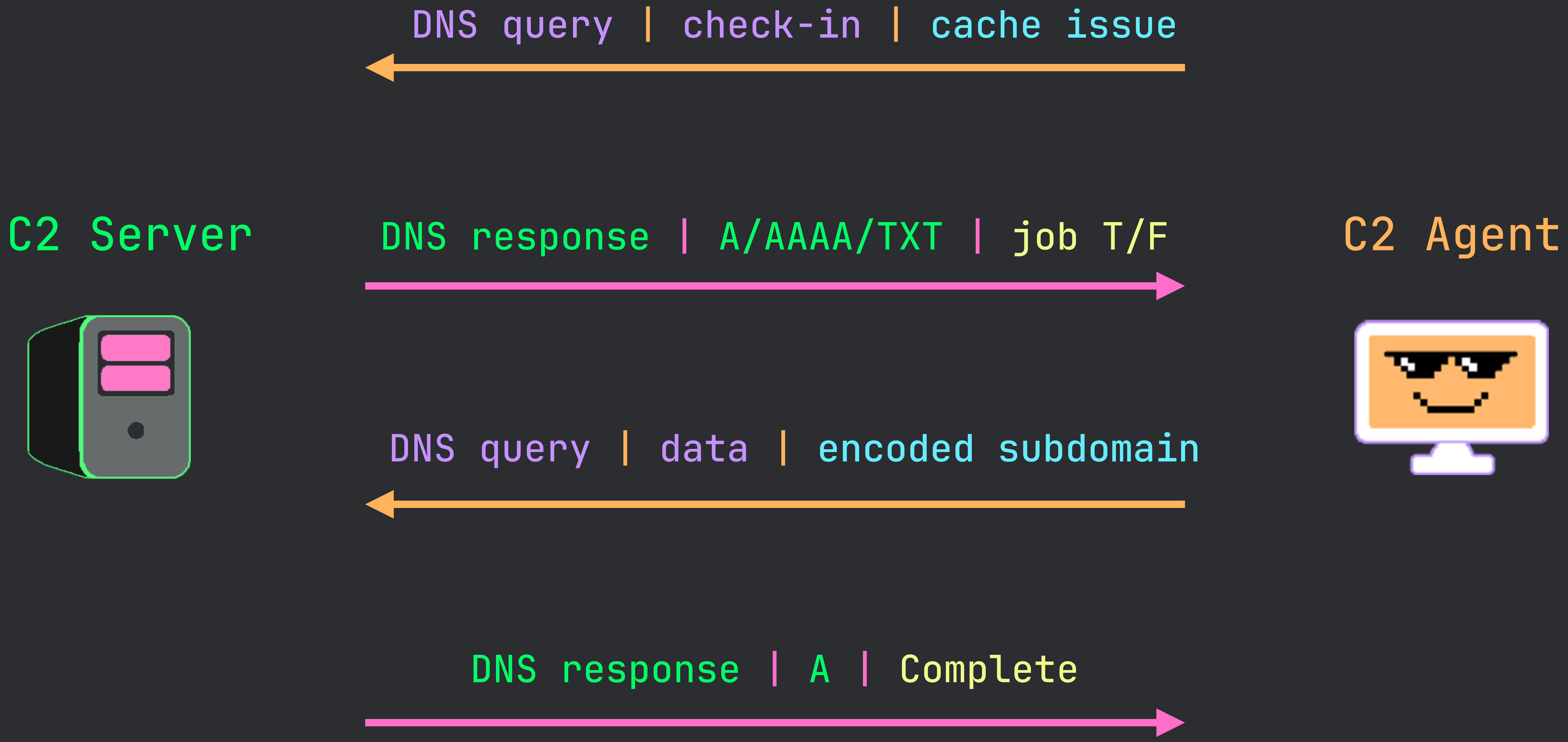
DNS response | A/AAAA/TXT | job T/F

DNS query | data | encoded subdomain

DNS response | A | Complete

now, let's talk more about how data is sent from agent → server





DNS query | check-in | cache issue

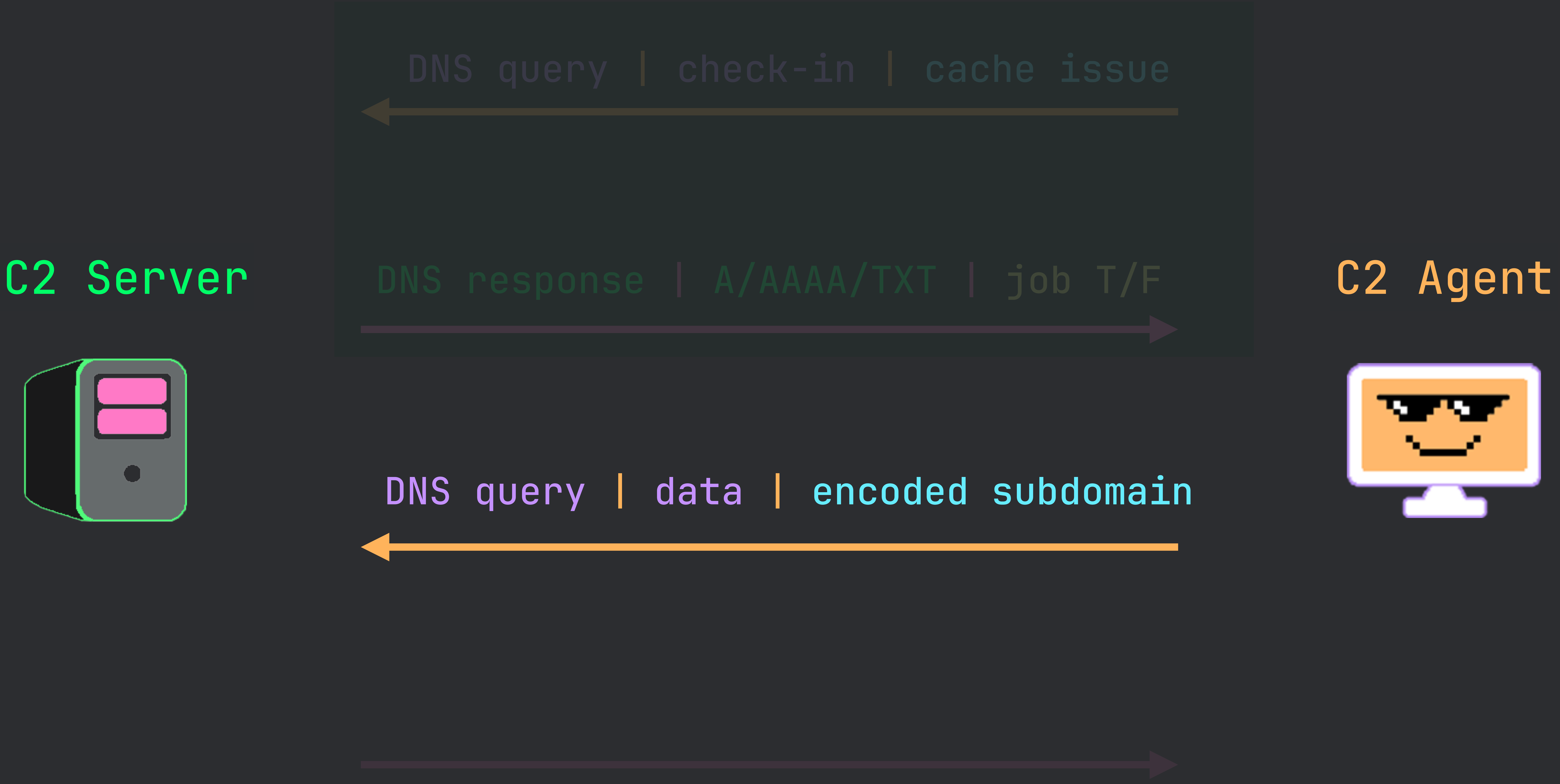
C2 Server

DNS response | A/AAAA/TXT | job T/F

C2 Agent

DNS query | data | encoded subdomain

DNS response | A | Complete



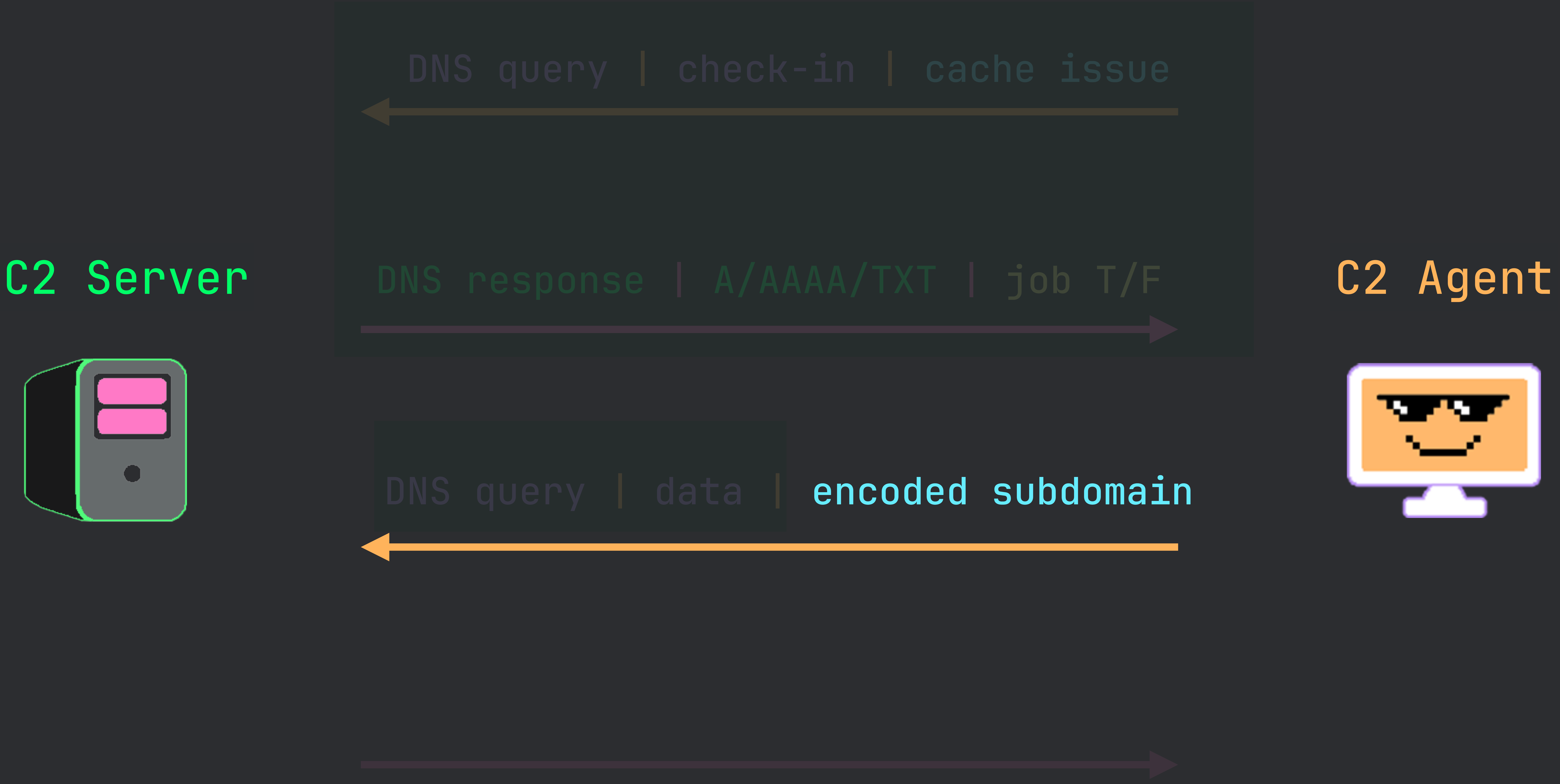
C2 Server

C2 Agent

DNS query | check-in | cache issue

DNS response | A/AAAA/TXT | job T/F

DNS query | data | encoded subdomain



C2 Server

DNS query | check-in | cache issue

DNS response | A/AAAA/TXT | job T/F

DNS query | data | encoded subdomain

C2 Agent

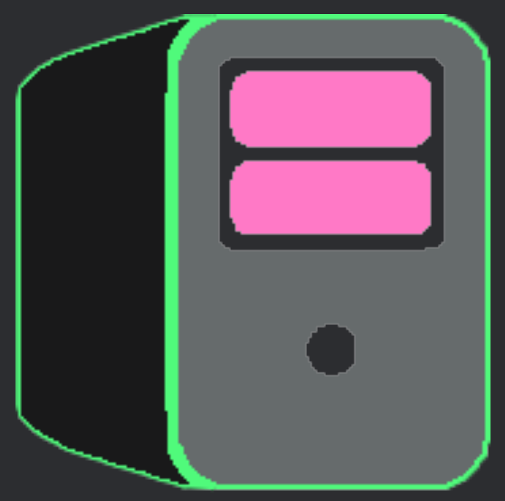
encoded subdomain

as data channel

the C2 agent sends a DNS query

it’s requesting to
resolve a domain

C2 Server

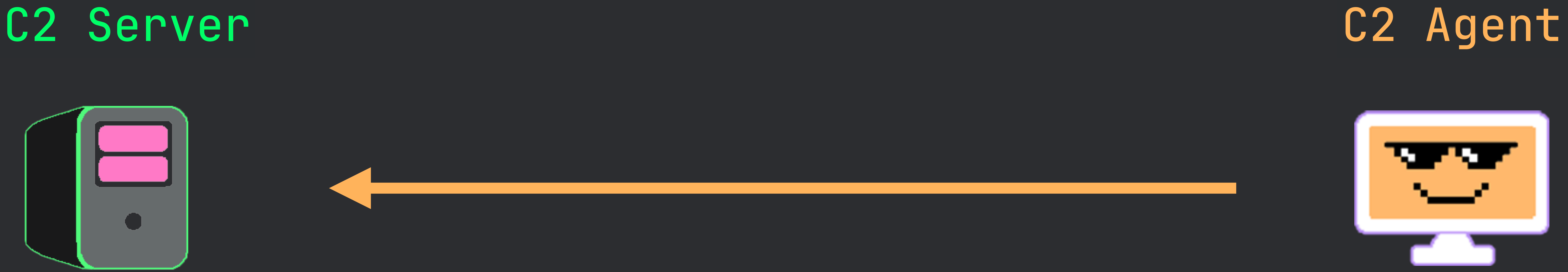


C2 Agent

C2 Server

C2 Agent

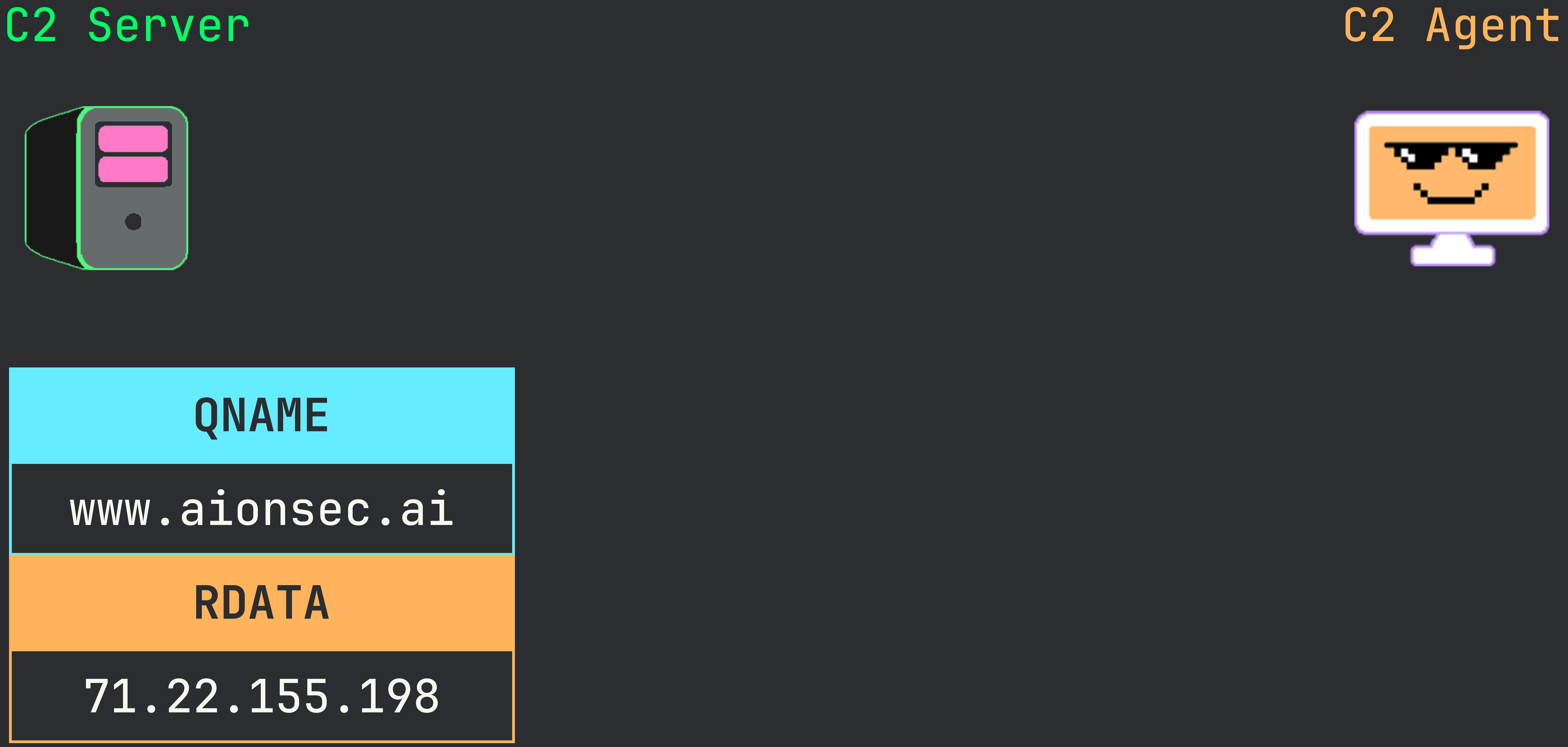
QNAME
www.aionsec.ai



C2 Server

C2 Agent

QNAME
www.aionsec.ai



C2 Server

C2 Agent

QNAME

www.aionsec.ai

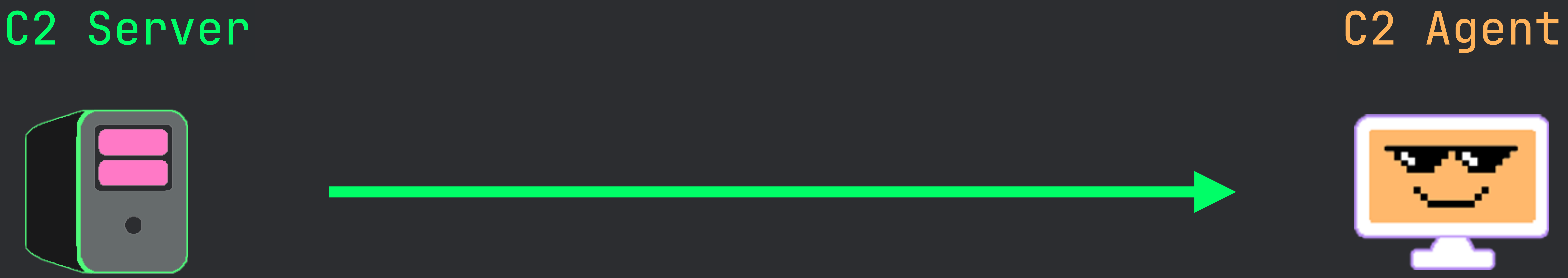
RDATA

71.22.155.198

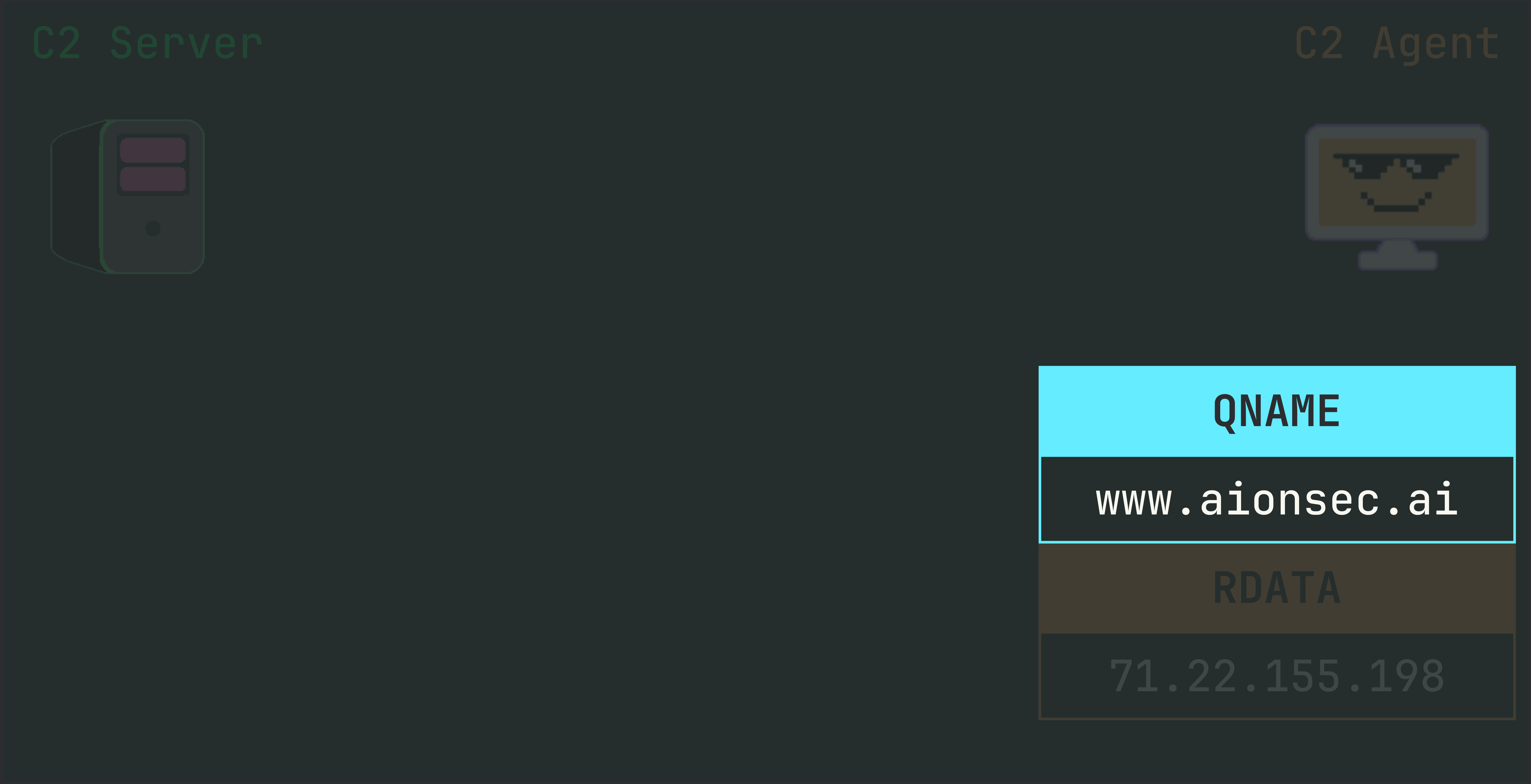
C2 Server

C2 Agent

QNAME
www.aionsec.ai
RDATA
71.22.155.198



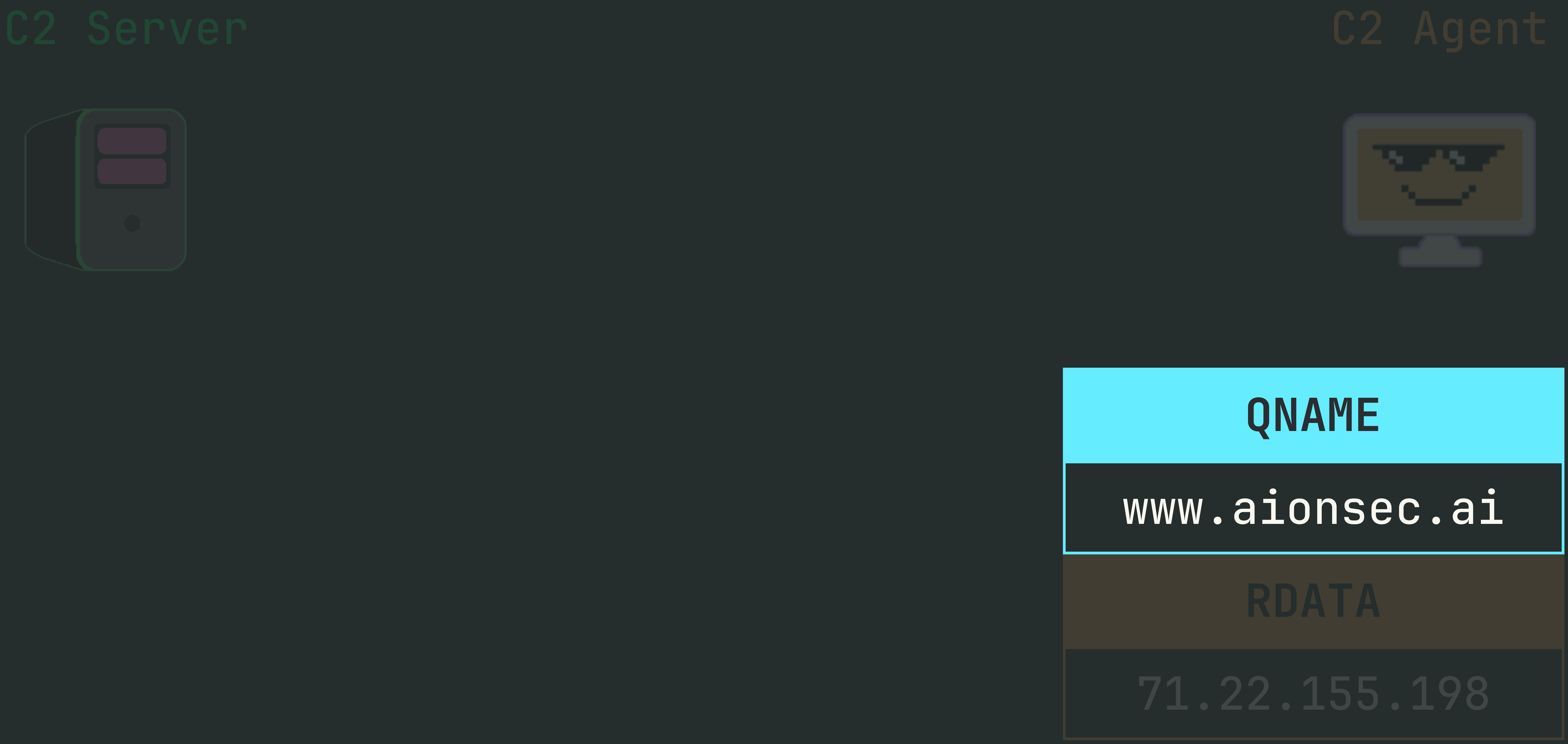
QNAME
www.aionsec.ai
RDATA
71.22.155.198



C2 Server

C2 Agent

QNAME
www.aionsec.ai
RDATA
71.22.155.198



C2 Server

C2 Agent

QNAME

www.aionsec.ai

RDATA

71.22.155.198

www.aionsec.ai

www.aionsec.ai

<subdomain>.aionsec.ai

<subdomain>

<subdomain>

→ 63 chars (“label”)

→ encoded data

<subdomain>

for ex dnscat2…

e7f1018ea0310f25bba0610936fd1cc2af

for ex dnscat2…

e7f1018ea0310f25bba0610936fd1cc2af

→ 63 chars capacity

→ 34 hex

```
e7f1  018e  a0  310f25bba0610936fd1cc2af
```

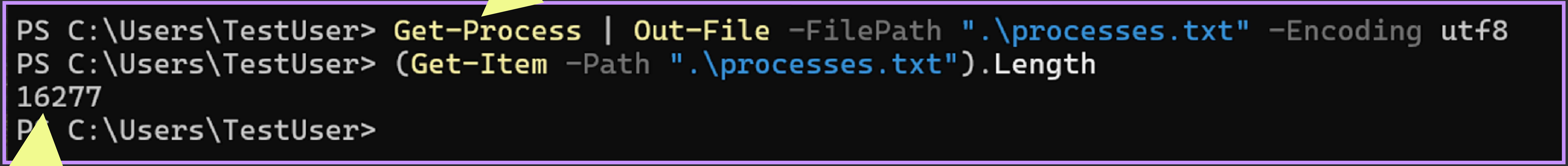
```
e7f1  018e  a0  310f25bba0610936fd1cc2af
```

→ Actual Payload

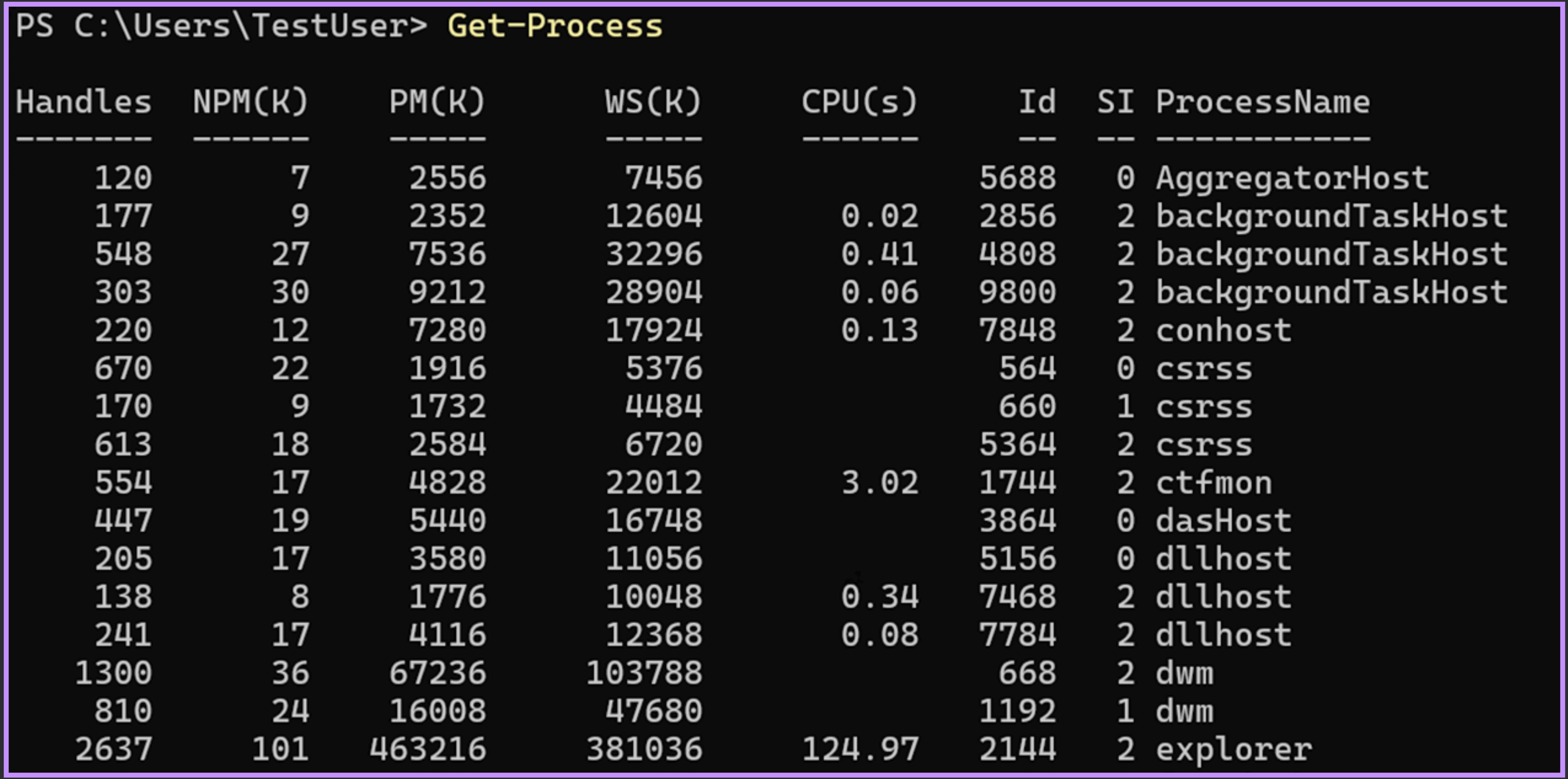
→ 24 hex chars

→ 12 bytes

12 bytes



```
PS C:\Users\TestUser> Get-Process | Out-File -FilePath ".\processes.txt" -Encoding utf8
PS C:\Users\TestUser> (Get-Item -Path ".\processes.txt").Length
16277
PS C:\Users\TestUser>
```



```
PS C:\Users\TestUser> Get-Process

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    120       7     2556       7456              5688   0 AggregatorHost
    177       9     2352      12604       0.02   2856   2 backgroundTaskHost
    548      27     7536      32296       0.41   4808   2 backgroundTaskHost
    303      30     9212      28904       0.06   9800   2 backgroundTaskHost
    220      12     7280      17924       0.13   7848   2 conhost
    670      22     1916       5376               564   0 csrss
    170       9     1732       4484               660   1 csrss
    613      18     2584       6720              5364   2 csrss
    554      17     4828      22012       3.02   1744   2 ctfmon
    447      19     5440      16748              3864   0 dasHost
    205      17     3580      11056              5156   0 dllhost
    138       8     1776      10048       0.34   7468   2 dllhost
    241      17     4116      12368       0.08   7784   2 dllhost
   1300      36    67236     103788               668   2 dwm
    810      24    16008      47680              1192   1 dwm
   2637     101   463216     381036     124.97   2144   2 explorer
```

capacity

12 bytes

total

16277 bytes

---

capacity

12 bytes

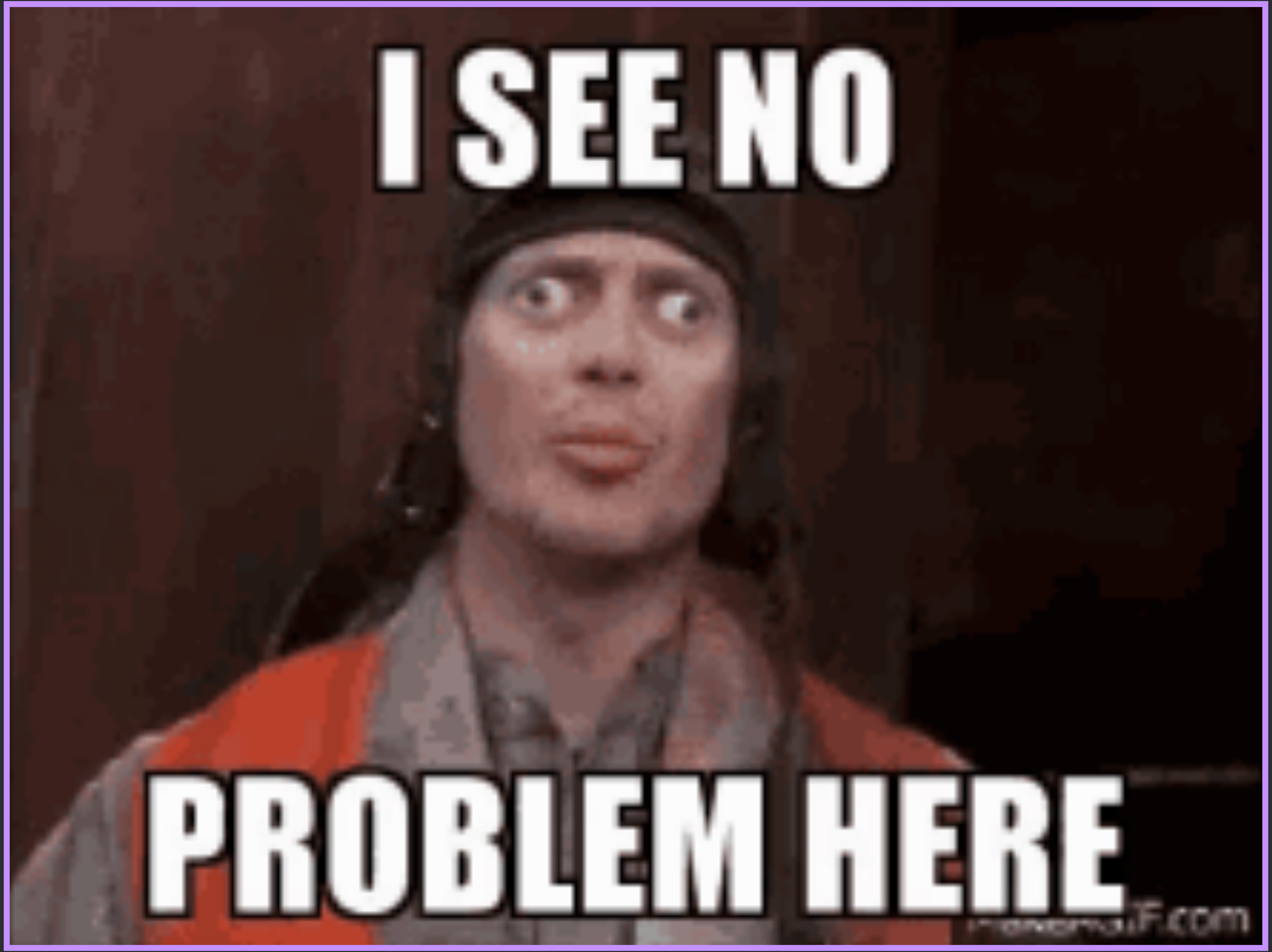
1356 queries

1356 subdomains

1356 unique FQDNs

1356 unique FQDNs

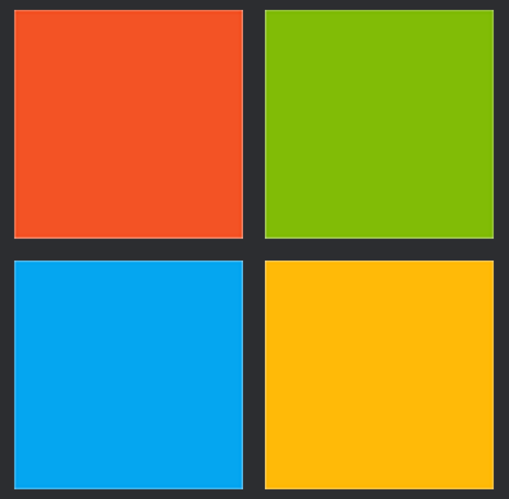
JUST FOR PIDs!


I SEE NO
PROBLEM HERE

the problem is…

over time you will have
10ks, 100ks, 1Ms+
unique FQDNS associated
with an unknown domain

a few 100 max



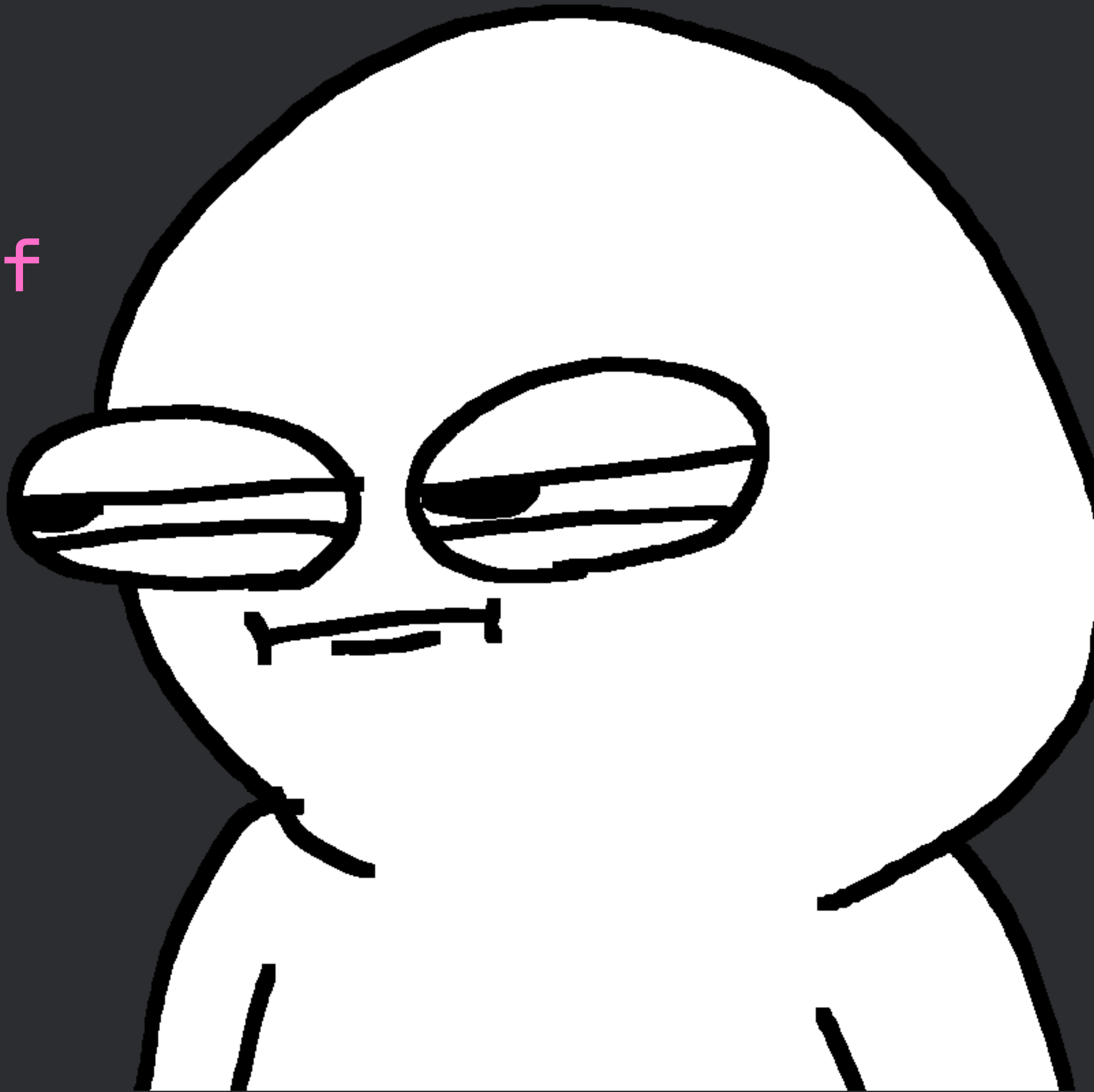
so when you have

xj40-defderp.com

with 800ks FQDNs…

especially if

```
e7f1018ea0310f25bba0610936fd1cc2af
```



so, for us as threat hunters
look for high unique FQDN count
showing high-entropy subdomains
associated with an unknown domain

"It's practically a solved problem."

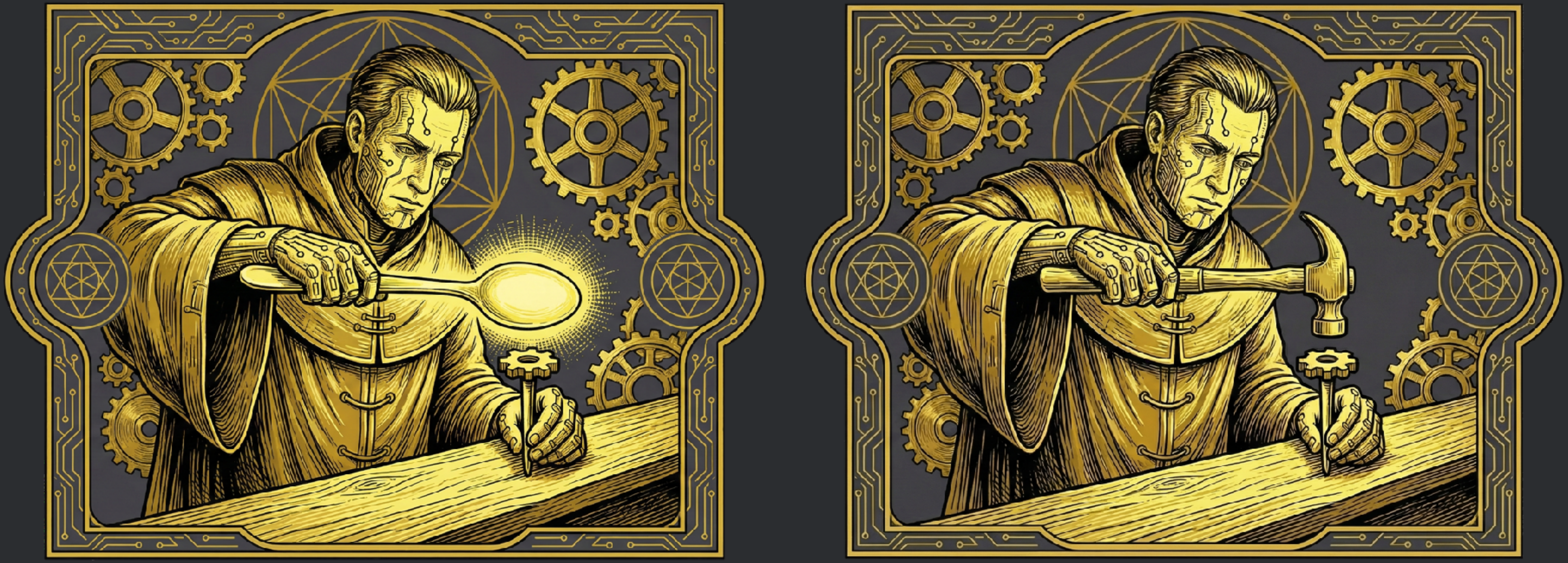
"It's practically a solved problem."

Except, it isn’t.

Two ways to use DNS as a covert channel

Two ways to use DNS as a covert channel

Two ways to use DNS as a covert channel



DNS is not high-bandwidth, don’t use it for that

Two ways to use DNS as a covert channel

encoded subdomains (exfil)

Two ways to use DNS as a covert channel

what we will look at today

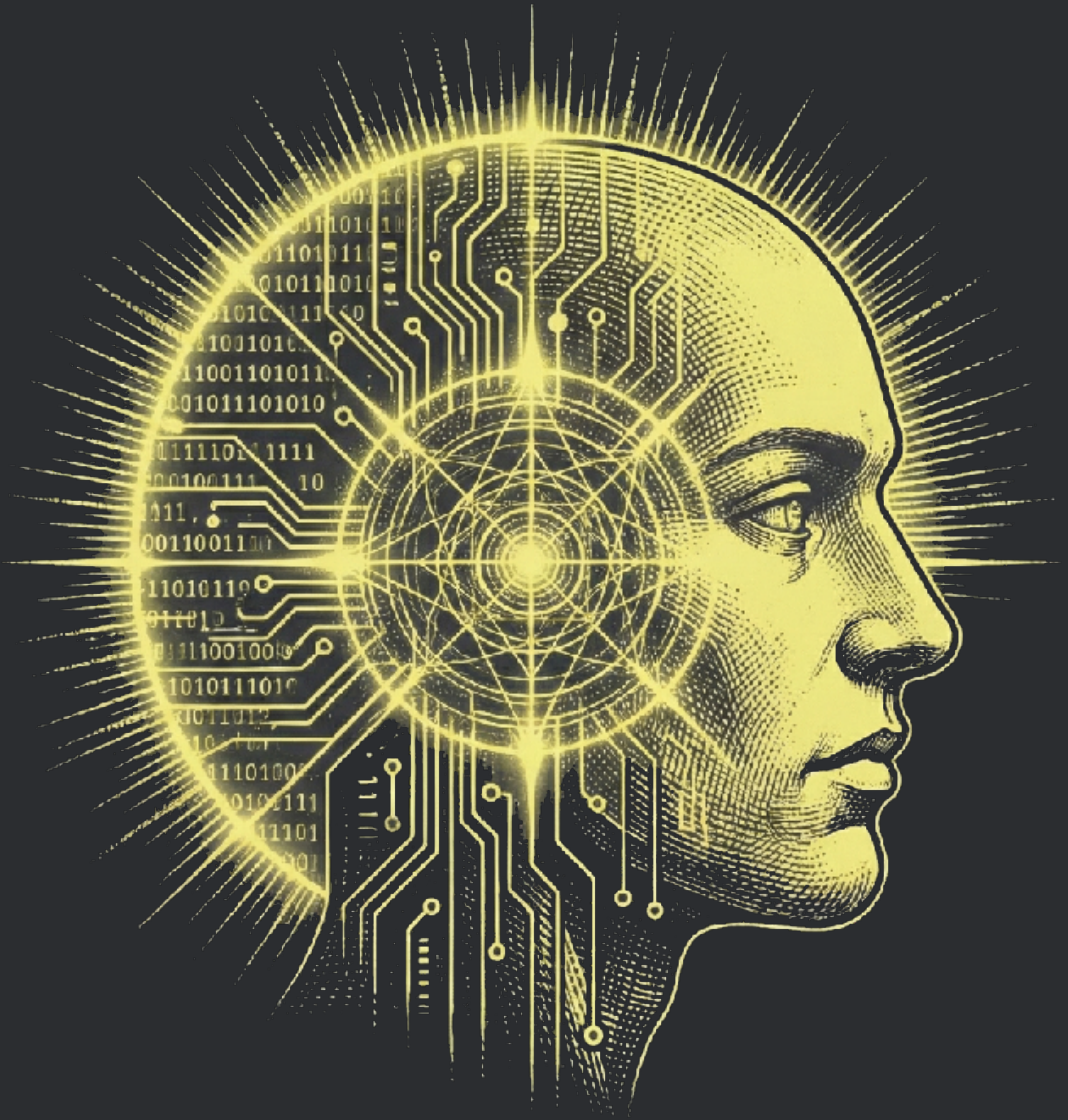
But if I can’t transfer lots of data

what’s even the point of using it?

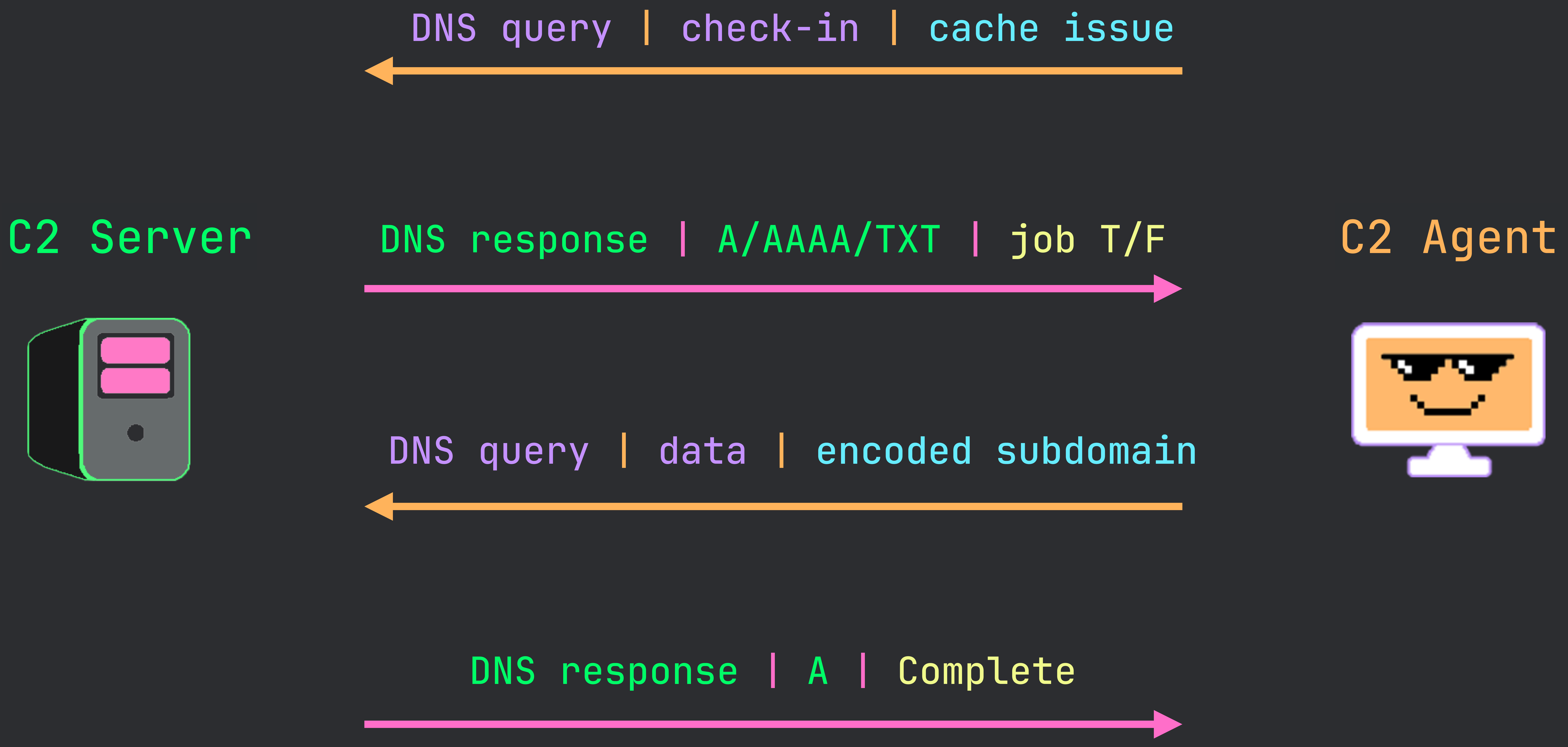
Start thinking “multi-modal”

what we will look at today

- TXT Record Abuse
- NULL Record Abuse
- CNAME, MX, SRV etc
- DNS Sandwich
- ID Field Abuse
- EDNS0
- Encrypted Channels



TXT Record Abuse



DNS query | check-in | cache issue

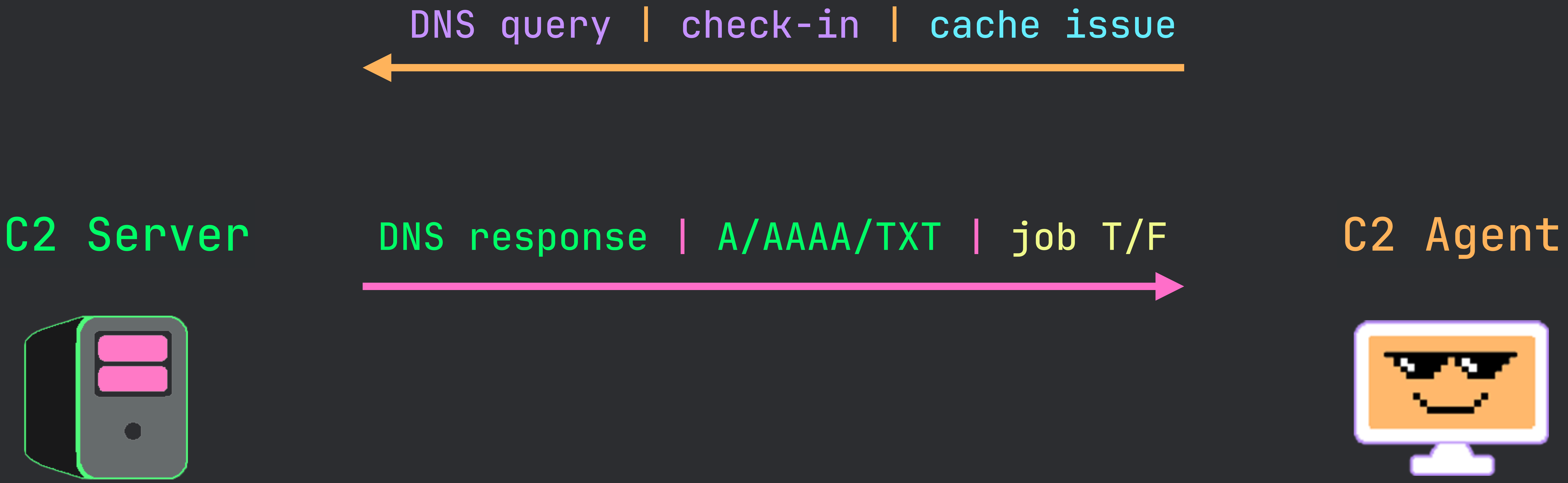
C2 Server

DNS response | A/AAAA/TXT | job T/F

C2 Agent

DNS query | data | encoded subdomain

DNS response | A | Complete

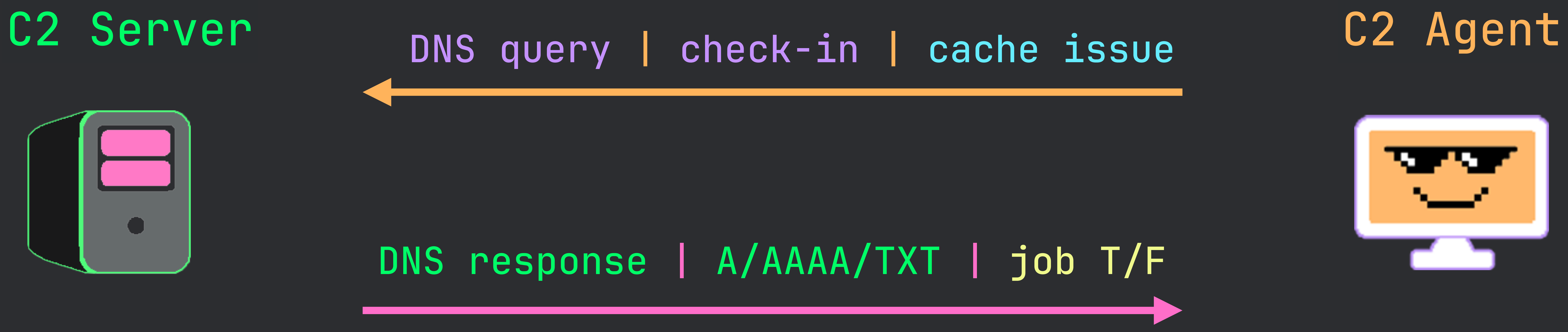


DNS query | check-in | cache issue

C2 Server

DNS response | A/AAAA/TXT | job T/F

C2 Agent

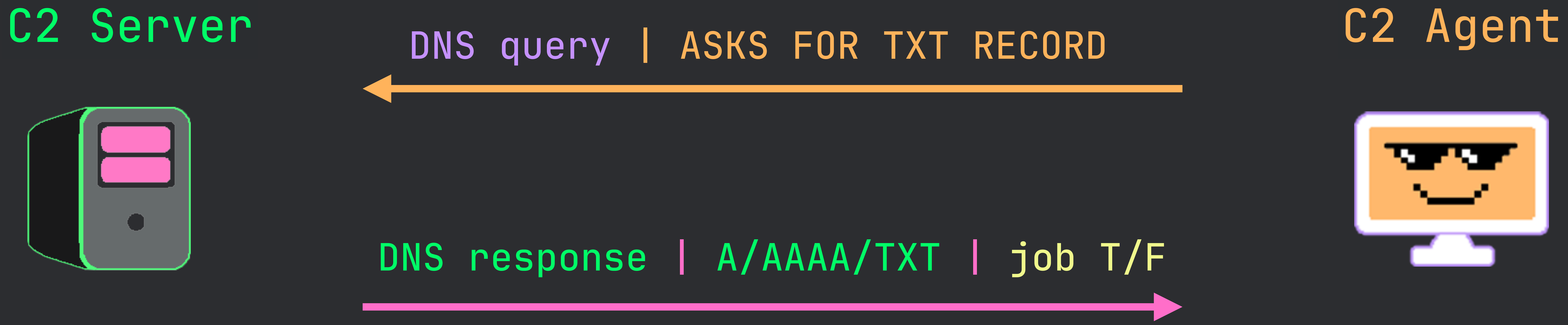


C2 Server

C2 Agent

DNS query | check-in | cache issue

DNS response | A/AAAA/TXT | job T/F

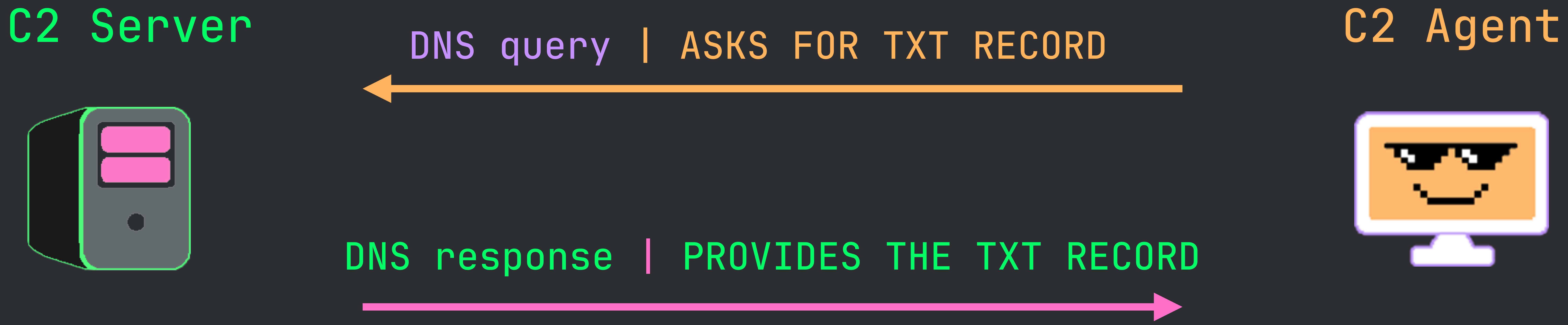


C2 Server

DNS query | ASKS FOR TXT RECORD

C2 Agent

DNS response | A/AAAA/TXT | job T/F

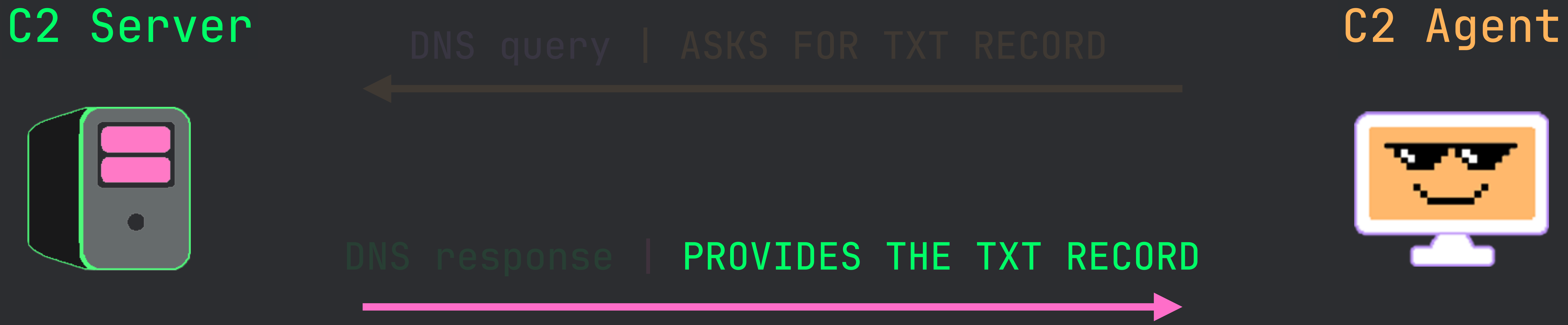


C2 Server

C2 Agent

DNS query | ASKS FOR TXT RECORD

DNS response | PROVIDES THE TXT RECORD



C2 Server

C2 Agent

DNS query | ASKS FOR TXT RECORD

DNS response | PROVIDES THE TXT RECORD

The agent (typically) uses encoded subdomains for data transfer

The server (typically) sends data in the record itself

Currently the most popular

choice for this - TXT Records

Why is it popular?

Why is it popular?

| 255 char per string (A = 4 b | AAAA = 16 b)

| fairly common(-ish)

| multiple strings allowed

| domain verification - encoded blobs

Detection

- TXT records are not unusual
- But, a sudden deluge
- From a single ext host
- To a single int host (sus af)

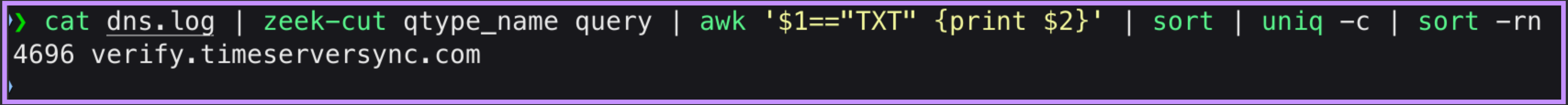
Zeek to the rescue

We can query dns.log and ask:

Show me all domains where TXT queries were sent

to, the amount, and sort by descending order

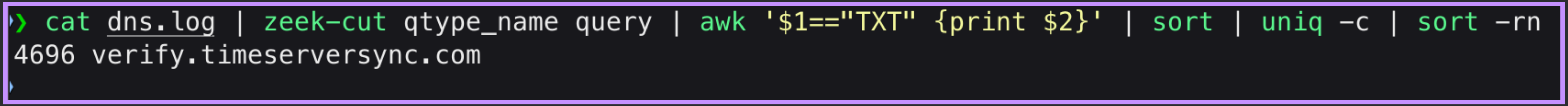
```
cat dns.log |

zeek-cut qtype_name query |

awk '$1=="TXT" {print $2}' |

sort |

uniq -c |

sort -rn
```



```
❯ cat dns.log | zeek-cut qtype_name query | awk '$1=="TXT" {print $2}' | sort | uniq -c | sort -rn
4696 verify.timeserversync.com
❯
```

```
cat dns.log |

zeek-cut qtype_name query |

awk '$1=="TXT" {print $2}' |

sort |

uniq -c |

sort -rn
```

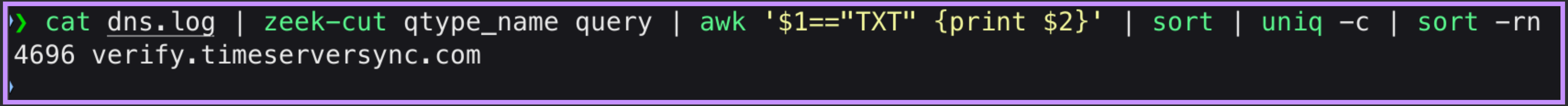
```
4696 verify.timeserversync.com

89 _dmarc.company-domain.com

45 default._domainkey.google.com

12 _verification.microsoft.com

3 amazonses.com

1 mailer.subs.com
```



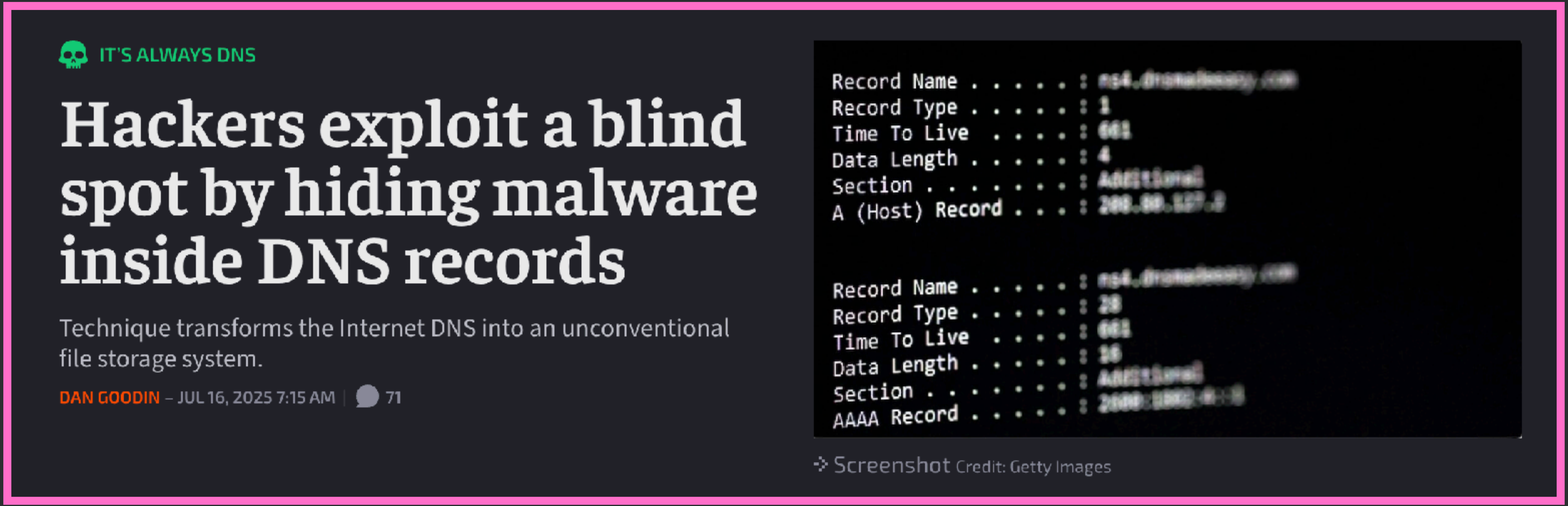
```
❯ cat dns.log | zeek-cut qtype_name query | awk '$1=="TXT" {print $2}' | sort | uniq -c | sort -rn
4696 verify.timeserversync.com
❯
```

```
cat dns.log |

zeek-cut qtype_name query |

awk '$1=="TXT" {print $2}' |

sort |

uniq -c |

sort -rn
```

```
4696 verify.timeserversync.com

89 _dmarc.company-domain.com

45 default._domainkey.google.com

12 _verification.microsoft.com

3 amazonses.com

1 mailer.subs.com
```



```
❯ cat dns.log | zeek-cut qtype_name query | awk '$1=="TXT" {print $2}' | sort | uniq -c | sort -rn
4696 verify.timeserversync.com
❯
```



IT'S ALWAYS DNS

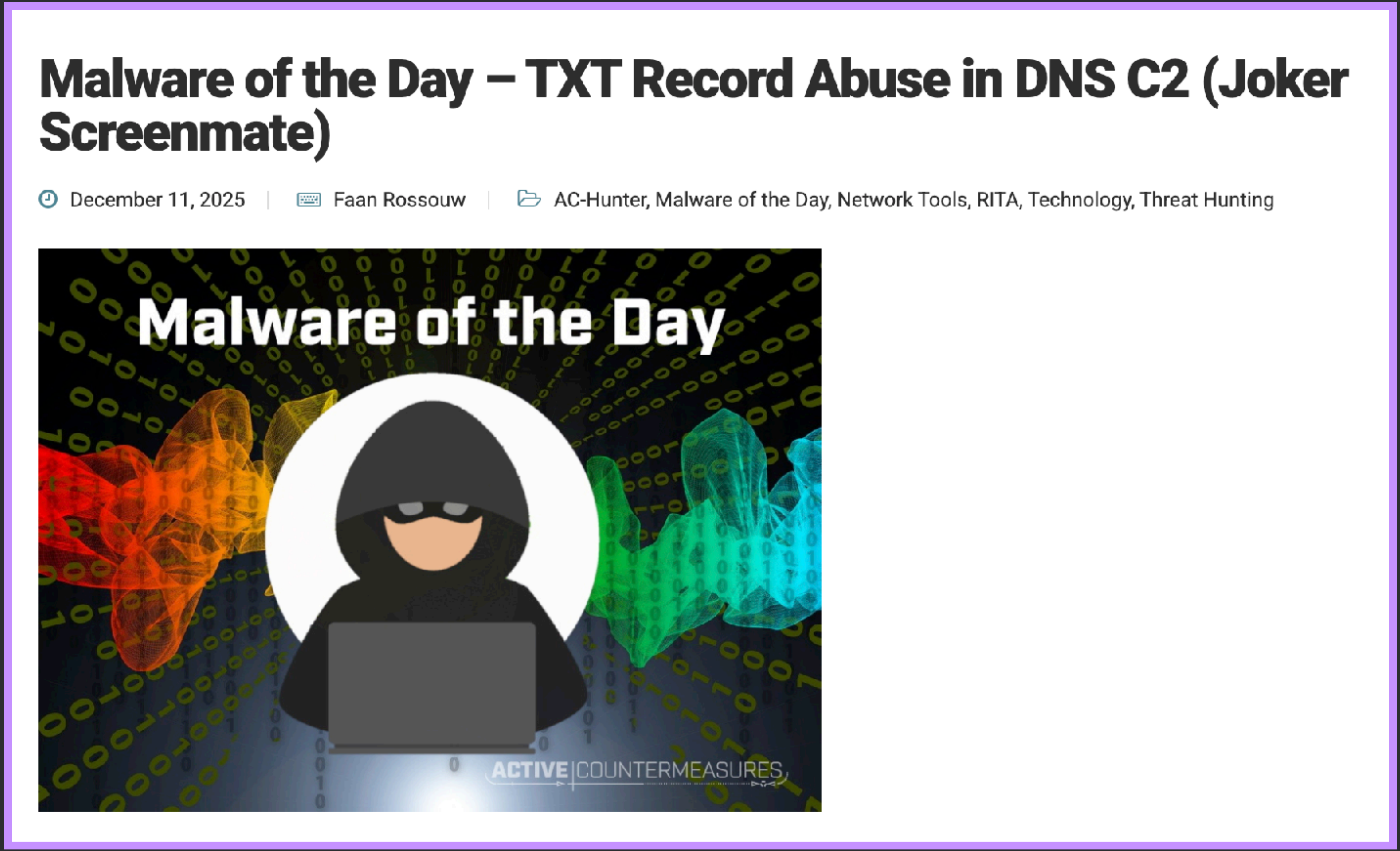
Hackers exploit a blind spot by hiding malware inside DNS records

Technique transforms the Internet DNS into an unconventional file storage system.

DAN GOODIN – JUL 16, 2025 7:15 AM | 71

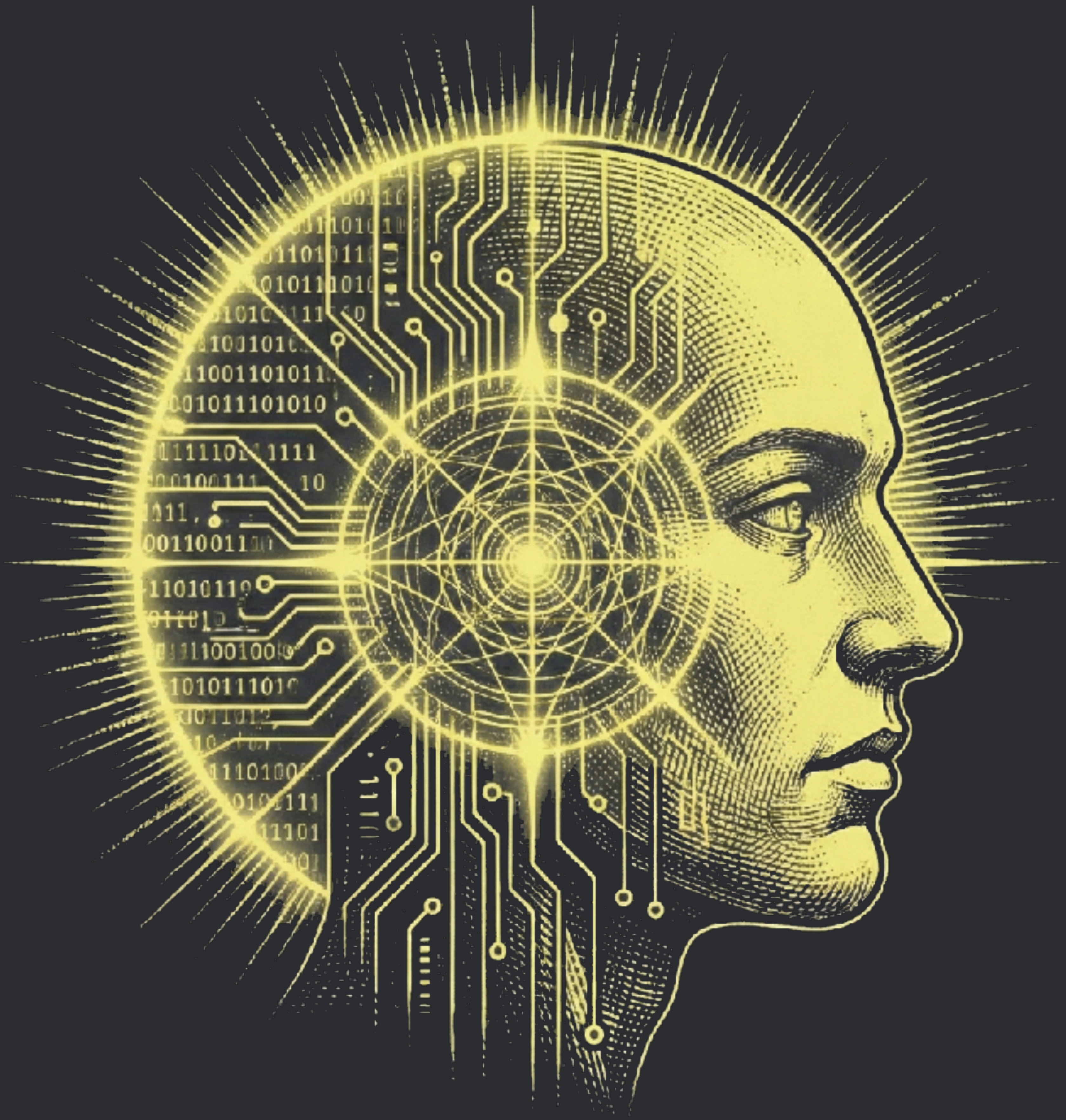
```
Record Name . . . . . . : [illegible]
Record Type . . . . . . : 1
Time To Live  . . . . . : [illegible]
Data Length . . . . . . : 4
Section . . . . . . . . : [illegible]
A (Host) Record . . . . : [illegible]


Record Name . . . . . . : [illegible]
Record Type . . . . . . : 28
Time To Live  . . . . . : [illegible]
Data Length . . . . . . : 16
Section . . . . . . . . : [illegible]
AAAA Record . . . . . . : [illegible]
```

Screenshot Credit: Getty Images



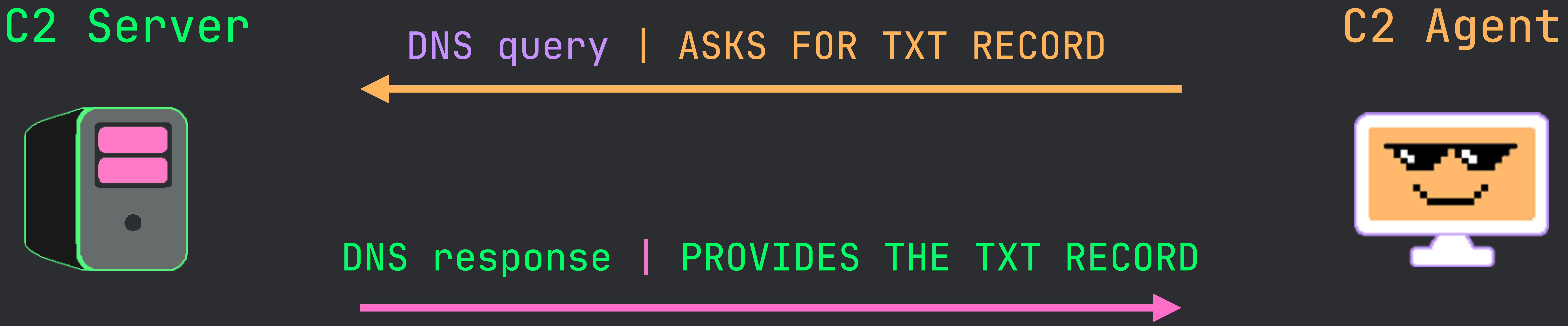
Malware of the Day – TXT Record Abuse in DNS C2 (Joker Screenmate)

December 11, 2025 | Faan Rossouw | AC-Hunter, Malware of the Day, Network Tools, RITA, Technology, Threat Hunting



NULL Record

```
we just established that:
| Agent → Srv = Encoded subdomains
| Srv → Agent = Actual record
```



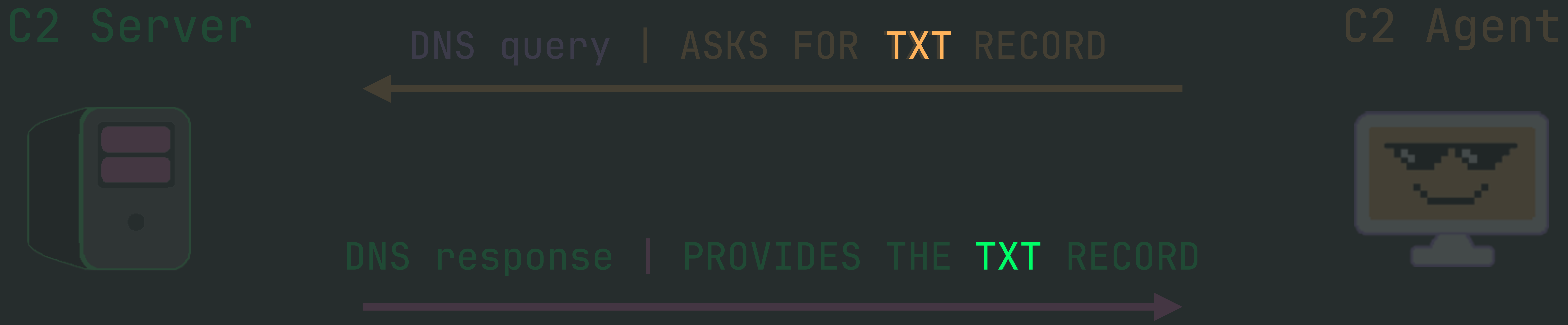
C2 Server

DNS query | ASKS FOR TXT RECORD

C2 Agent

DNS response | PROVIDES THE TXT RECORD

There are other options



C2 Server

DNS query | ASKS FOR TXT RECORD

C2 Agent

DNS response | PROVIDES THE TXT RECORD

NULL Record Abuse

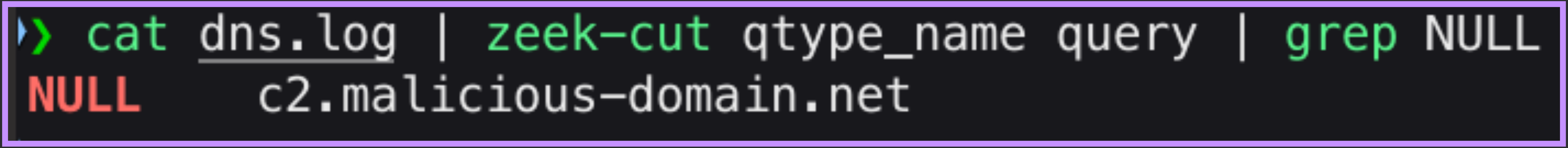
- Defined in RFC 1035 (1987)
- RDATA can contain “anything at all”
- Only record with no imposed structure
- Placeholder that was “reserved” (future)

Why Attacker Love(d) It

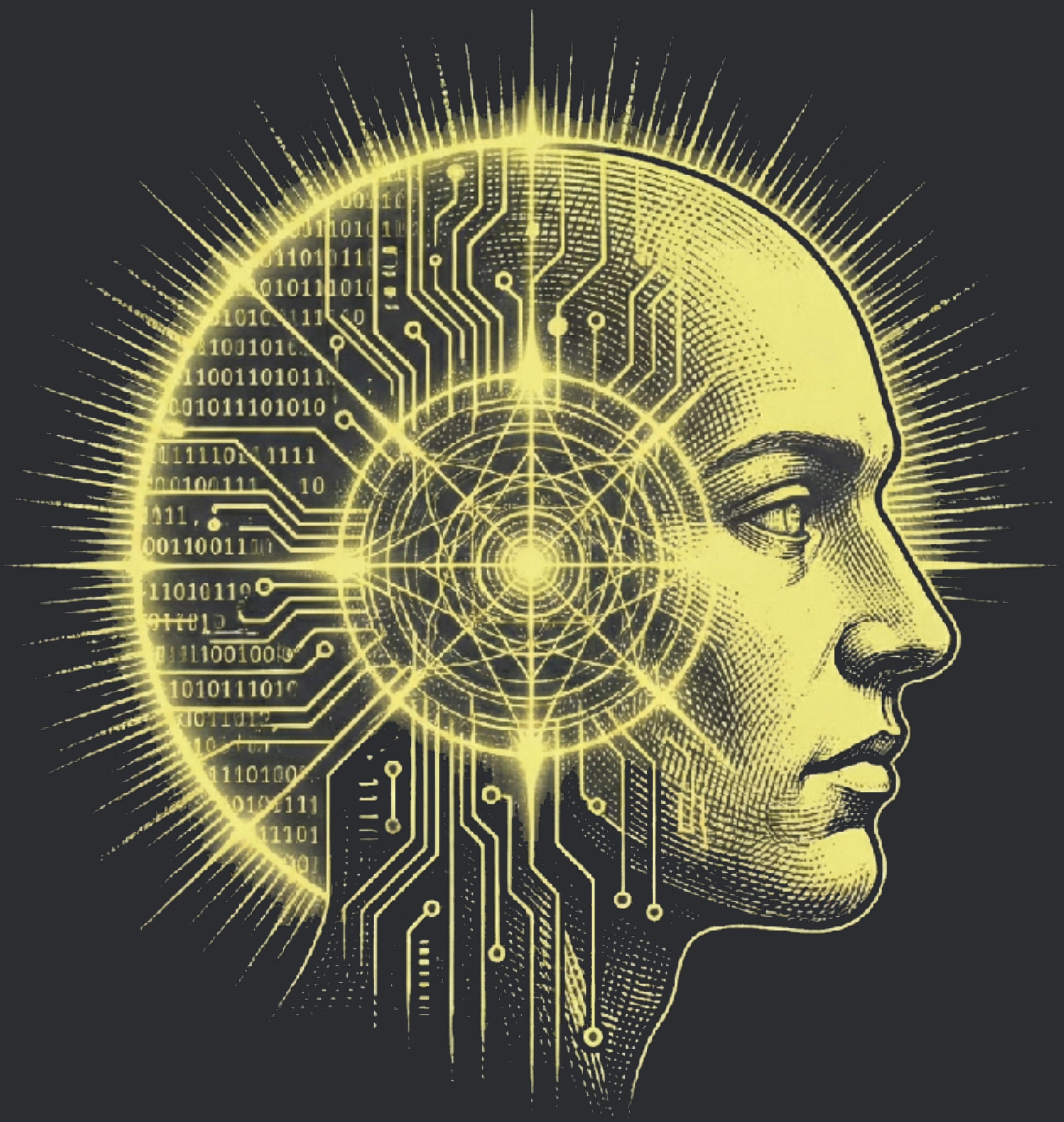
- Raw binary data - No encoding overhead
- Up to 65KB per response!
- Started off real popular, but…
- No legitimate use so…
- Simple: Flag ALL instances of use

Zeek to the rescue (again)

```
cat dns.log |

zeek-cut qtype_name query |

grep NULL
```



```
❯ cat dns.log | zeek-cut qtype_name query | grep NULL
NULL    c2.malicious-domain.net
```



CNAME, MX, SRV… Oh my

CNAME, MX, SRV… Oh my

- There are many types (80 ideal, 10-15 real)
- Almost any record can be used (in principle)
- Does not mean all are equally suited
- And those that are - diff tradeoffs
- Capacity ⟷ Stealth

CNAME, MX, SRV… Oh my

| These all return a hostname

| So can be abused in much the same way as exfil

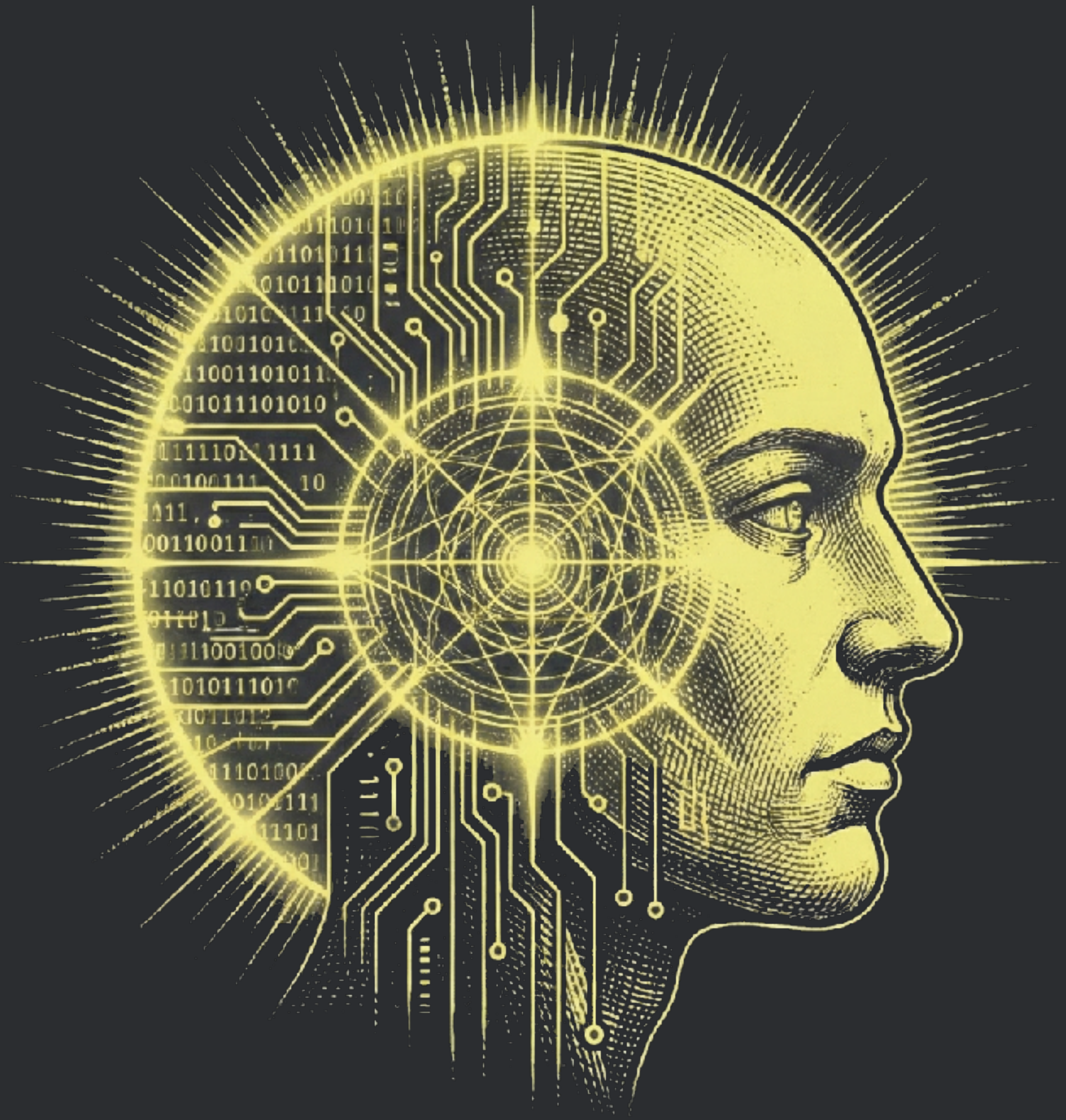
| <encoded-subdomain>.evil.com

| SRV = hostname + 3 numeric fields (+48 bits)

| Leads to same risk (high FQDN count + entropy)

The point remains

- Moving a lot of data has clear tells
- So know what to look for + look for it
- Inspect BOTH QNAME and RDATA for funky subs
- Zeek can detect most (bonus add ent)
- Add Zeek scripting and you’re at 99%



DNS Sandwich

So far we’ve considered 2 fields

QNAME for AGENT → SERVER

RDATA for SERVER → AGENT

But DNS has MANY fields!

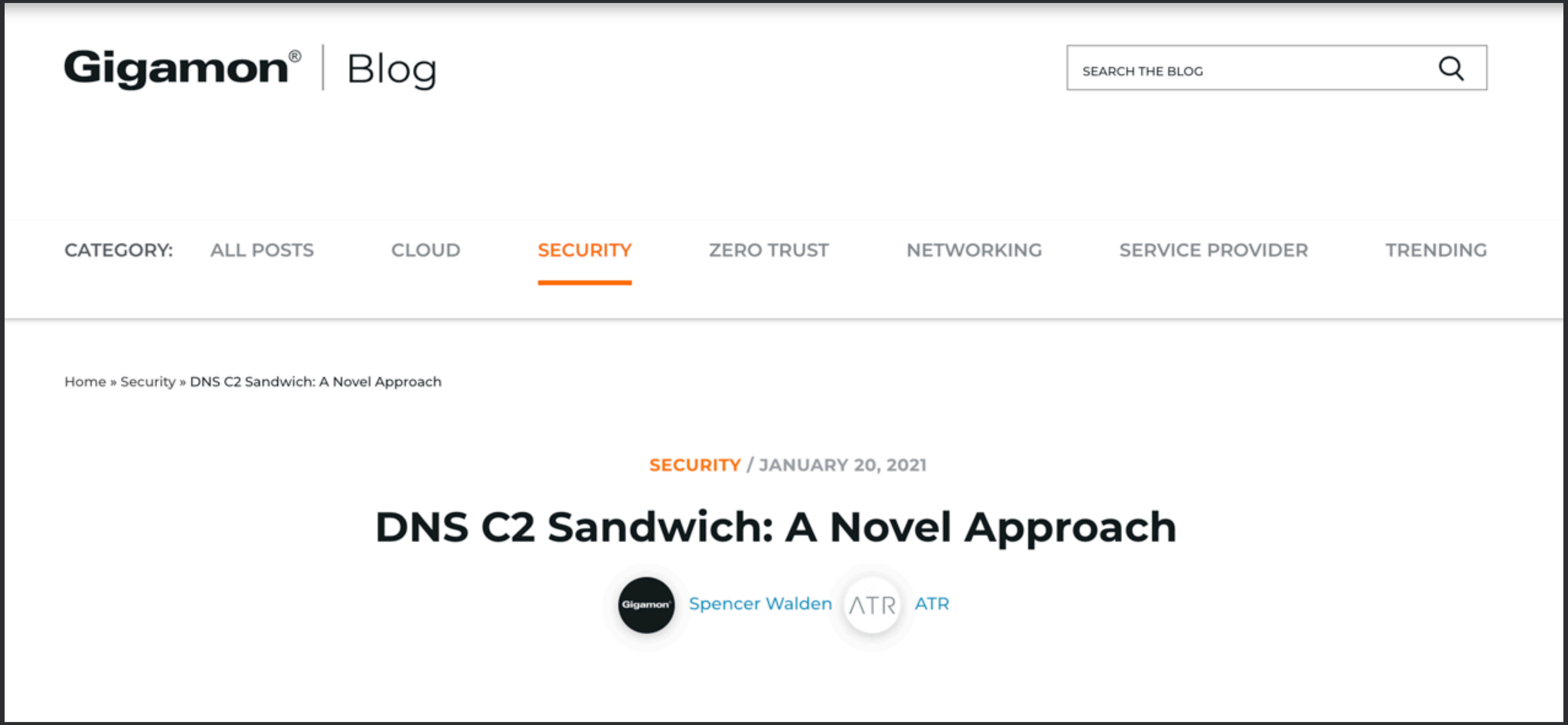
Does not mean you can use all of them to carry data, some will break

But a few will be ignored,

or can carry random data

DNS Sandwich defines 2

fields that are ignored



Gigamon® | Blog

SEARCH THE BLOG

CATEGORY: ALL POSTS CLOUD SECURITY ZERO TRUST NETWORKING SERVICE PROVIDER TRENDING

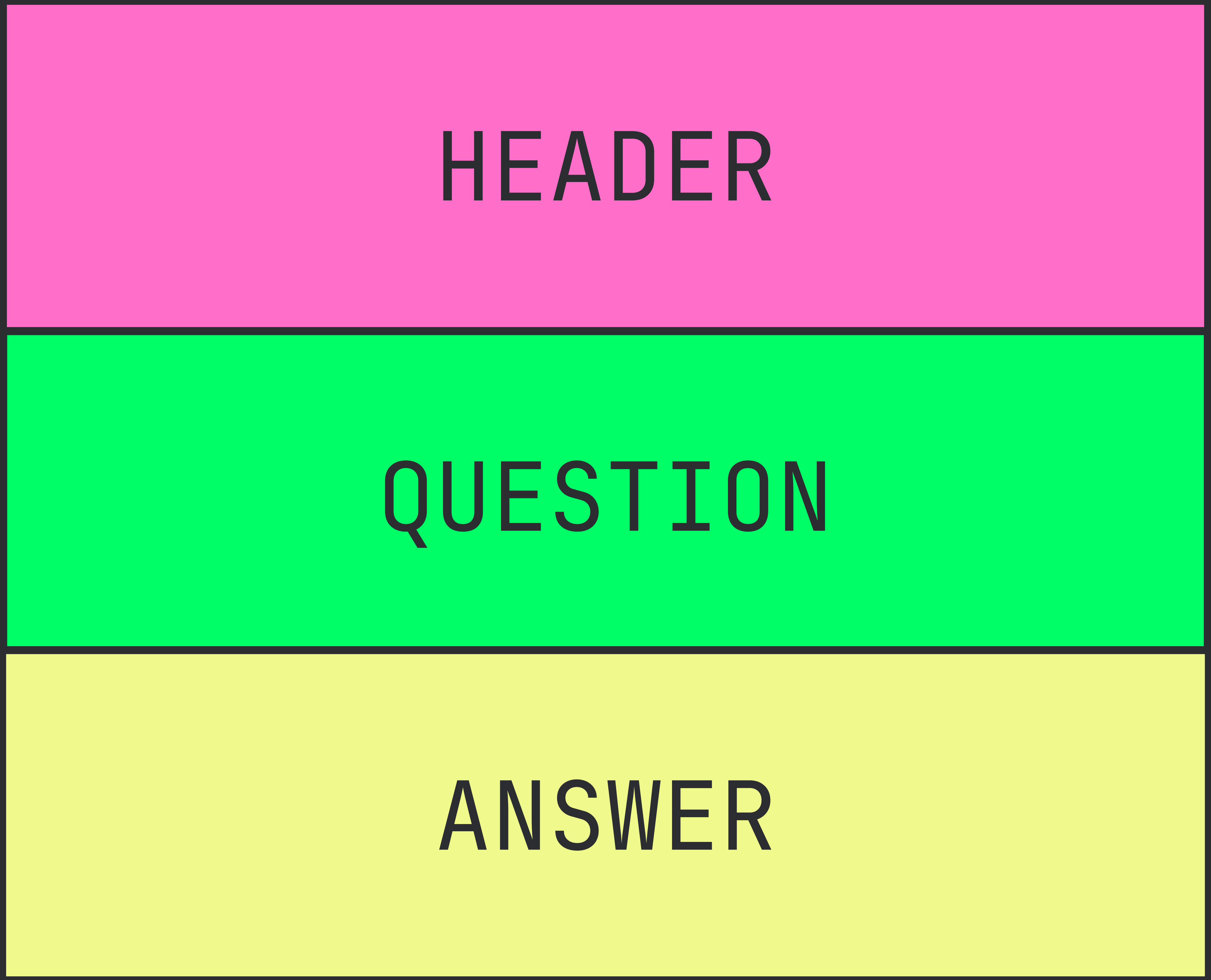
Home » Security » DNS C2 Sandwich: A Novel Approach

SECURITY / JANUARY 20, 2021

DNS C2 Sandwich: A Novel Approach

Spencer Walden ATR

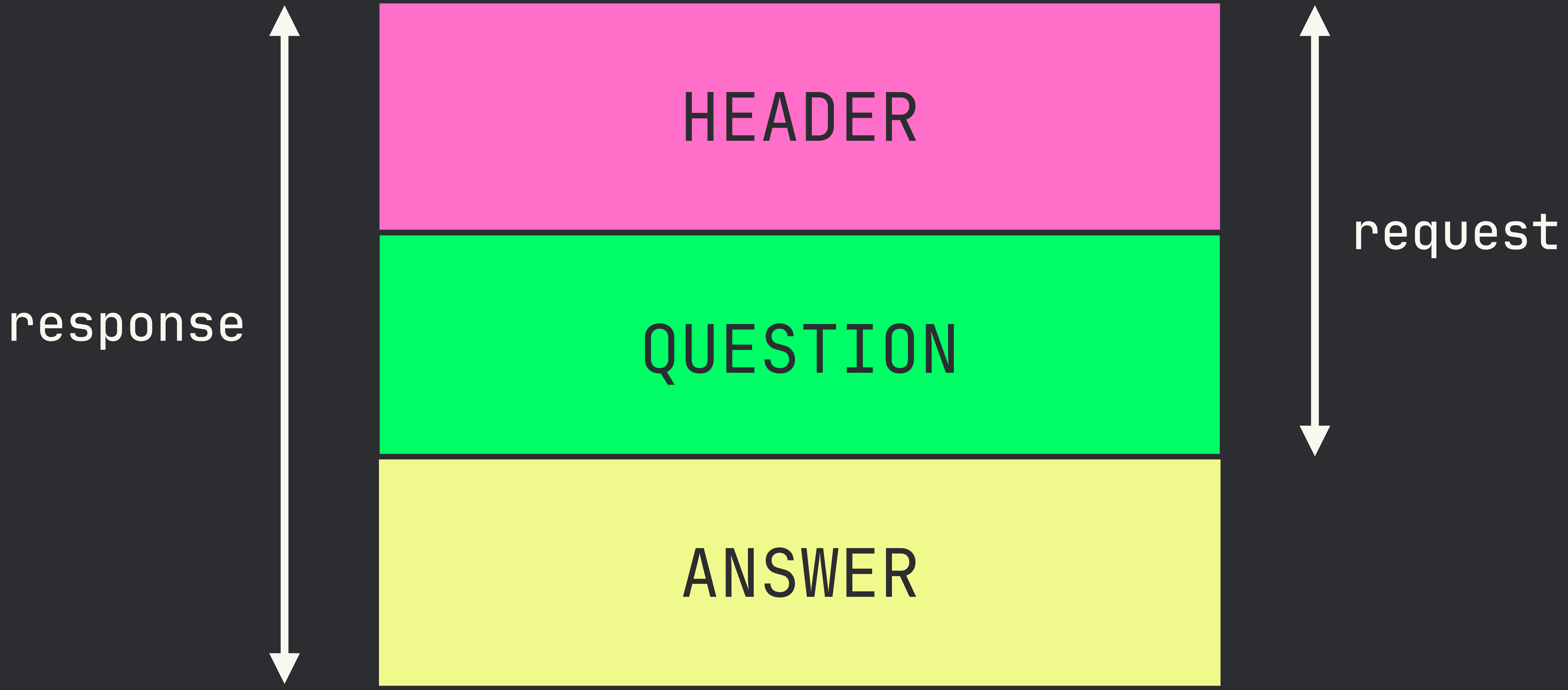
To understand, let’s just take a closer look at the structure of a DNS packet



HEADER

QUESTION

ANSWER



HEADER

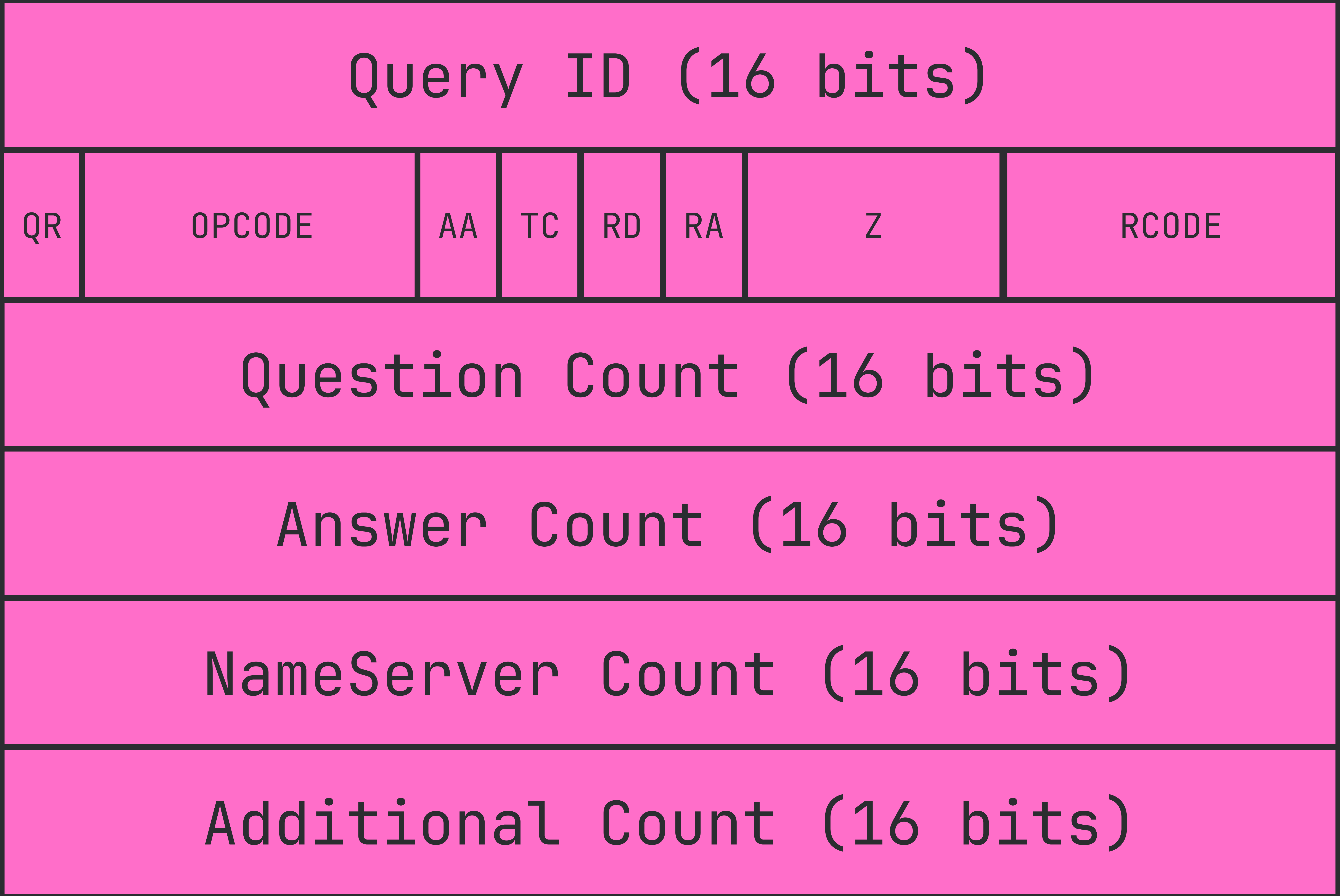
QUESTION

ANSWER

request

response

HEADER



Query ID (16 bits)

QR OPCODE AA TC RD RA Z RCODE

Question Count (16 bits)

Answer Count (16 bits)

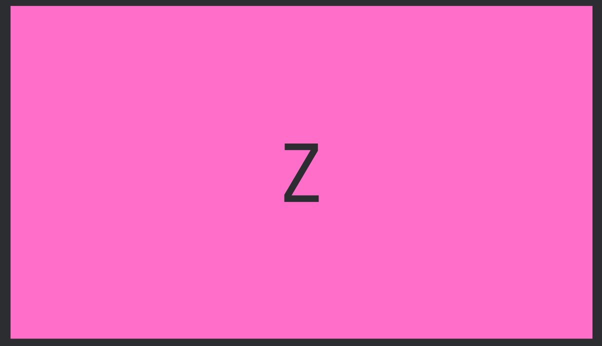
NameServer Count (16 bits)

Additional Count (16 bits)

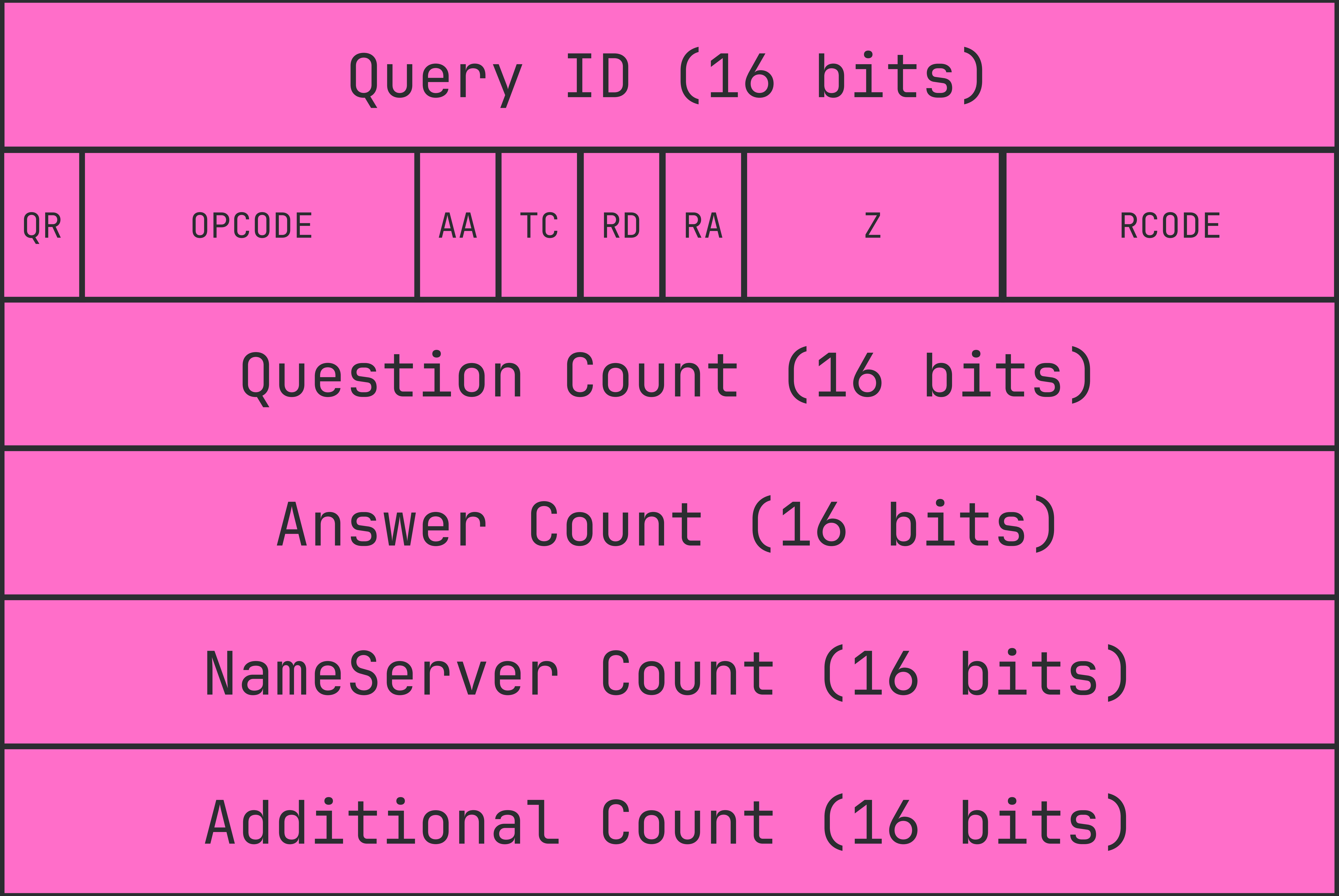
Z

Z Value

- → 3 bits reserved for future use
- → according to RFC - “must be 0”
- → most middlebox ignore (test!)



Z



Query ID (16 bits)

QR | OPCODE | AA | TC | RD | RA | Z | RCODE

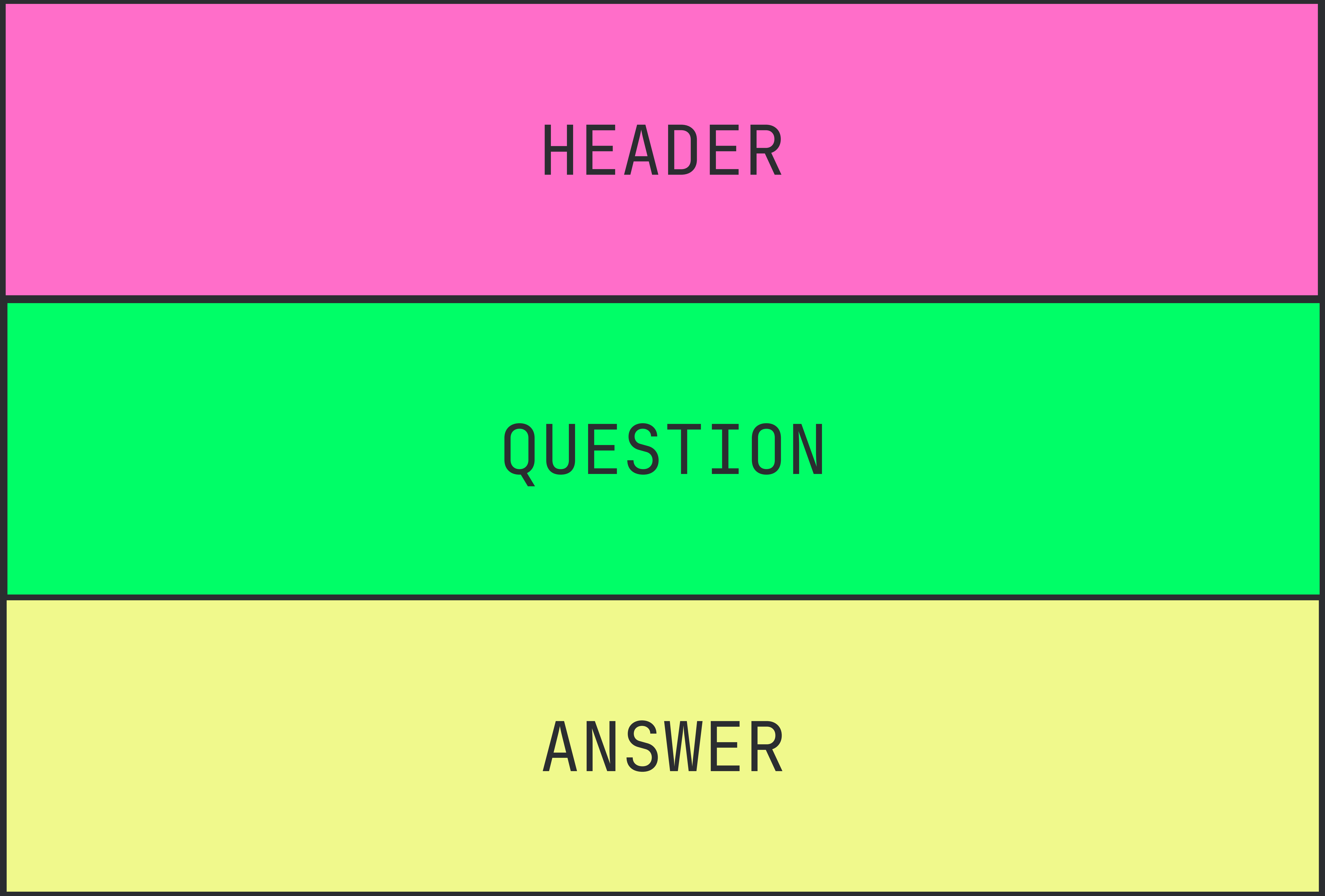
Question Count (16 bits)

Answer Count (16 bits)

NameServer Count (16 bits)

Additional Count (16 bits)

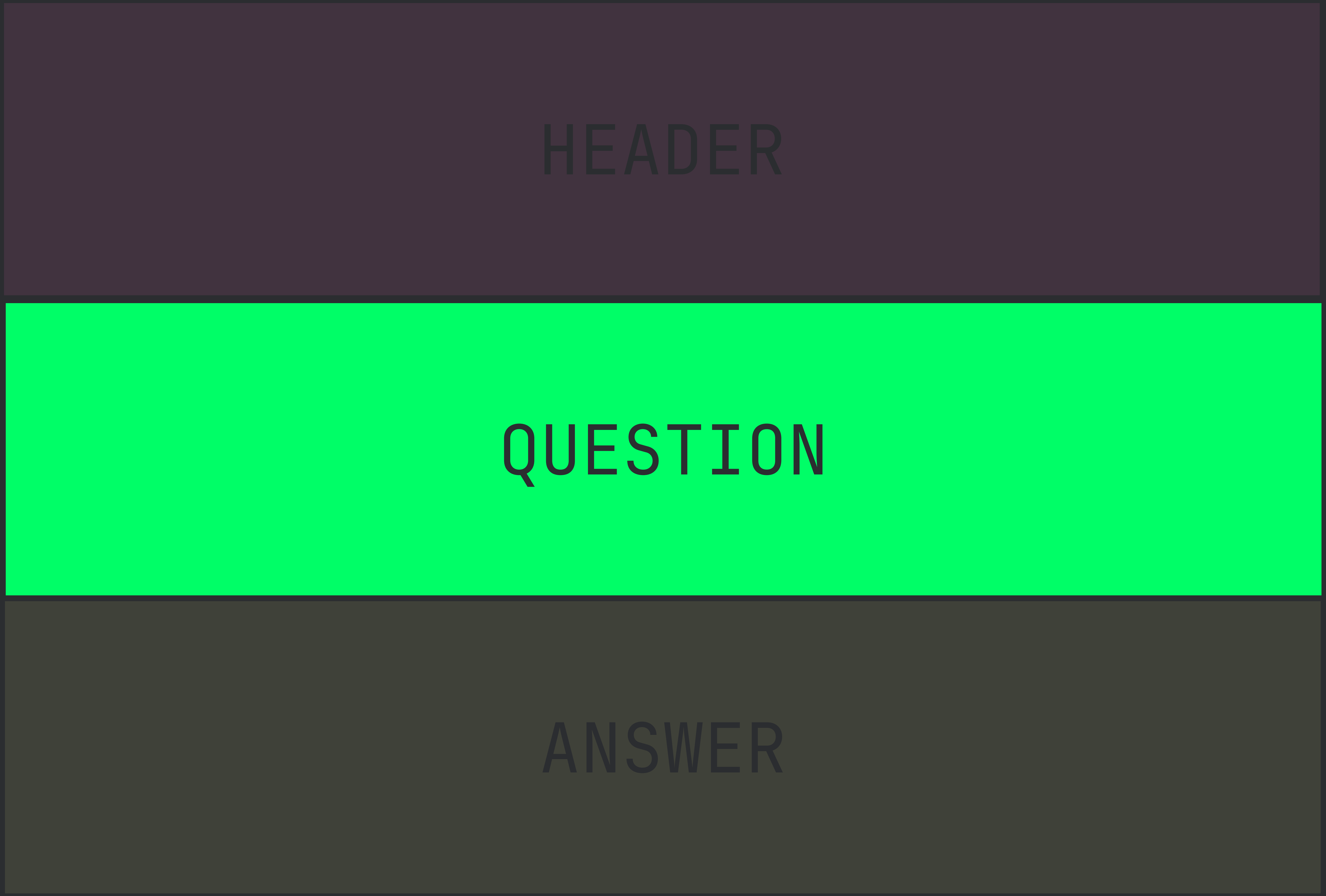
HEADER



HEADER

QUESTION

ANSWER

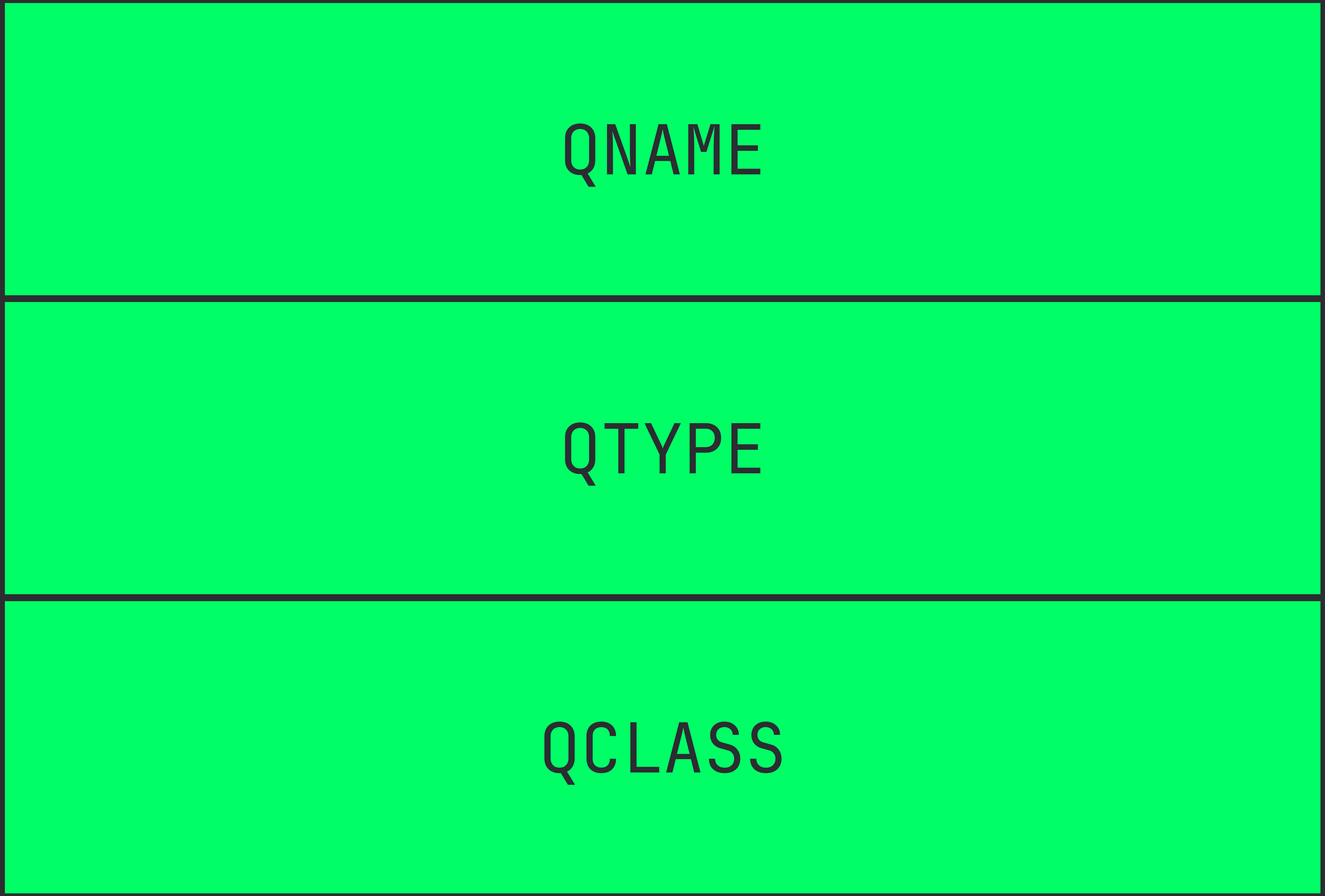


HEADER

QUESTION

ANSWER

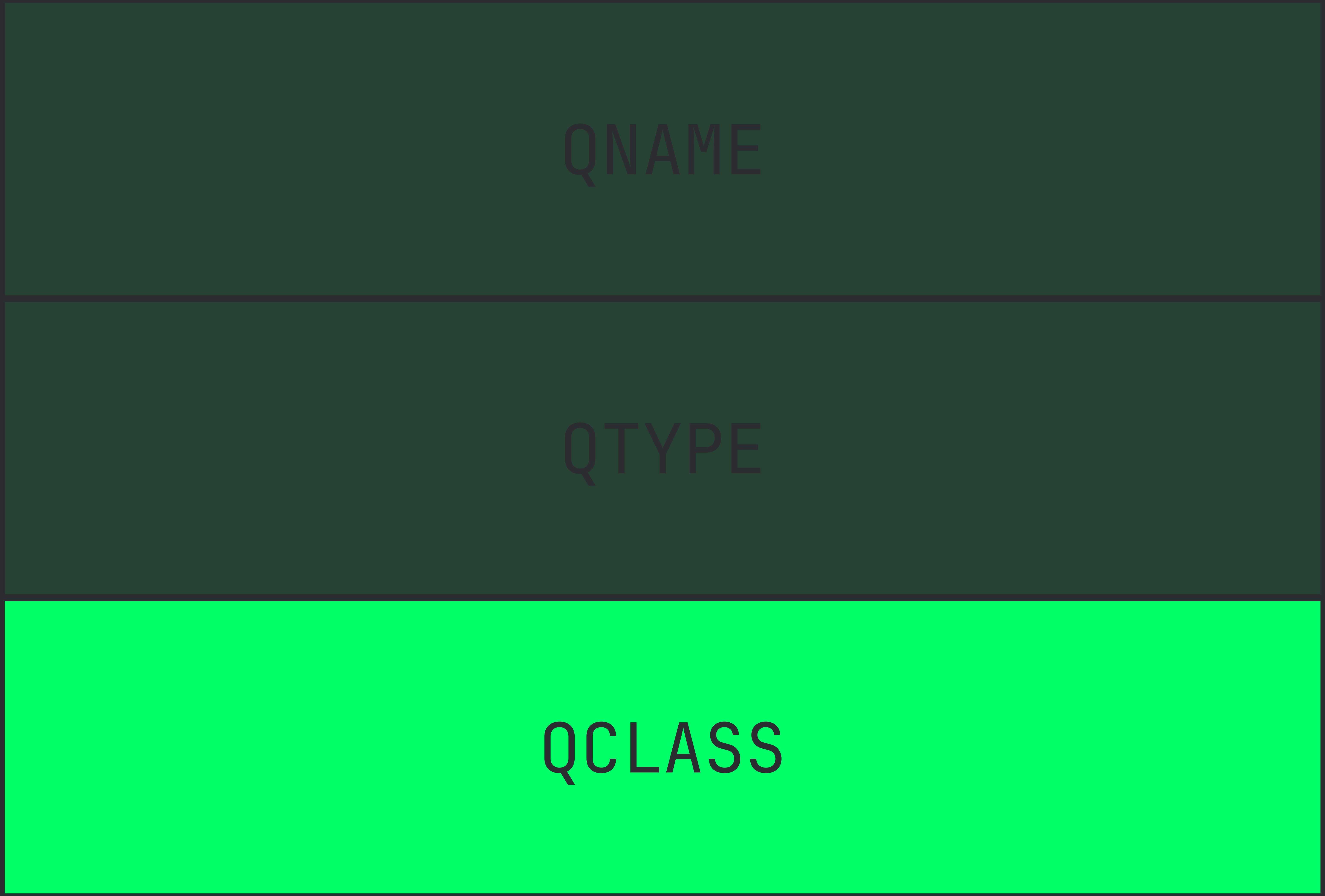
QUESTION



QNAME

QTYPE

QCLASS



QNAME

QTYPE

QCLASS

QCLASS

- → 16 bit int, 0 - 65535 options
- → it’s “always” IN(ternet) (1)
- → most middlebox ignore (test!)

DNS Sandwich

- So we have Z (4 bits) and QCLASS (16 bits)
- Not a lot of data but…
- You can manipulate since middleboxes ignore and
- Most traditional tools similiarly ignore it!
- Low bandwidth = useful for semantic signalling

Detecting DNS Sandwich

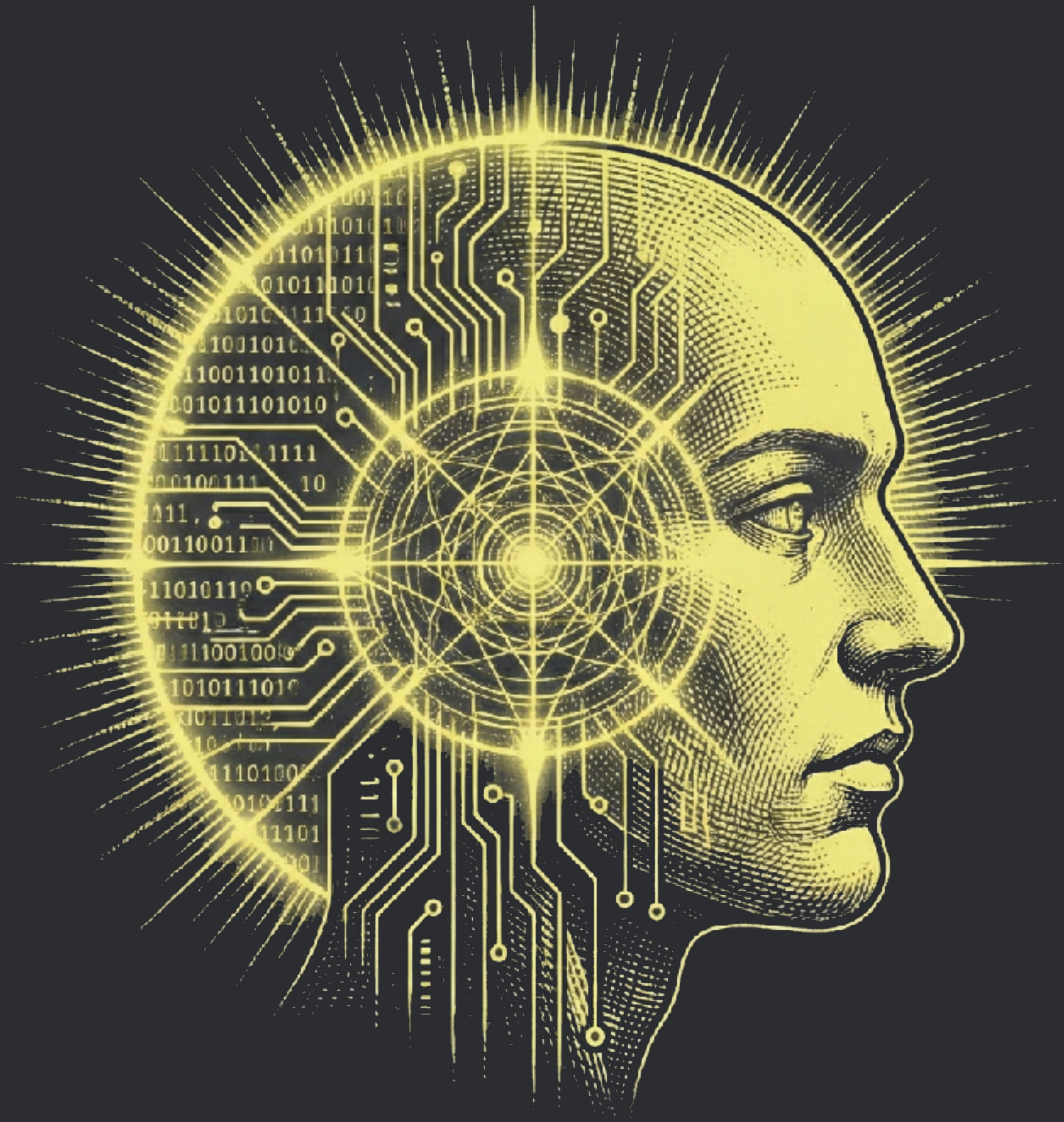
- Z should always be 0 (even with DNSSEC)
- QCLASS is 1 (99.999% of time)
- RARE: 3 (CH), 4 (HS), 254 or 255
- Zeek does not produce default events
- BUT, default parser exposes it!

```
# Z field check

if ( msg$Z ≠ 0 ) → ALERT
```

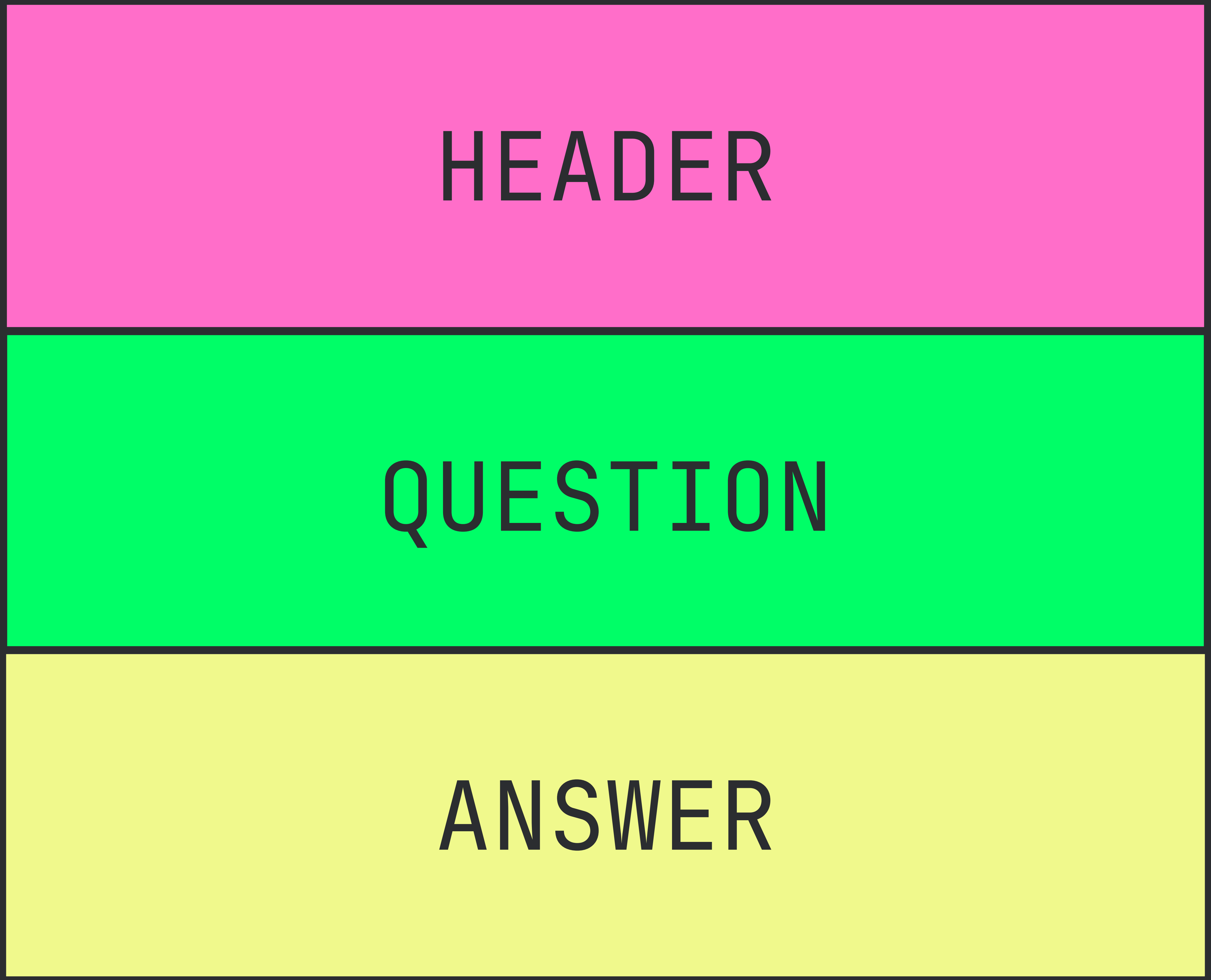
```
ALERT: Z field non-zero! 192.168.1.142 →
beacon.malware-c2.net [Z=7]
```

```
# QCLASS check

if ( qclass ≠ 1 ) → ALERT
```

```
ALERT: Unusual QCLASS 254! 192.168.1.142 →
data.exfil-domain.com [NONE]
```



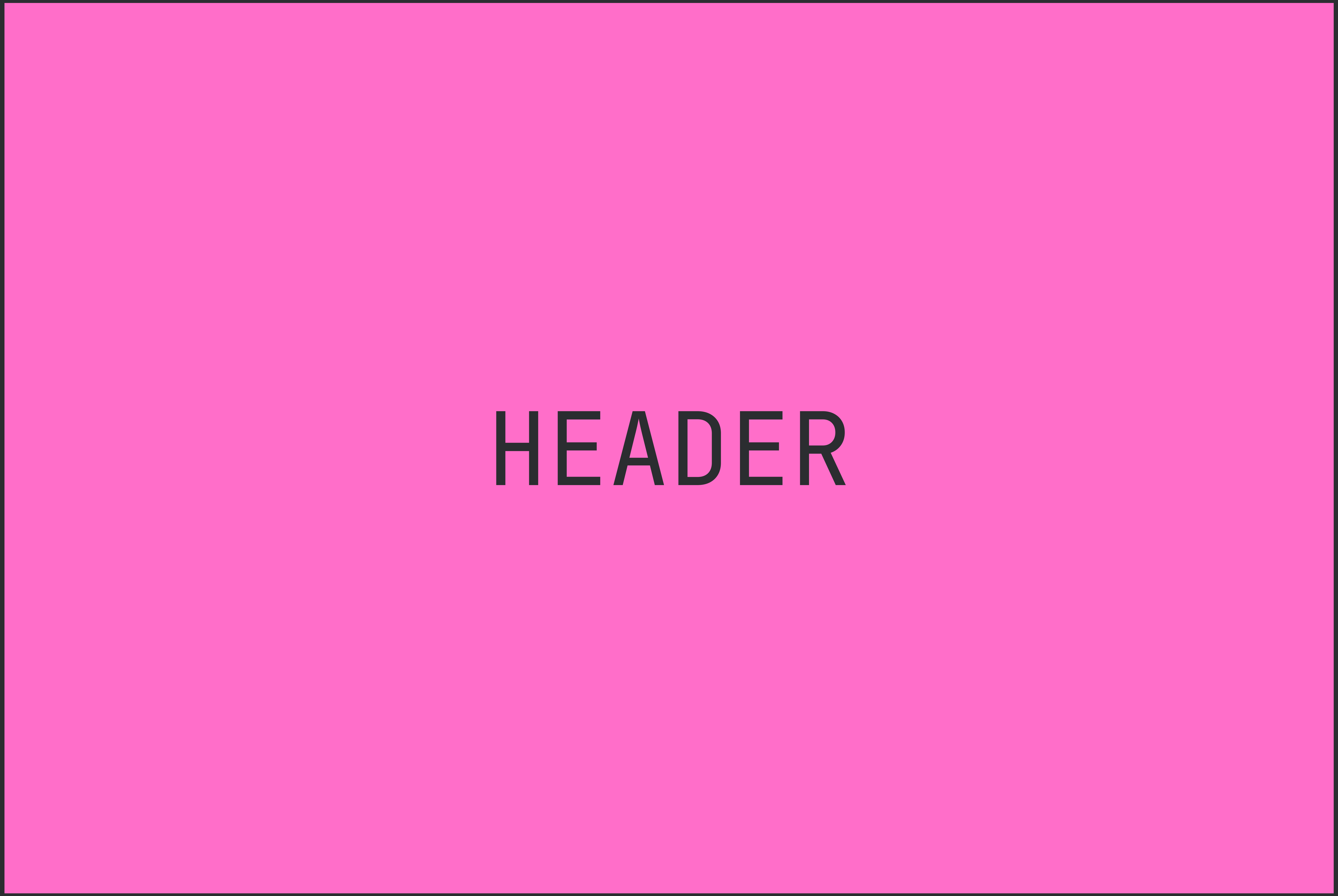
ID Field Misuse



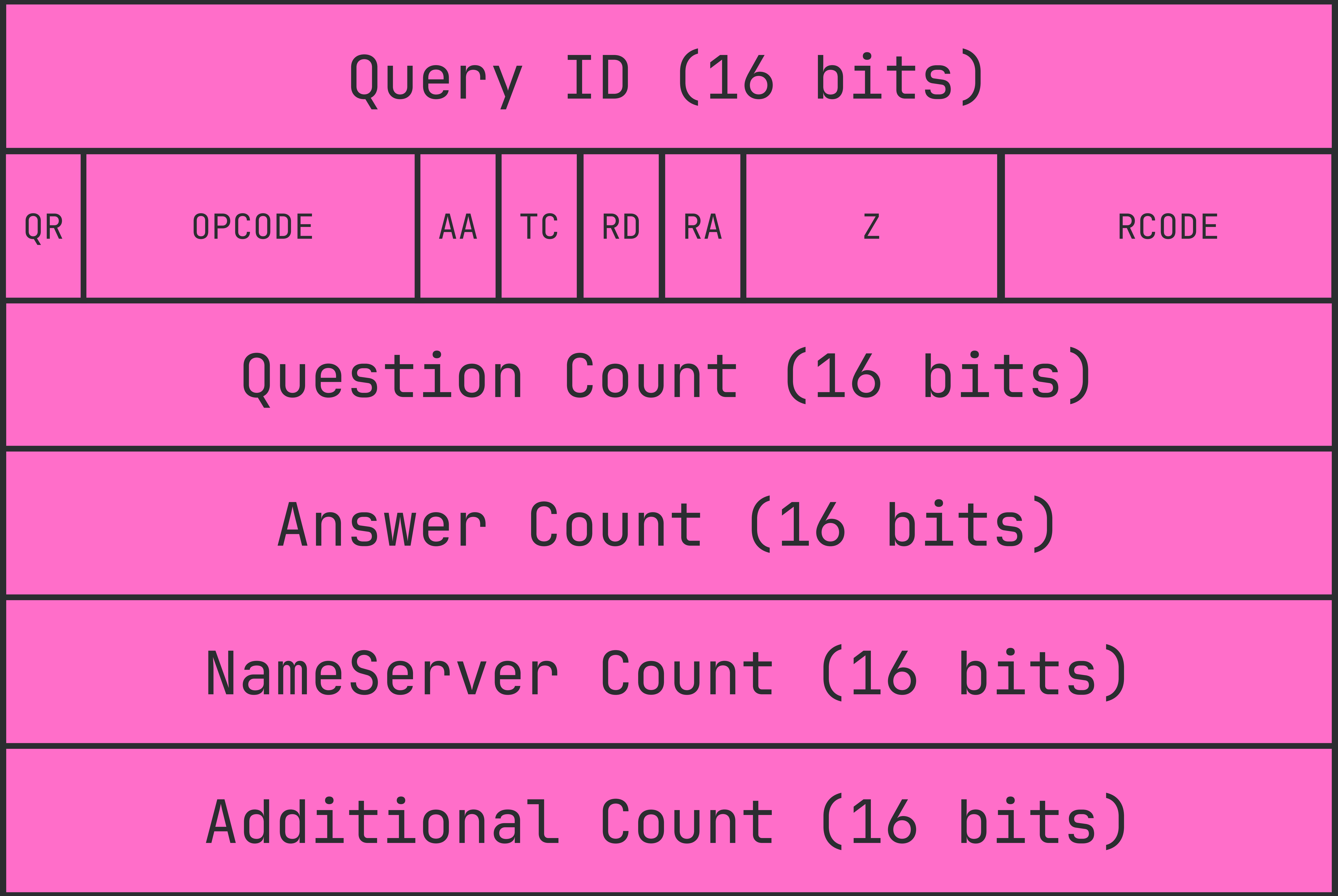
HEADER

QUESTION

ANSWER



HEADER



Query ID (16 bits)

QR | OPCODE | AA | TC | RD | RA | Z | RCODE

Question Count (16 bits)

Answer Count (16 bits)

NameServer Count (16 bits)

Additional Count (16 bits)

Query ID (16 bits)

Query ID (16 bits)

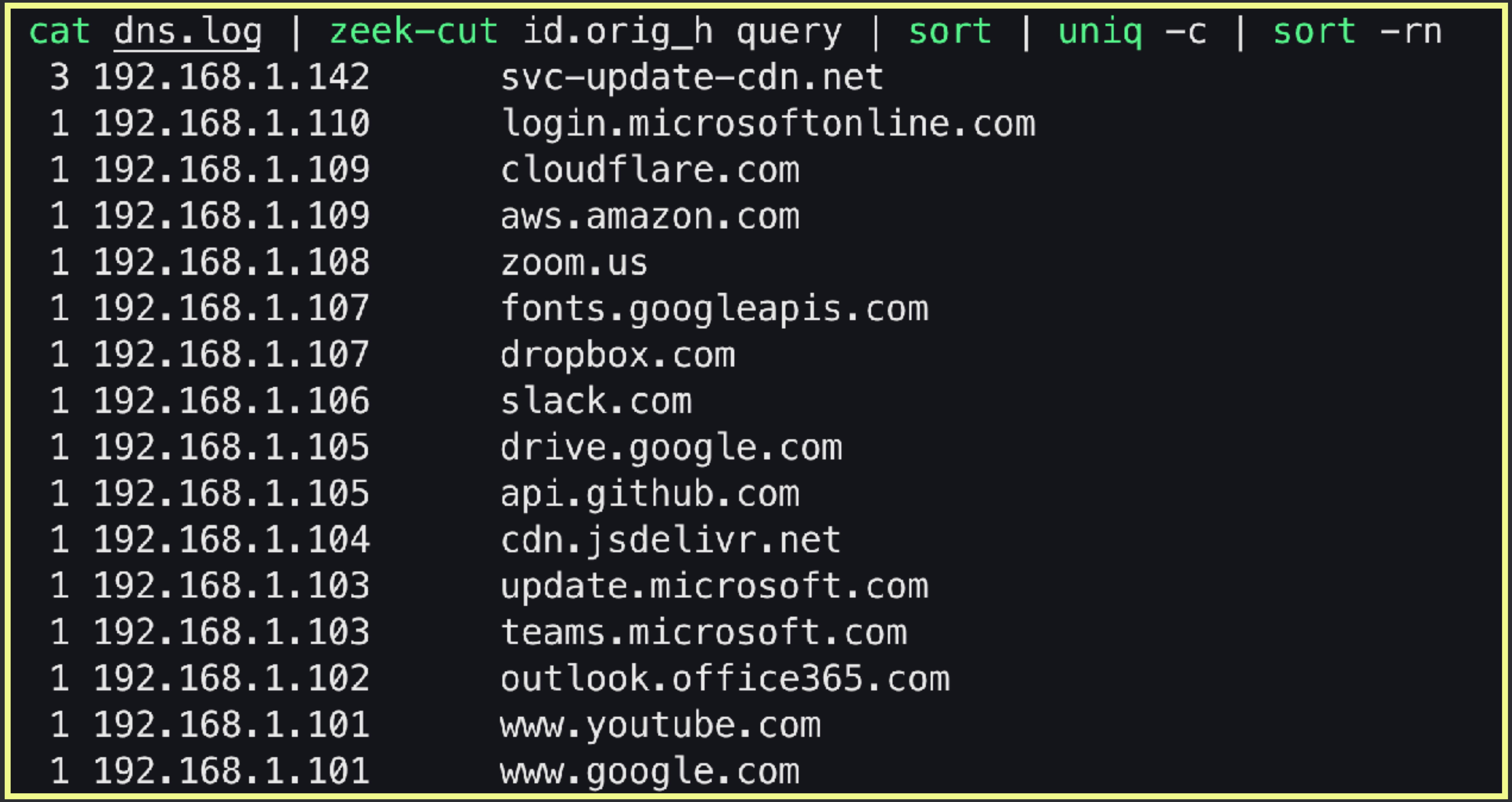
- randomly generated by client
- Allows query ⟷ response matching
- Mostly for Agent → Server (Server has to echo)
- Also very limited, def not bulk (2 bytes)

So, what does it look like when
its normal, vs when it’s malicious?

Well, it depends…

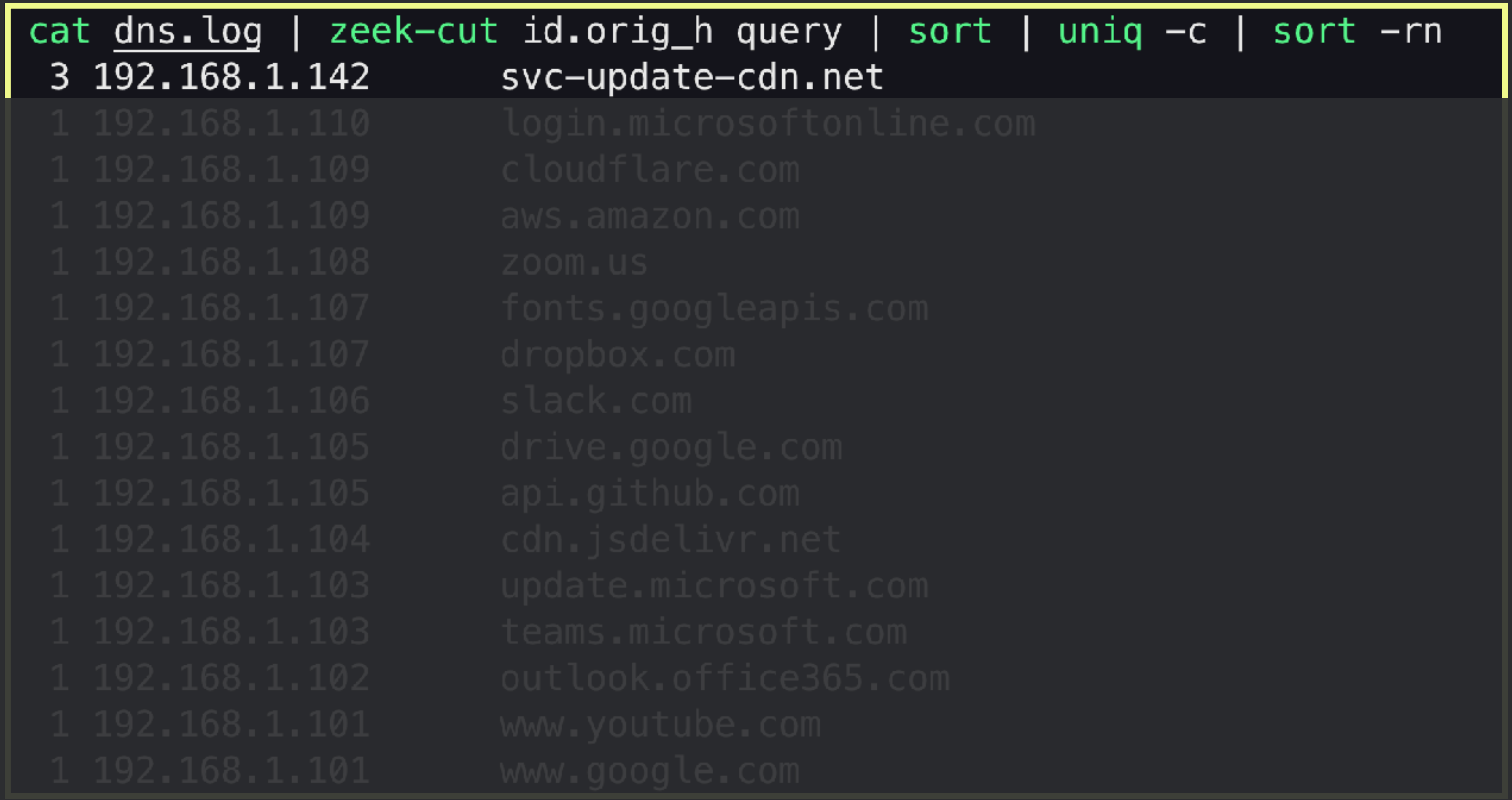
Let’s simulate “a hunt”

```
cat dns.log | zeek-cut id.orig_h query |
sort | uniq -c | sort -rn
```



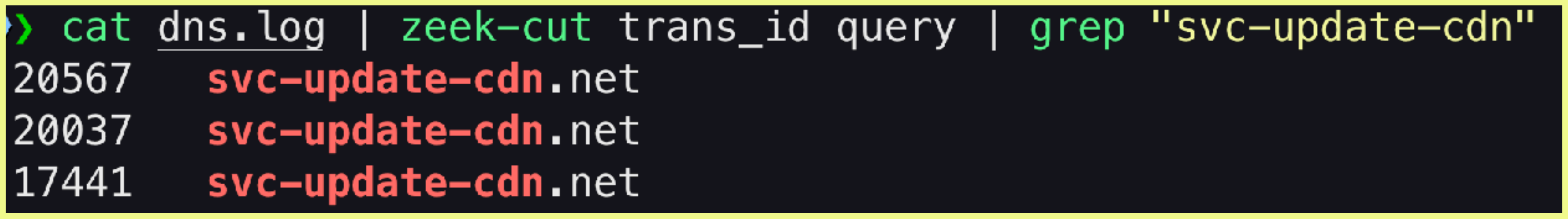
```
cat dns.log | zeek-cut id.orig_h query | sort | uniq -c | sort -rn
 3 192.168.1.142      svc-update-cdn.net
 1 192.168.1.110      login.microsoftonline.com
 1 192.168.1.109      cloudflare.com
 1 192.168.1.109      aws.amazon.com
 1 192.168.1.108      zoom.us
 1 192.168.1.107      fonts.googleapis.com
 1 192.168.1.107      dropbox.com
 1 192.168.1.106      slack.com
 1 192.168.1.105      drive.google.com
 1 192.168.1.105      api.github.com
 1 192.168.1.104      cdn.jsdelivr.net
 1 192.168.1.103      update.microsoft.com
 1 192.168.1.103      teams.microsoft.com
 1 192.168.1.102      outlook.office365.com
 1 192.168.1.101      www.youtube.com
 1 192.168.1.101      www.google.com
```

```
cat dns.log | zeek-cut id.orig_h query |
sort | uniq -c | sort -rn
```



```
cat dns.log | zeek-cut id.orig_h query | sort | uniq -c | sort -rn
 3 192.168.1.142      svc-update-cdn.net
 1 192.168.1.110      login.microsoftonline.com
 1 192.168.1.109      cloudflare.com
 1 192.168.1.109      aws.amazon.com
 1 192.168.1.108      zoom.us
 1 192.168.1.107      fonts.googleapis.com
 1 192.168.1.107      dropbox.com
 1 192.168.1.106      slack.com
 1 192.168.1.105      drive.google.com
 1 192.168.1.105      api.github.com
 1 192.168.1.104      cdn.jsdelivr.net
 1 192.168.1.103      update.microsoft.com
 1 192.168.1.103      teams.microsoft.com
 1 192.168.1.102      outlook.office365.com
 1 192.168.1.101      www.youtube.com
 1 192.168.1.101      www.google.com
```

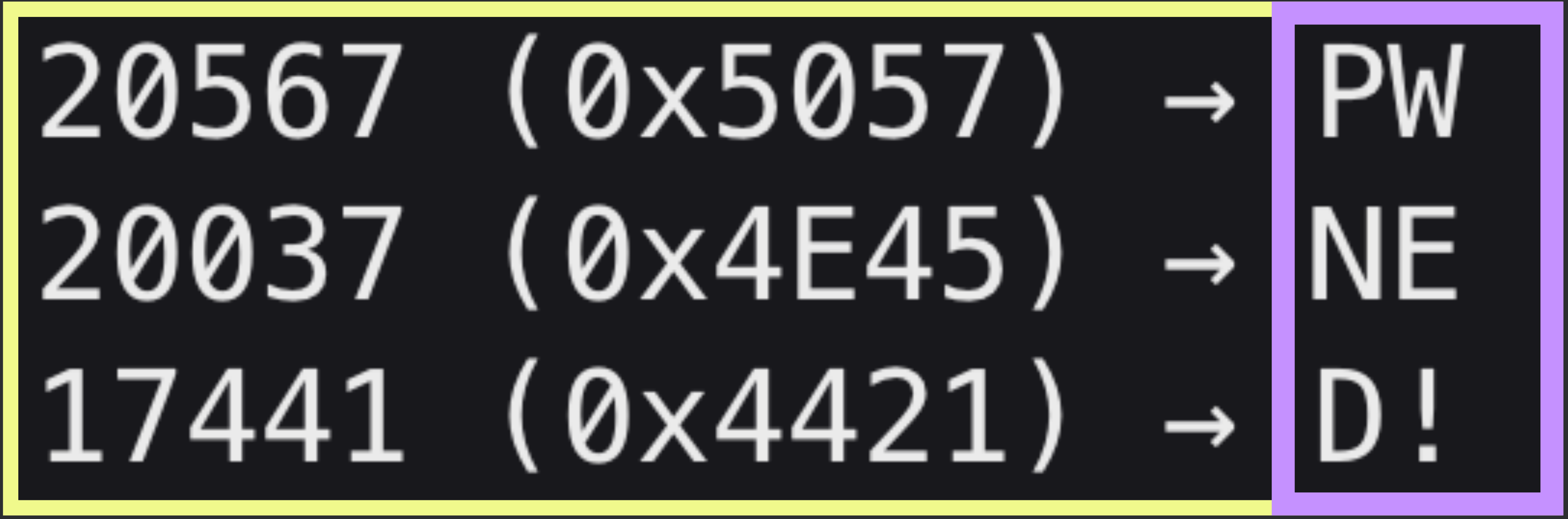
```
cat dns.log | zeek-cut trans_id query |
grep "svc-update-cdn"
```



```
❯ cat dns.log | zeek-cut trans_id query | grep "svc-update-cdn"
20567   svc-update-cdn.net
20037   svc-update-cdn.net
17441   svc-update-cdn.net
```

Zeek logs trans_id as decimal, not hex

```
cat dns.log | zeek-cut trans_id query |
grep "svc-update-cdn" | awk '{printf "%5d (0x%04X) →
%c%c\n", $1, $1, int($1/256), $1%256}'
```



```
20567 (0x5057) → PW
20037 (0x4E45) → NE
17441 (0x4421) → D!
```

PWNED!… Not so “random” looking, eh?

So, if we suspect ID Field abuse,

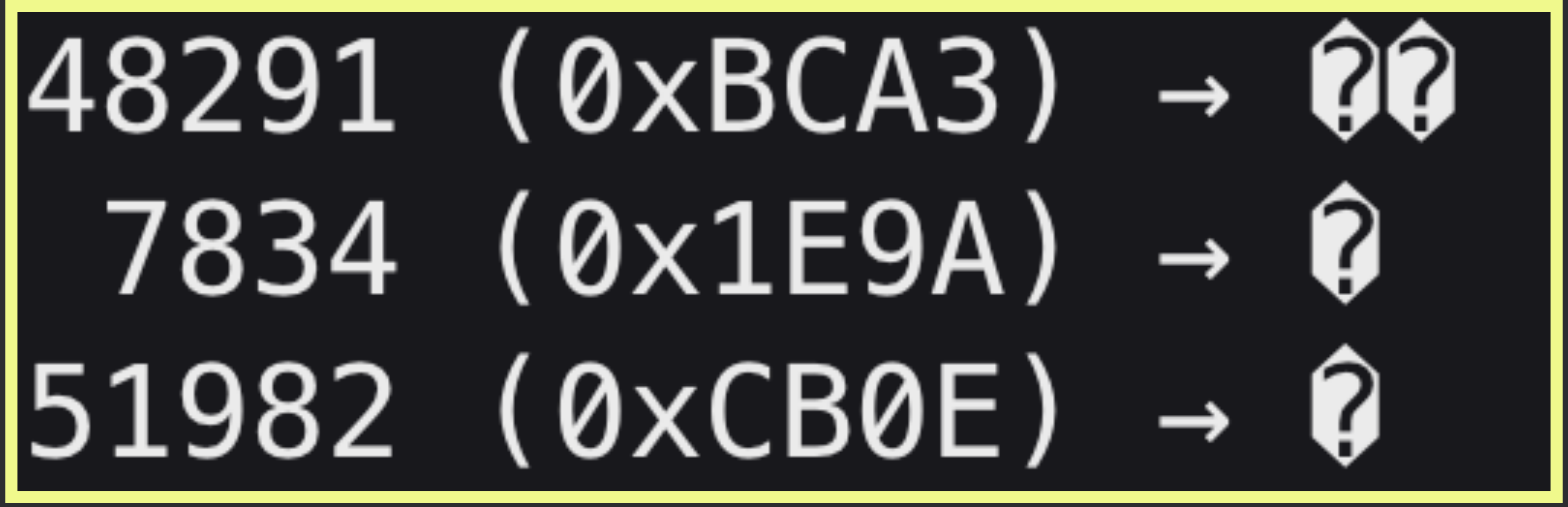
we can decode and inspect

BUT… We were lucky here

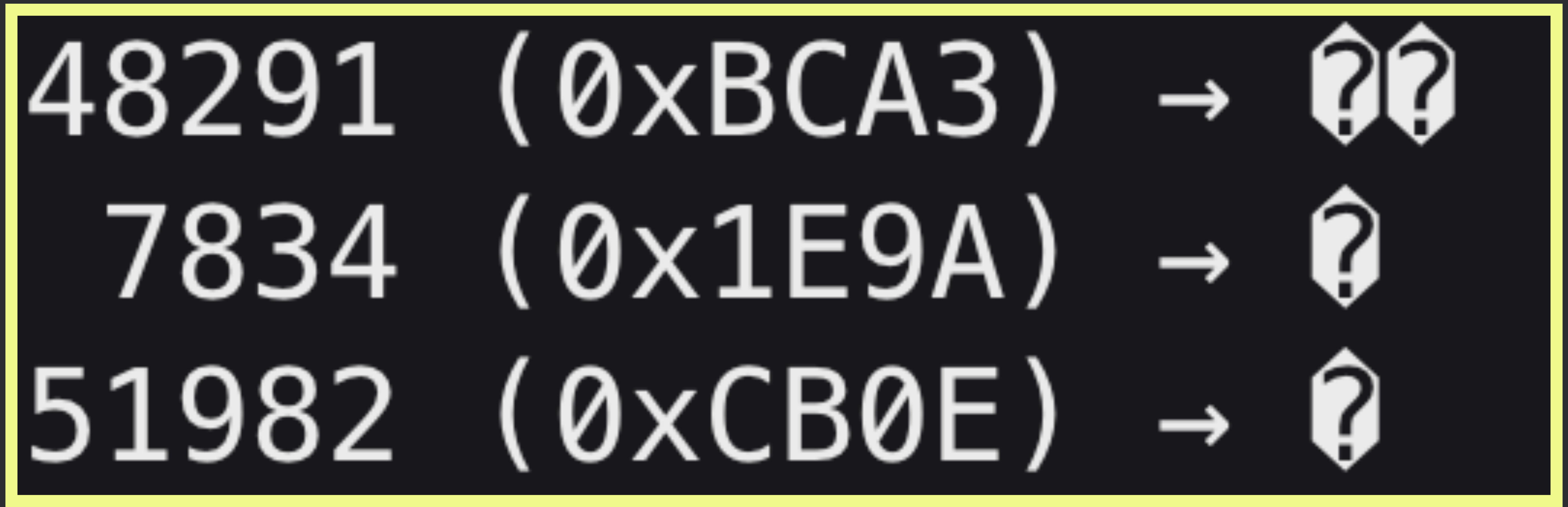
Why? Adversary “forgot” to encrypt data before encoding

If they didn’t…

```
cat dns.log | zeek-cut trans_id query |
grep "svc-update-cdn" | awk '{printf "%5d (0x%04X) →
%c%c\n", $1, $1, int($1/256), $1%256}'
```



```
48291 (0xBCA3) → ��
 7834 (0x1E9A) → �
51982 (0xCB0E) → �
```



```
48291 (0xBCA3) → ��
 7834 (0x1E9A) → �
51982 (0xCB0E) → �
```

Non-printable bytes

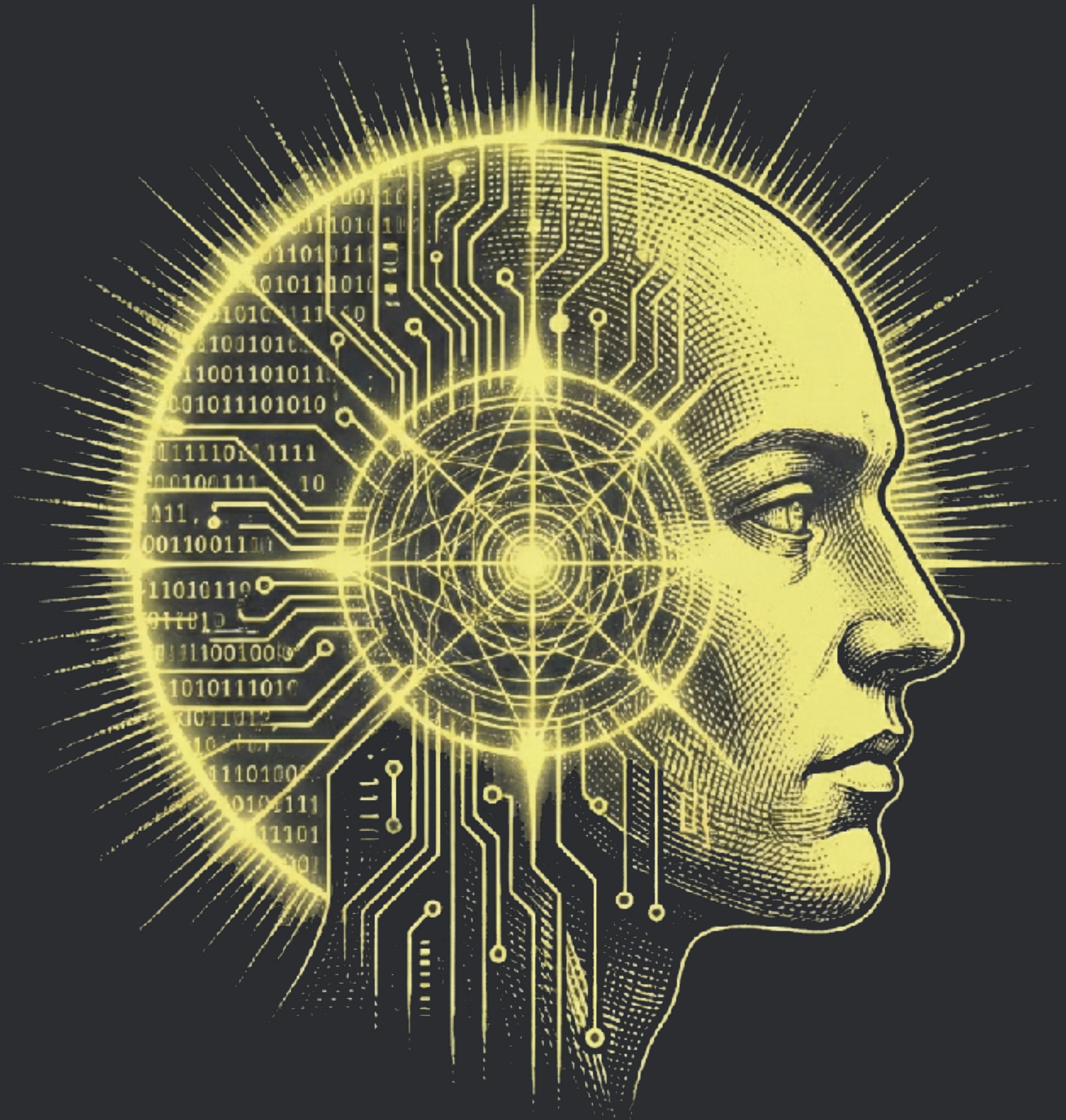
> Are they encrypted, or random?

> No way to tell

This means that if an adversary is using Field ID for exfil and is encrypting prior to encoding, there is no real way to detect it, at least not directly…

Behavioural Detections

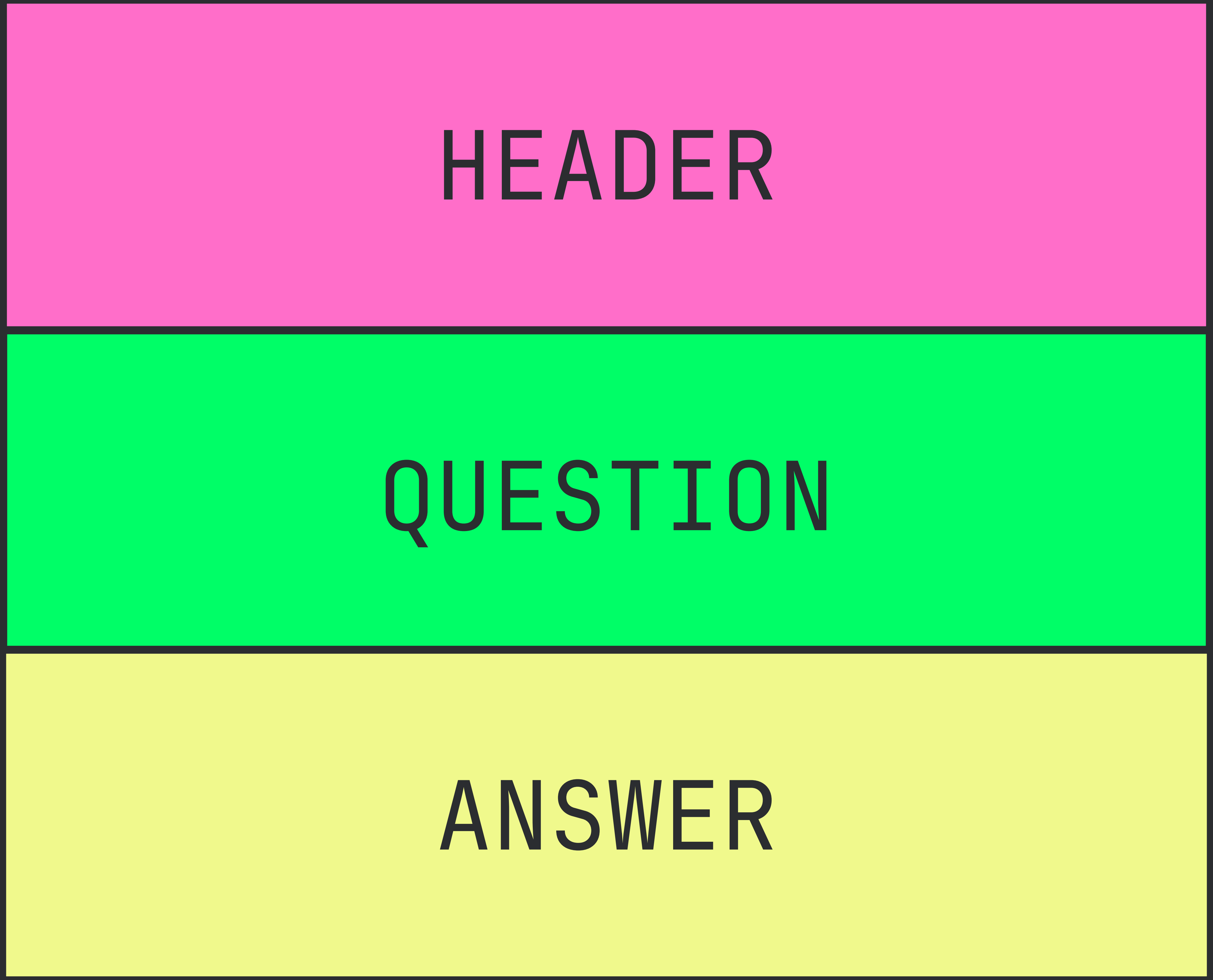
- Domain reputation/age - New? Known?
- Query frequency (ID Field LOW capacity)
- Timing patterns (DNS can still beacon)
- Resolver bypass… (The “Caching Conundrum”)
- No corresponding traffic (!!!)



EDNS0

Extension Mechanism for DNS

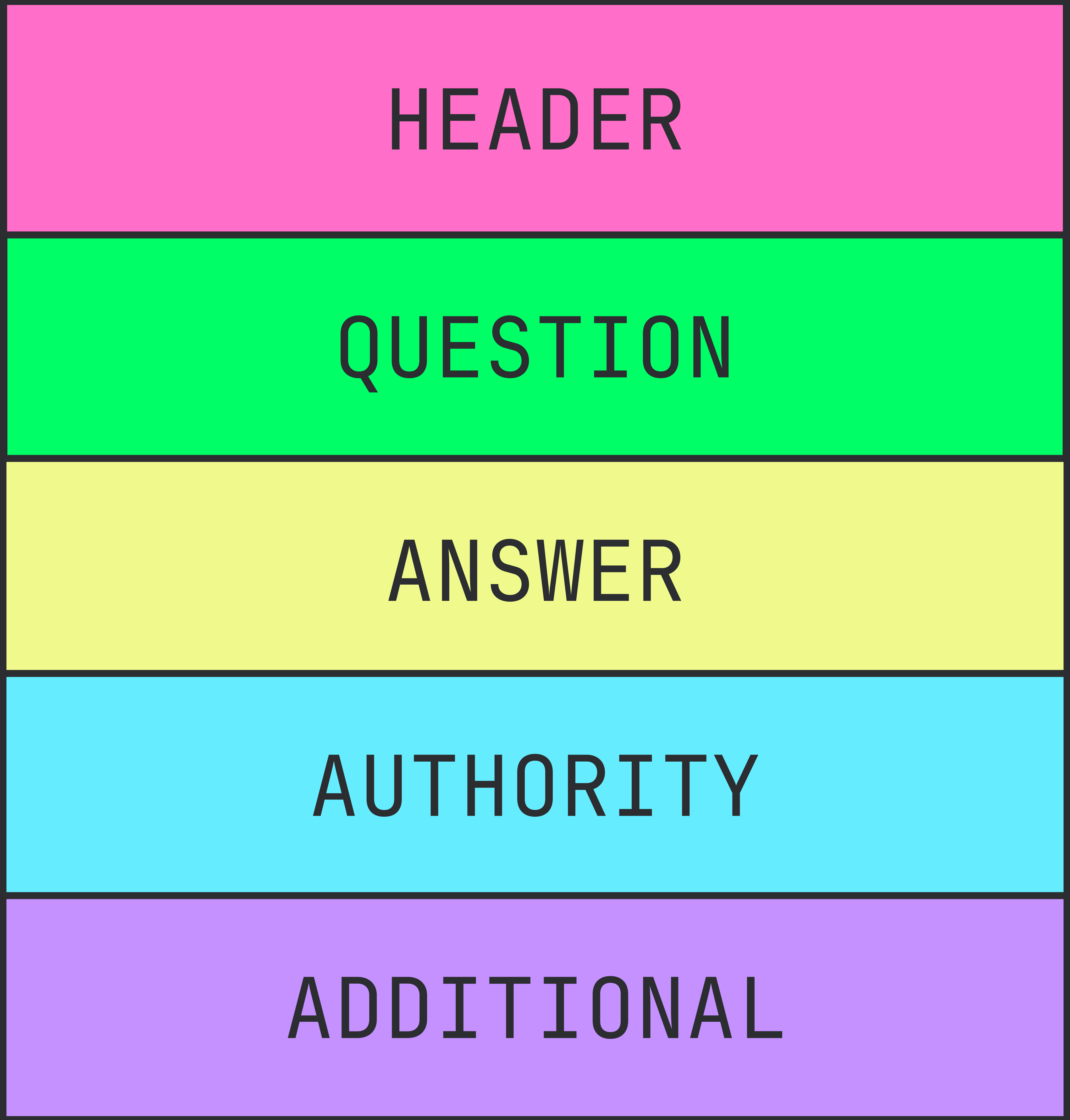
- 1987 - original DNS protocol limiting
- 1999 - new functionality required (larger, DNSSEC later)
- Cannot redesign, introduce backward-compatible hack
- Repurpose resource record and place in Additional
- Creates extensible FW that is pliable for new use cases



HEADER

QUESTION

ANSWER



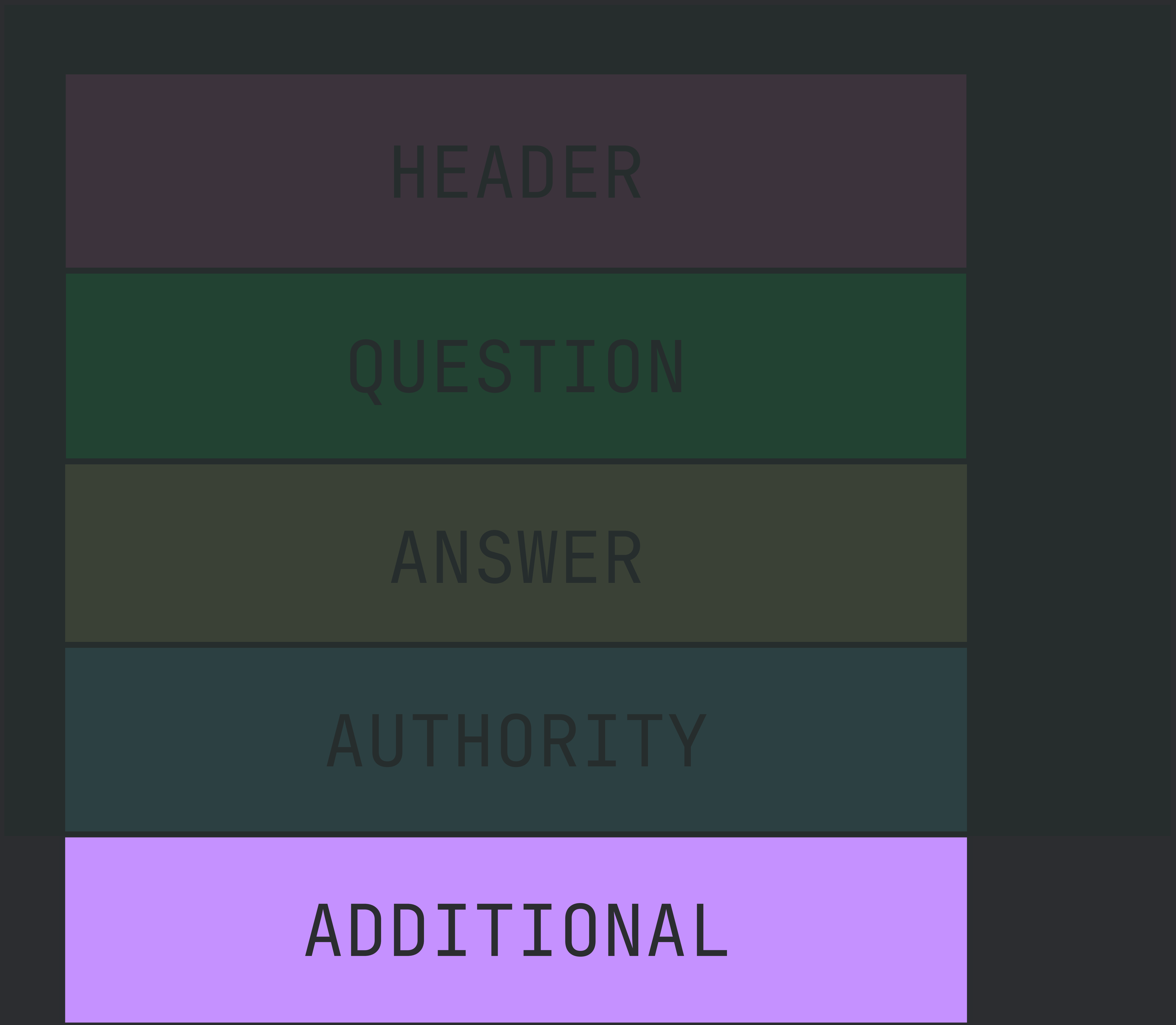
HEADER

QUESTION

ANSWER

AUTHORITY

ADDITIONAL



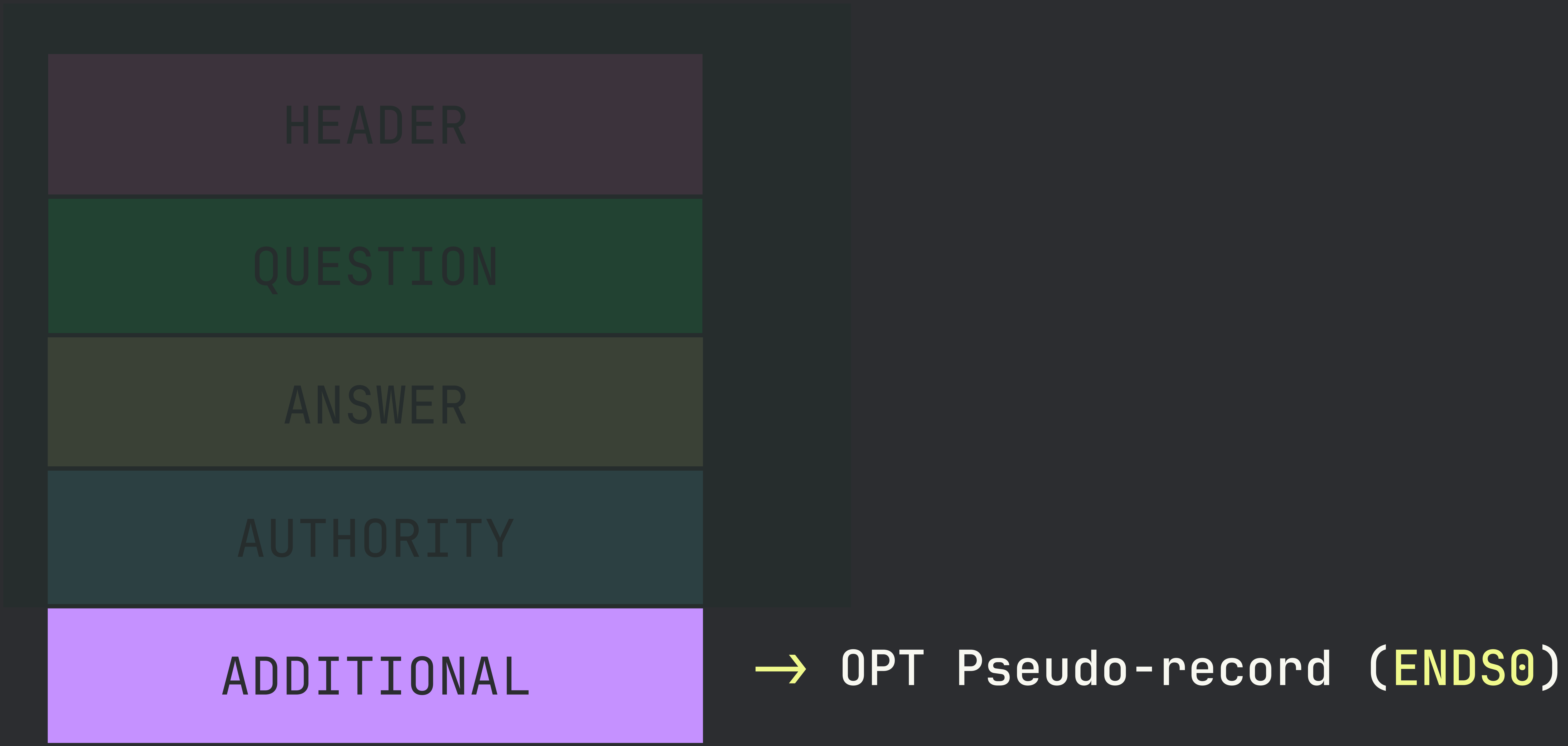
HEADER

QUESTION

ANSWER

AUTHORITY

ADDITIONAL



HEADER

QUESTION

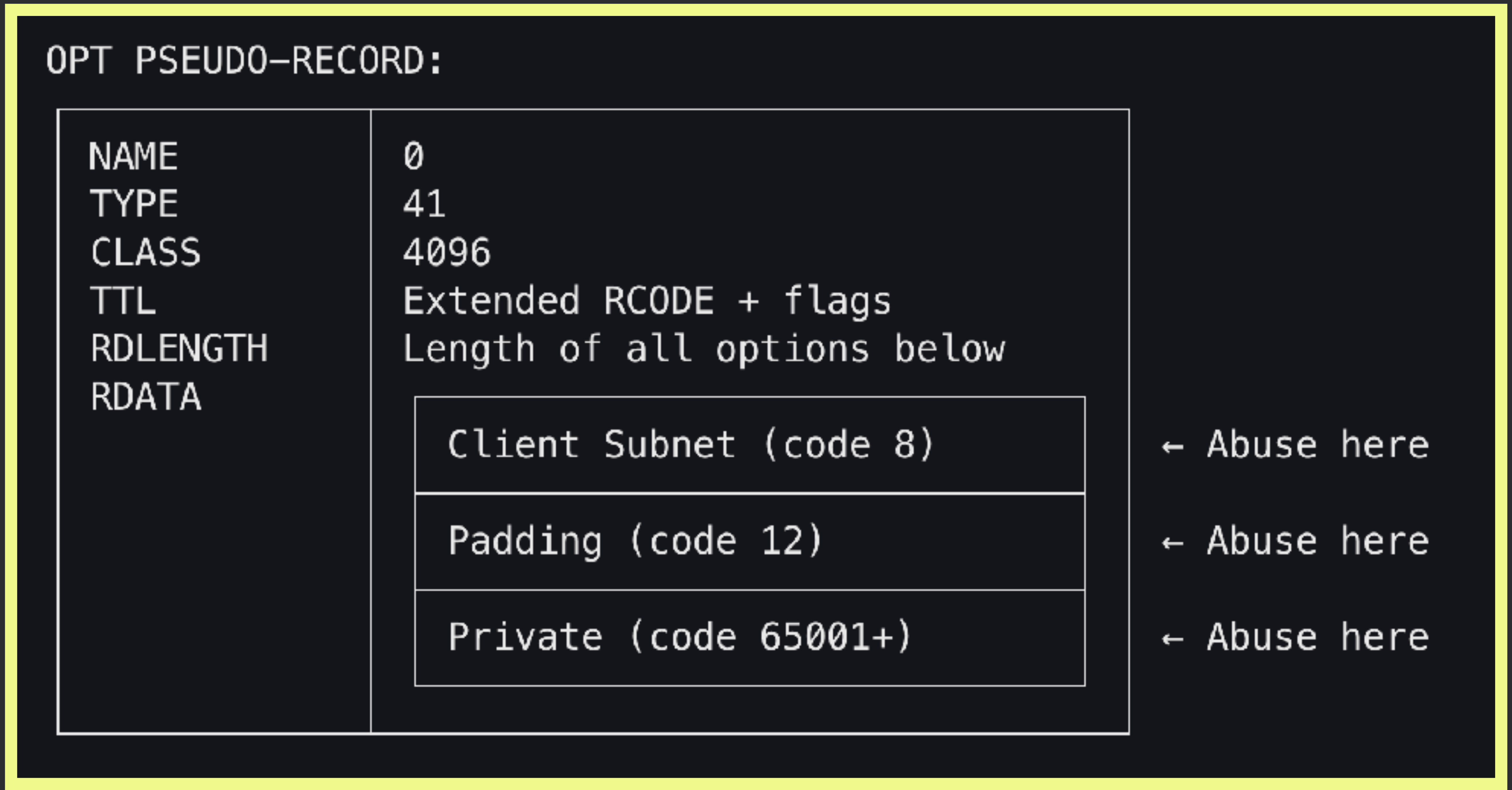
ANSWER

AUTHORITY

ADDITIONAL → OPT Pseudo-record (ENDS0)

Why Adversaries Love It

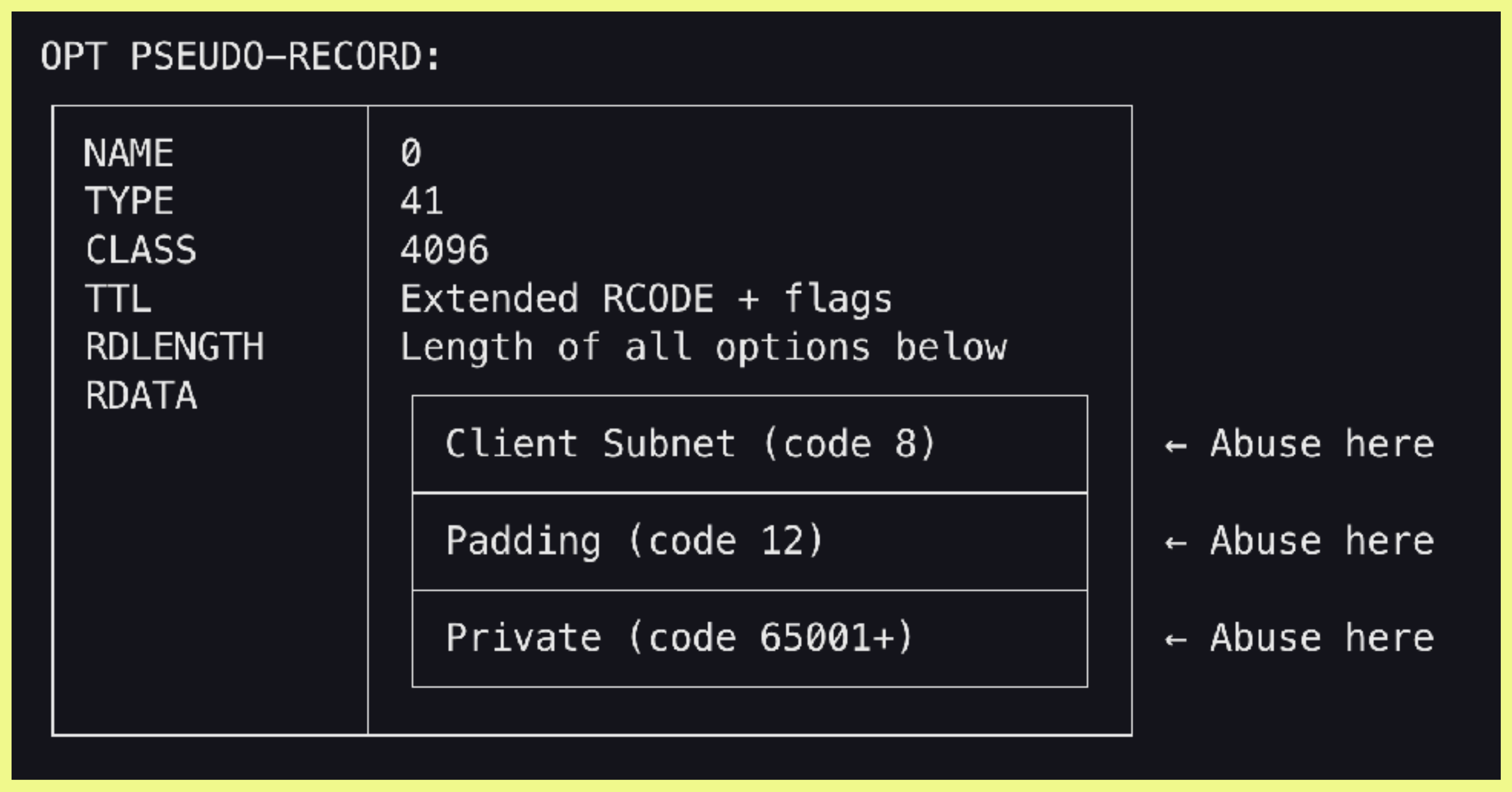
- With EDNS0, Client says: I can handle 4096 bytes
- Server can then send a packet up to 4096 bytes
- Gives 3 extra fields (comb up to 4096 bytes)
- Very often ignored!



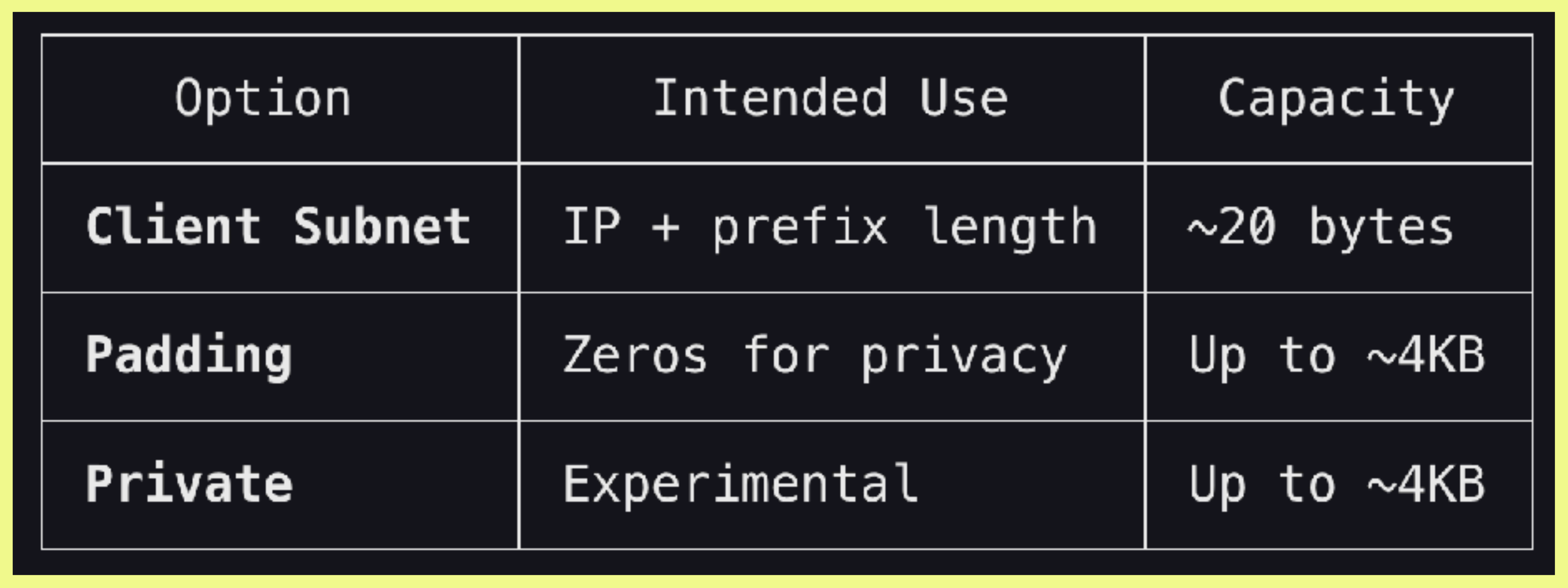
OPT PSEUDO-RECORD:

NAME	0
TYPE	41
CLASS	4096
TTL	Extended RCODE + flags
RDLENGTH	Length of all options below
RDATA	Client Subnet (code 8) ← Abuse here Padding (code 12) ← Abuse here Private (code 65001+) ← Abuse here

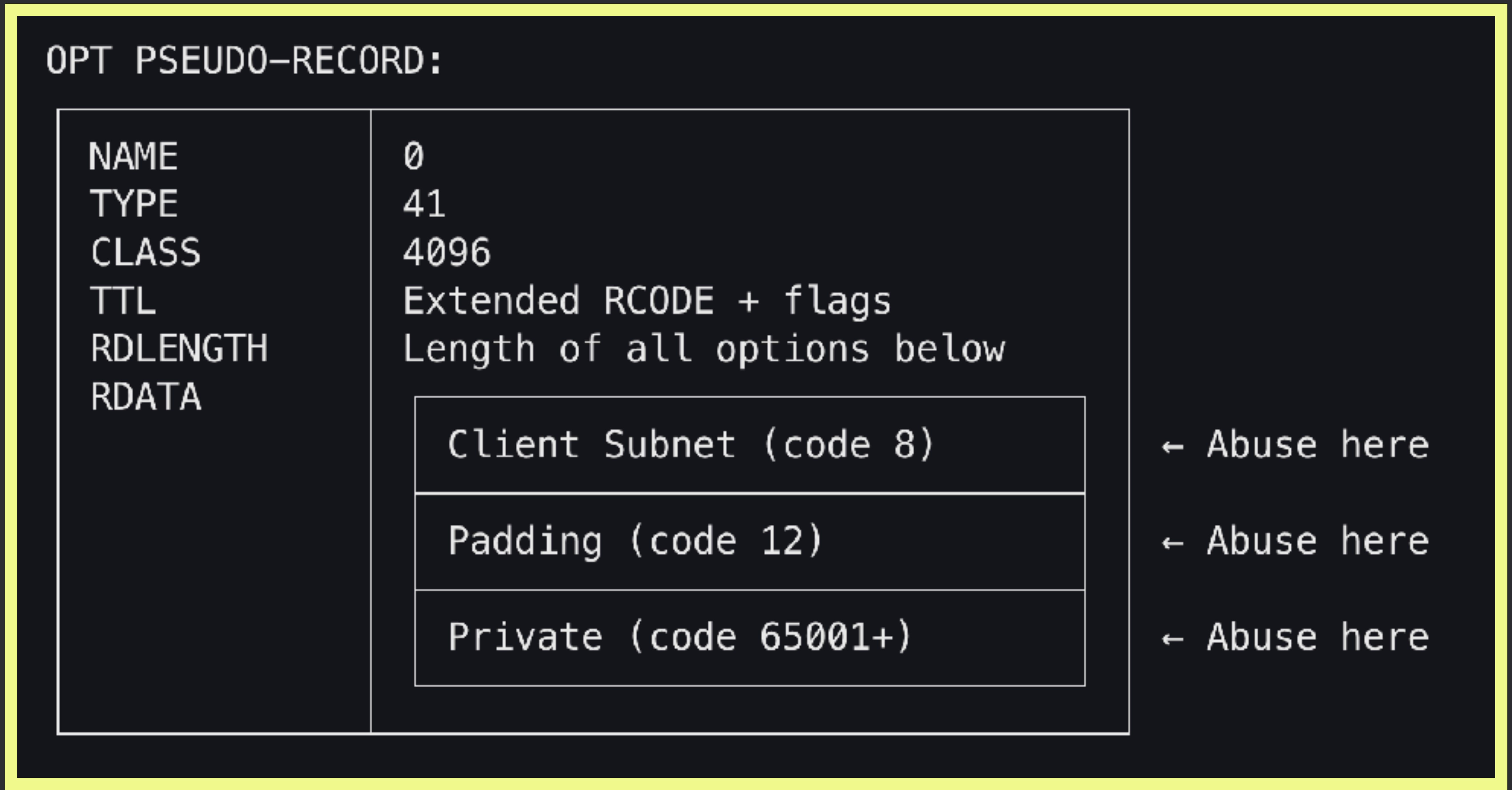
Option	Intended Use	Capacity
Client Subnet	IP + prefix length	~20 bytes
Padding	Zeros for privacy	Up to ~4KB
Private	Experimental	Up to ~4KB



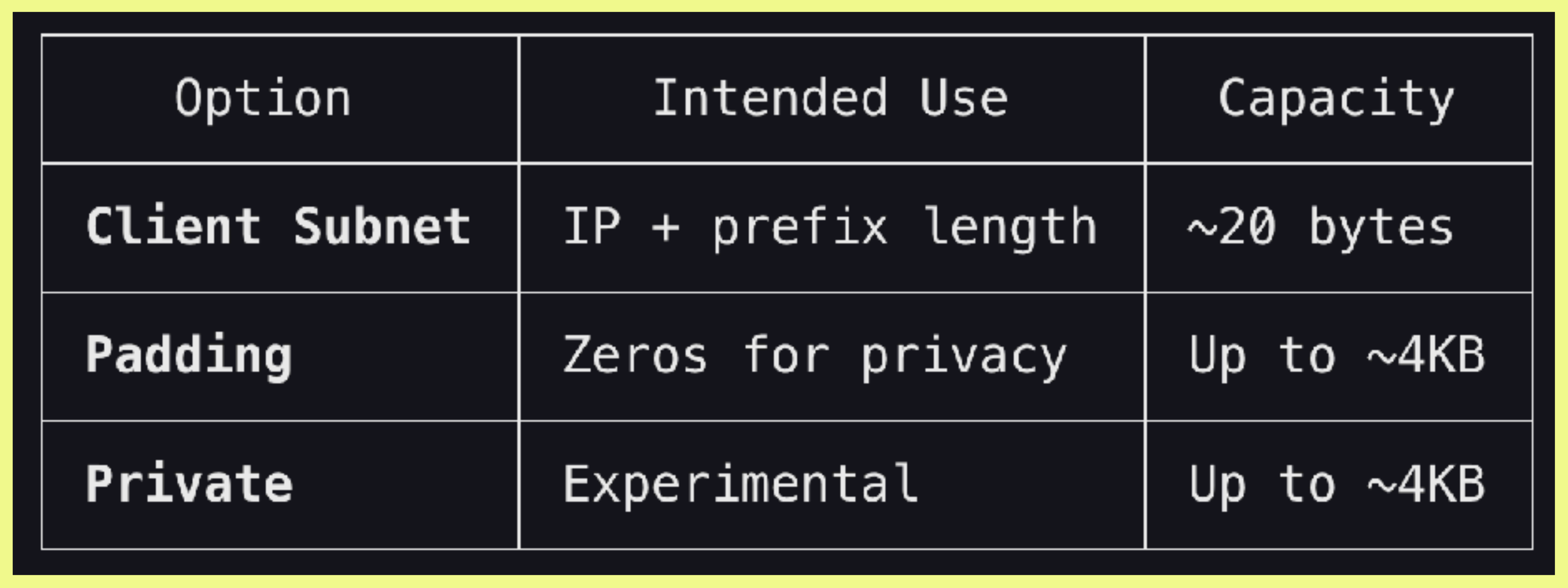
```
OPT PSEUDO-RECORD:

NAME        0
TYPE        41
CLASS       4096
TTL         Extended RCODE + flags
RDLENGTH    Length of all options below
RDATA
            Client Subnet (code 8)        ← Abuse here
            Padding (code 12)             ← Abuse here
            Private (code 65001+)         ← Abuse here
```



Option	Intended Use	Capacity
Client Subnet	IP + prefix length	~20 bytes
Padding	Zeros for privacy	Up to ~4KB
Private	Experimental	Up to ~4KB



```
OPT PSEUDO-RECORD:
┌─────────────┬──────────────────────────────────┐
│ NAME        │ 0                                │
│ TYPE        │ 41                               │
│ CLASS       │ 4096                             │
│ TTL         │ Extended RCODE + flags           │
│ RDLENGTH    │ Length of all options below      │
│ RDATA       │ ┌──────────────────────────────┐ │
│             │ │ Client Subnet (code 8)       │ │  ← Abuse here
│             │ ├──────────────────────────────┤ │
│             │ │ Padding (code 12)            │ │  ← Abuse here
│             │ ├──────────────────────────────┤ │
│             │ │ Private (code 65001+)        │ │  ← Abuse here
│             │ └──────────────────────────────┘ │
└─────────────┴──────────────────────────────────┘
```

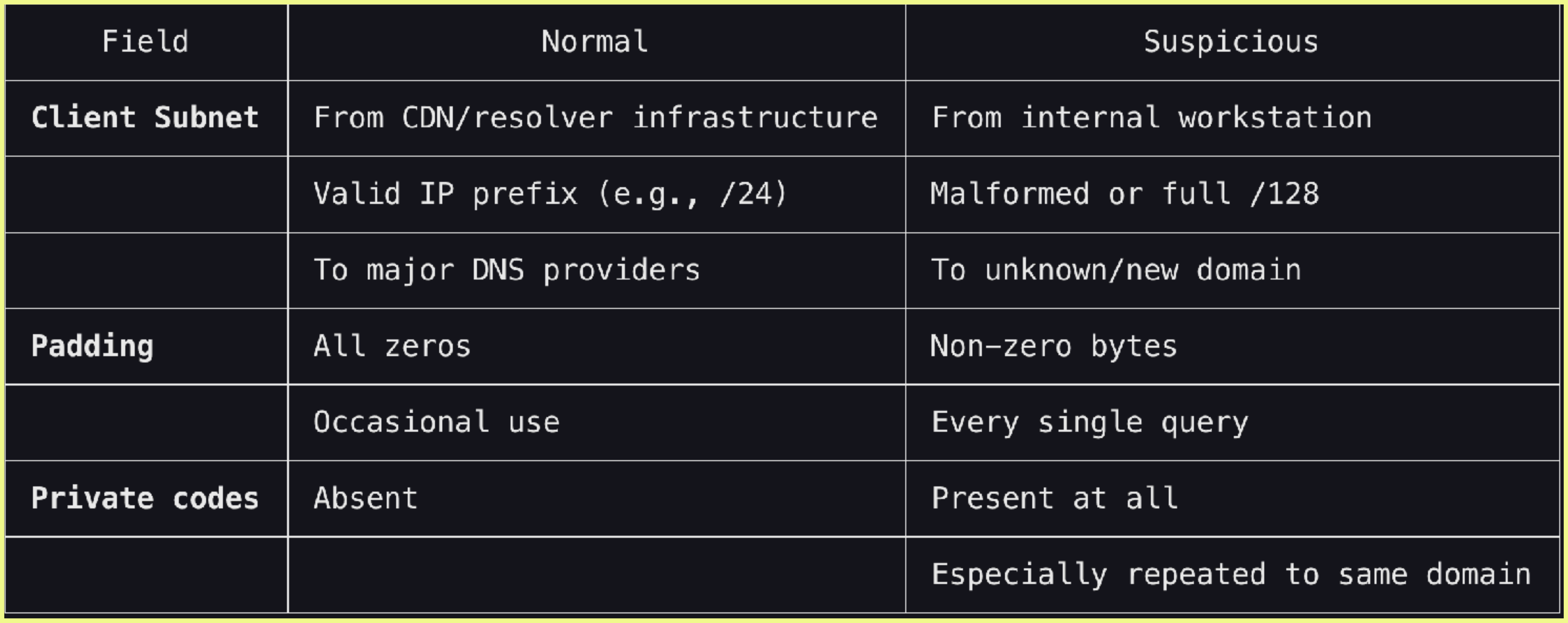


Option	Intended Use	Capacity
Client Subnet	IP + prefix length	~20 bytes
Padding	Zeros for privacy	Up to ~4KB
Private	Experimental	Up to ~4KB

Good news:

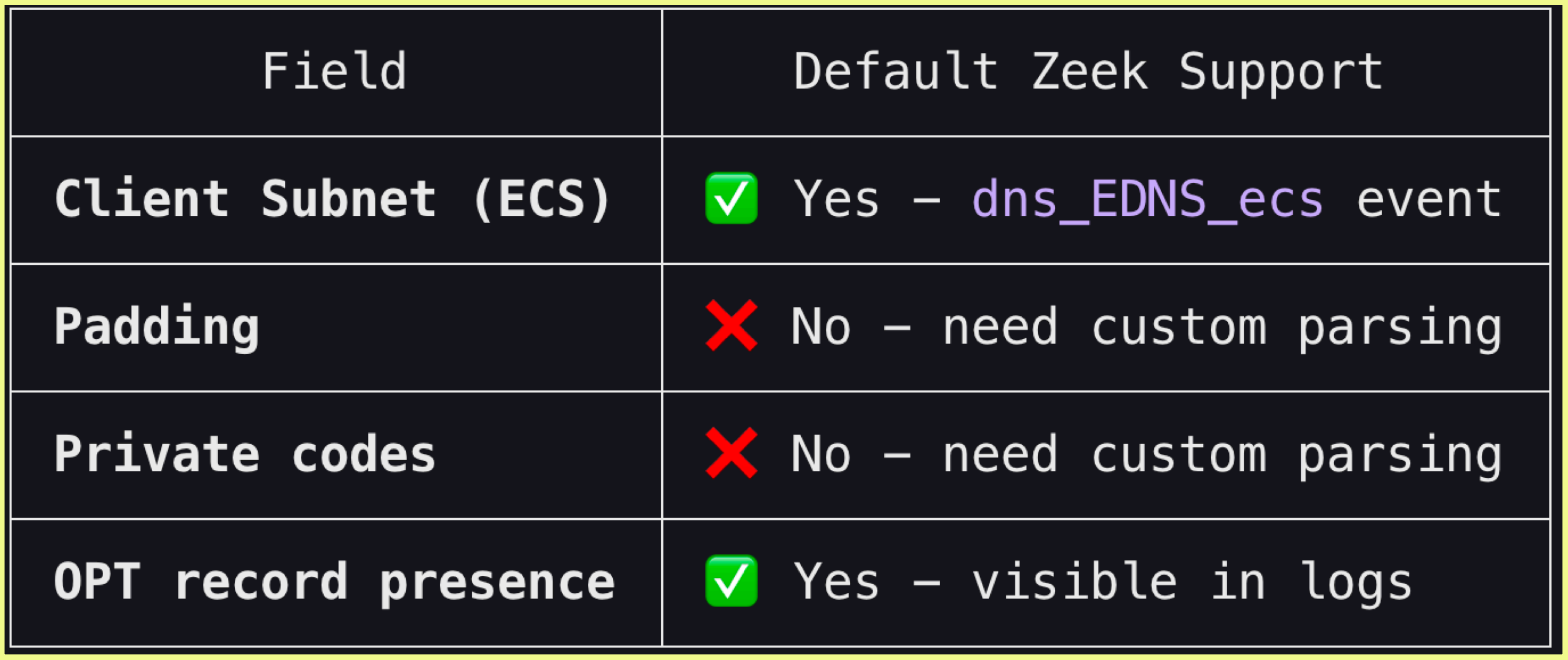
→ EDNS0 is common (no blocking)

→ misuse of 3 fields easy to spot

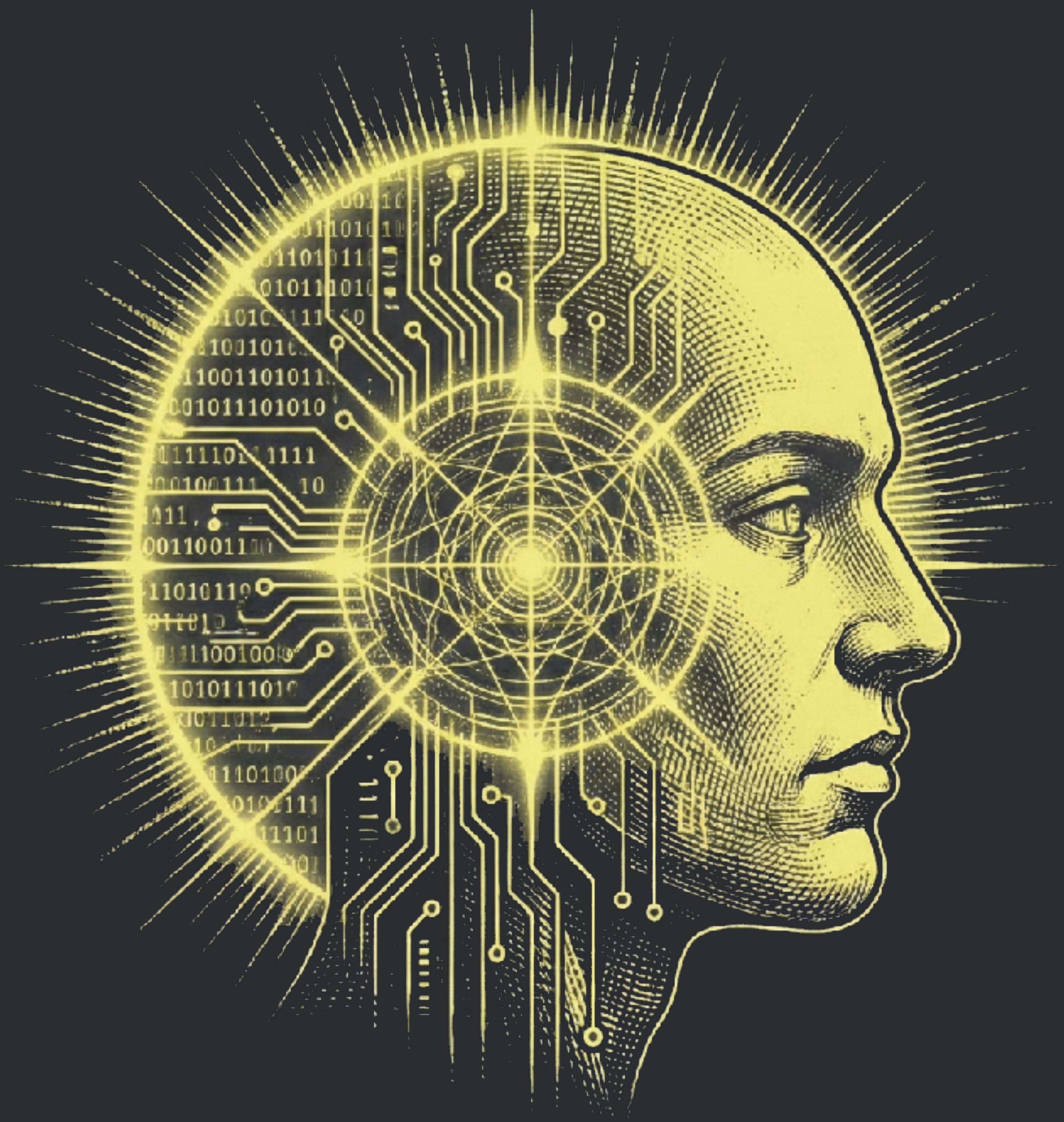


Field	Normal	Suspicious
Client Subnet	From CDN/resolver infrastructure	From internal workstation
	Valid IP prefix (e.g., /24)	Malformed or full /128
	To major DNS providers	To unknown/new domain
Padding	All zeros	Non-zero bytes
	Occasional use	Every single query
Private codes	Absent	Present at all
		Especially repeated to same domain

Bad news… need custom parser

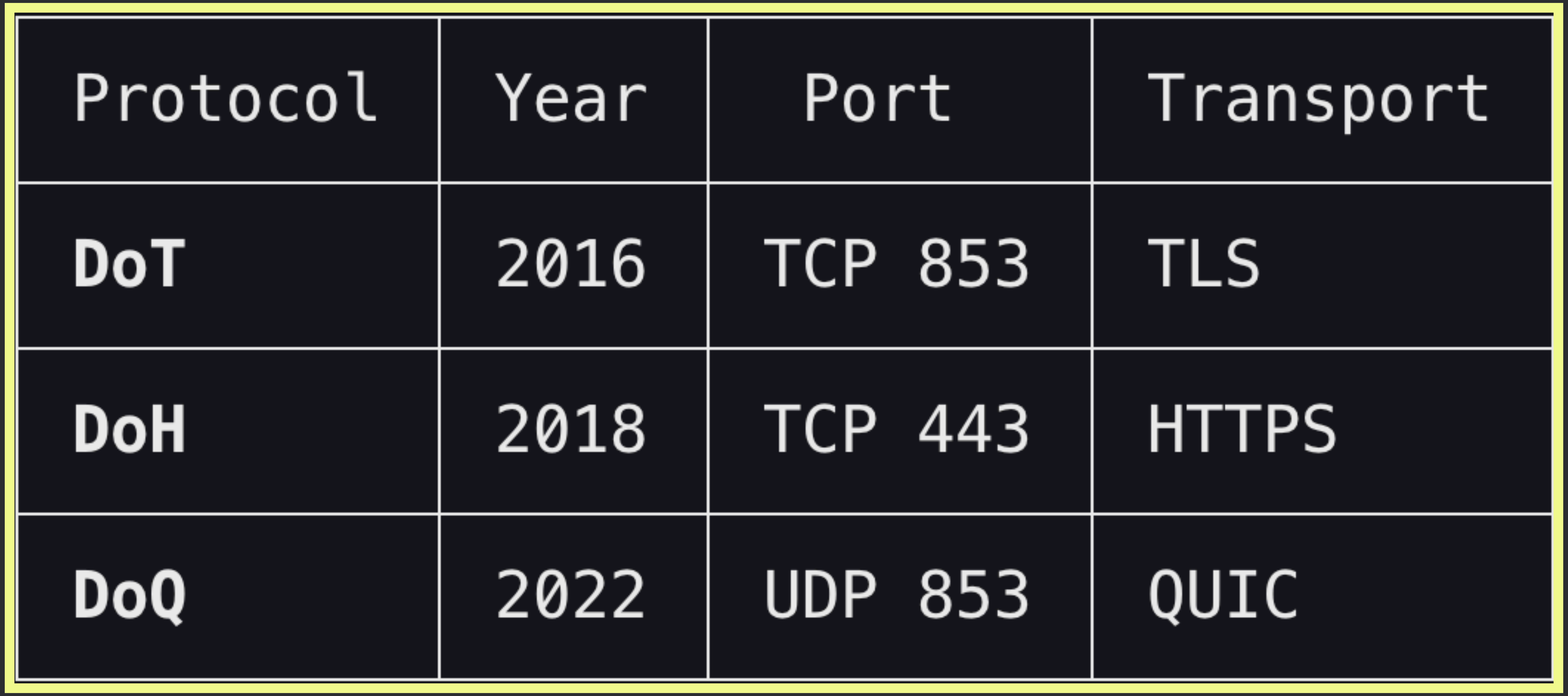


Field	Default Zeek Support
Client Subnet (ECS)	✅ Yes – dns_EDNS_ecs event
Padding	❌ No – need custom parsing
Private codes	❌ No – need custom parsing
OPT record presence	✅ Yes – visible in logs



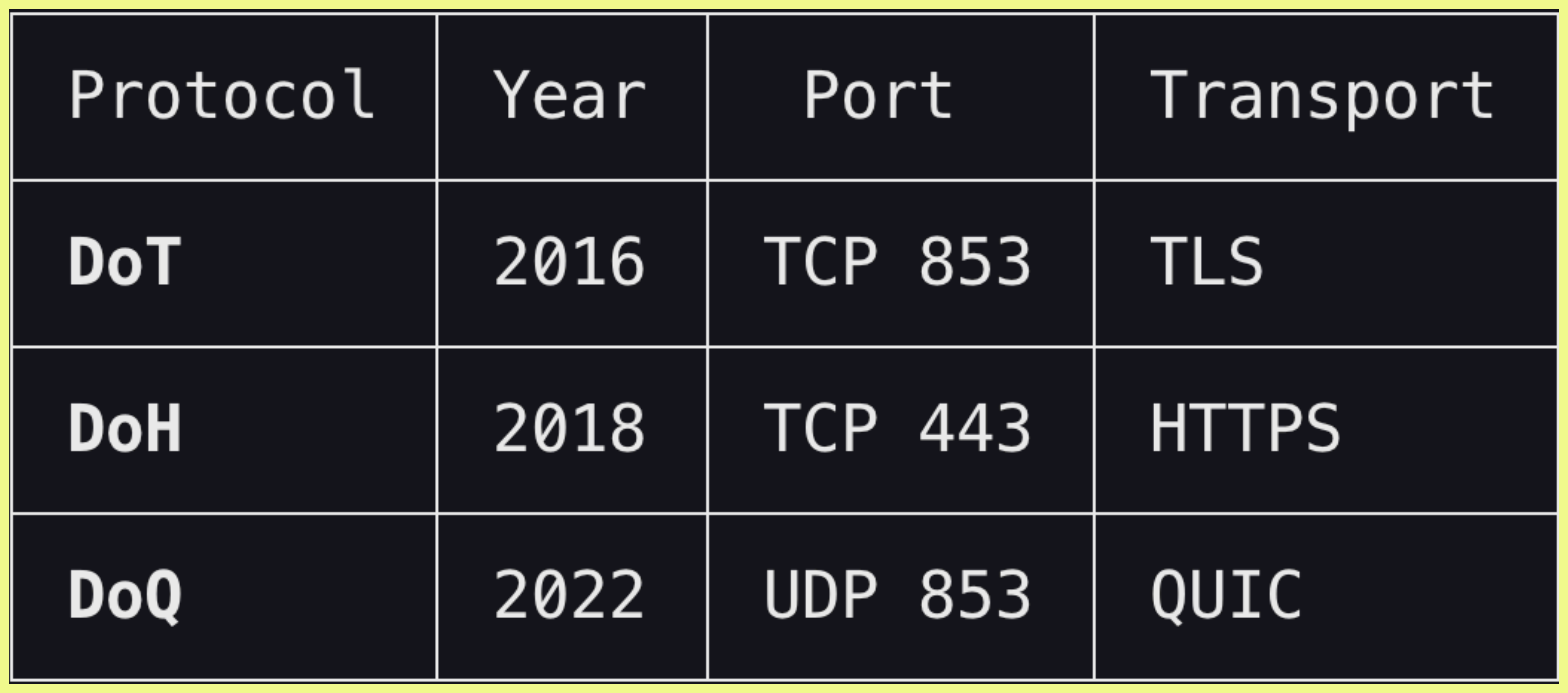
Encrypted DNS

3 Versions of DNS Encryption



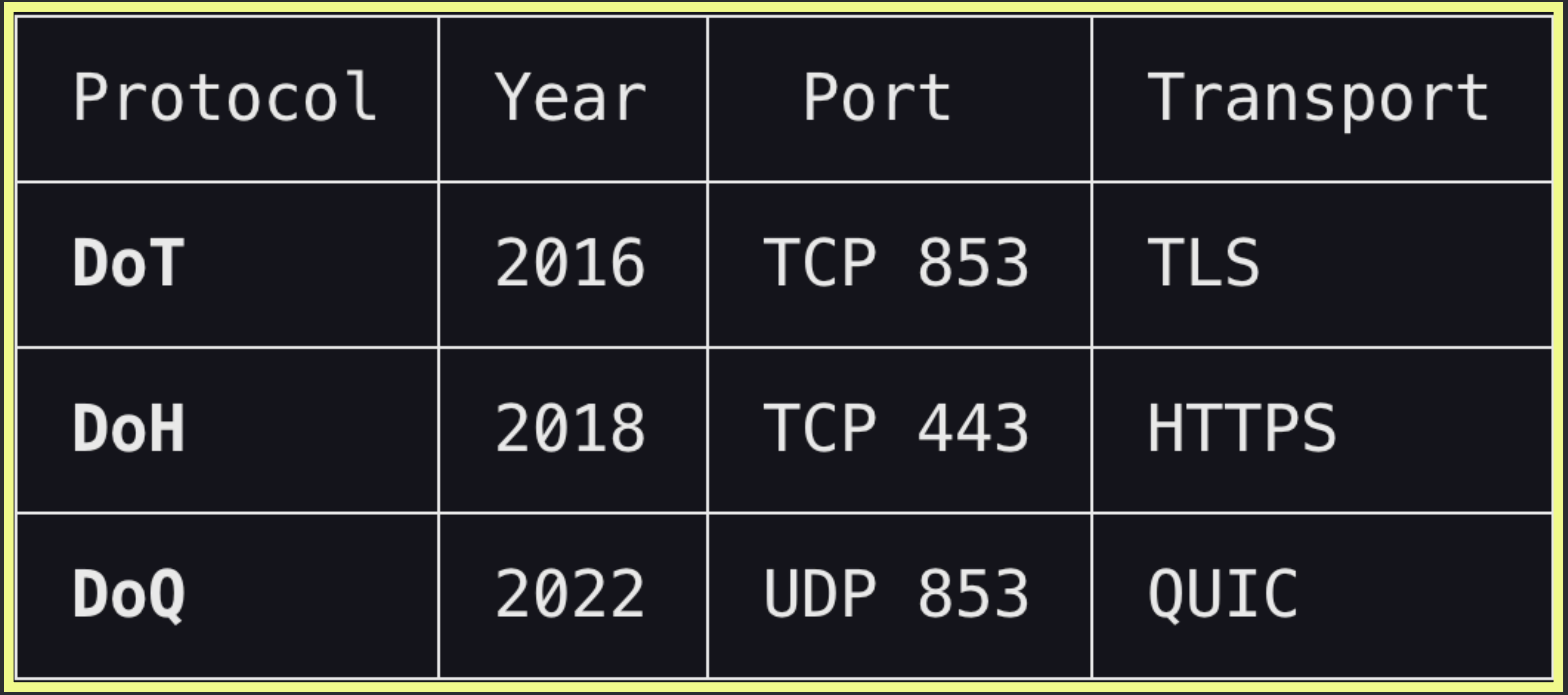
Protocol	Year	Port	Transport
DoT	2016	TCP 853	TLS
DoH	2018	TCP 443	HTTPS
DoQ	2022	UDP 853	QUIC

3 Versions of DNS Encryption



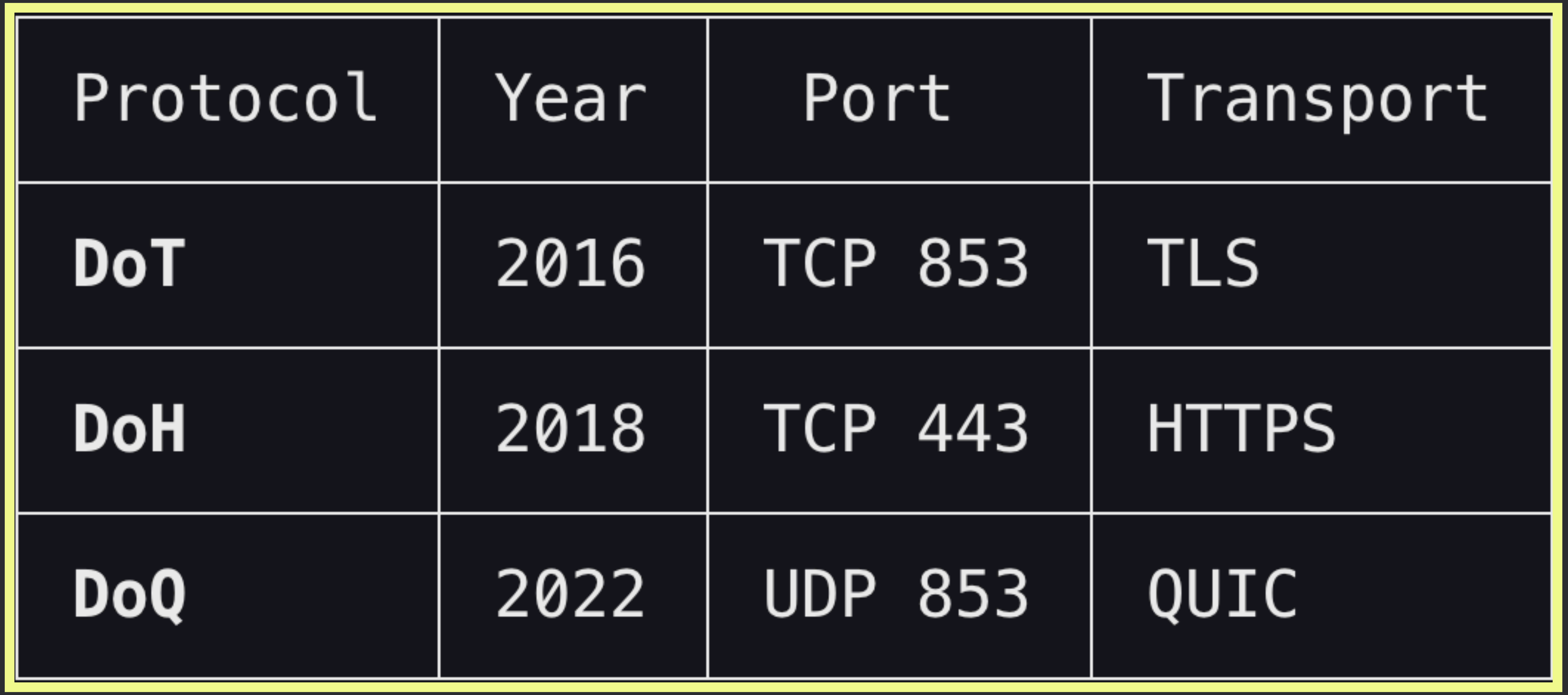
Protocol	Year	Port	Transport
DoT	2016	TCP 853	TLS
DoH	2018	TCP 443	HTTPS
DoQ	2022	UDP 853	QUIC

3 Versions of DNS Encryption



Protocol	Year	Port	Transport
DoT	2016	TCP 853	TLS
DoH	2018	TCP 443	HTTPS
DoQ	2022	UDP 853	QUIC

3 Versions of DNS Encryption



Protocol	Year	Port	Transport
DoT	2016	TCP 853	TLS
DoH	2018	TCP 443	HTTPS
DoQ	2022	UDP 853	QUIC

	DoT	DoH	DoQ
Blockable?	Easy (853)	Hard (443)	Easy (853)
Blends in?	No	Yes (looks like web)	No

So, DoT and DoQ SHOULD be blocked since most enterprises don’t need to use it.

Besides, they skip local resolvers!

Any application using it might complain, but will just revert to plaintext DNS in any case.

But cannot block DoH - looks like HTTPS

But then question from adversary’s POV becomes…

If it appears as HTTPS on network, then why not just use HTTPS - why constrain oneself to DoH at all?

Is there a benefit?

Kinda, yeah.

“Resolver-as-Proxy”

“Resolver-as-Proxy”

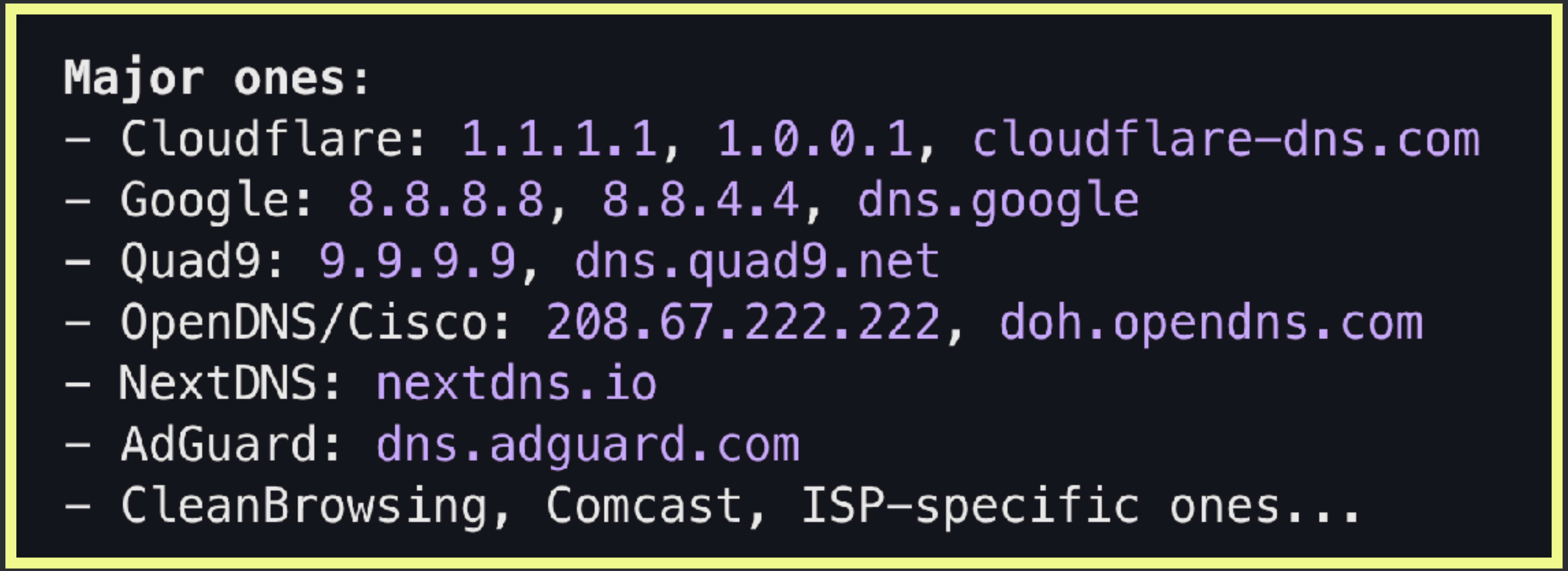
- Victim sends encrypted DNS query to 1.1.1.1 or 8.8.8.8
- Resolver decrypts, sees query for cmd.evil.com
- Resolver contacts attacker's auth nameserver to resolve it
- Attacker’s server returns data in the response
- Resolver encrypts and sends back to victim

Now obvs, unlike DoT and DoQ,

we can’t just block DoH/HTTPS

But, we can block the destinations

Known, finite list of DoH resolvers

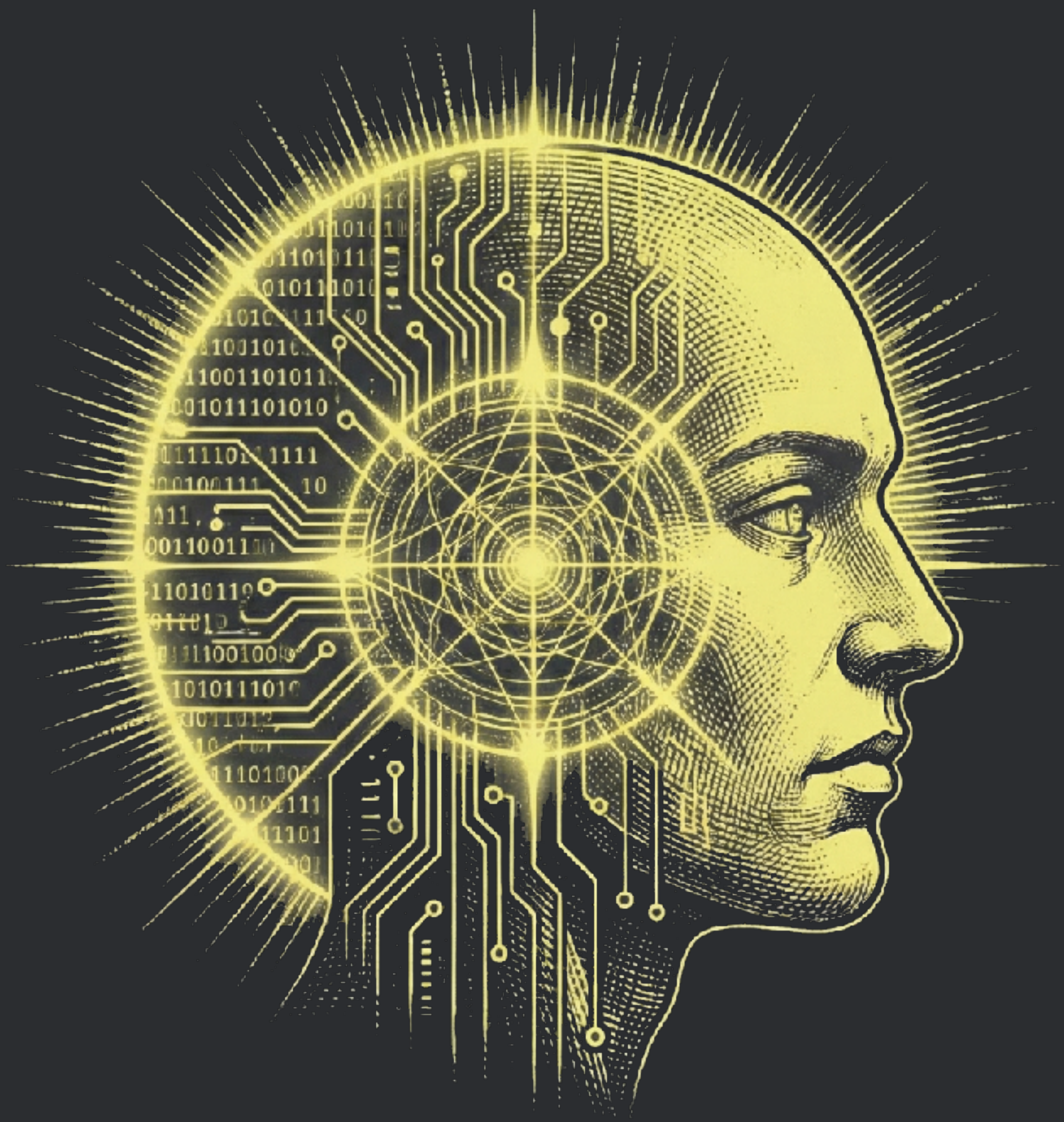


Major ones:

- Cloudflare: 1.1.1.1, 1.0.0.1, cloudflare-dns.com
- Google: 8.8.8.8, 8.8.4.4, dns.google
- Quad9: 9.9.9.9, dns.quad9.net
- OpenDNS/Cisco: 208.67.222.222, doh.opendns.com
- NextDNS: nextdns.io
- AdGuard: dns.adguard.com
- CleanBrowsing, Comcast, ISP-specific ones...

curl maintains a DoH providers list

If organization has internal DNS working as it should, then blocking these does not impact any business functions… Do it!



Main Takeaway

Understand that there are MANY ways to misuse DNS beyond using encoded subdomains for exfil

As we saw here, they are almost always easy to detect, but the key is - you have to look for them!

The specifics differ but if you:

→ Use Zeek + Blocklists (80%)

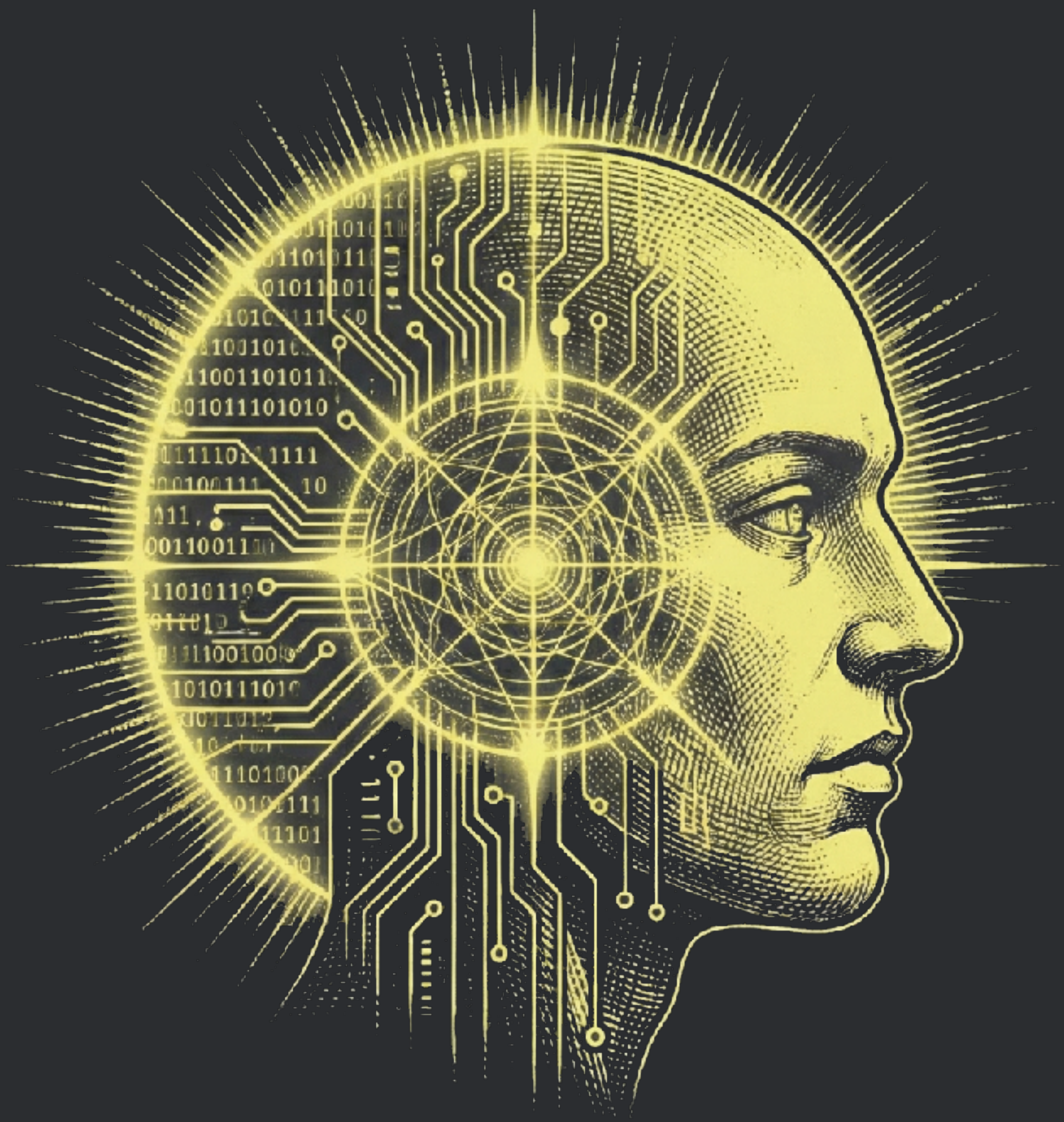
→ Add custom Zeek Scripts (95%)

→ Add custom parsers (99%)

Final thing to keep in mind…

Adversaries operate under a law

Inverse relationship between between stealth and operational efficiency



The Workshop

January 23 - Next Friday

- Build a Reflective Shellcode Loader C2 in Golang
- Brand new, focus on integrating EP action!
- Emphasis on design/patterns/architecture
- Lots (even more) value in “Agentic” revolution
- Sliding scale, $25 minimum - PLEASE JOIN!

www.faanross.com

www.aionsec.ai

thank you!