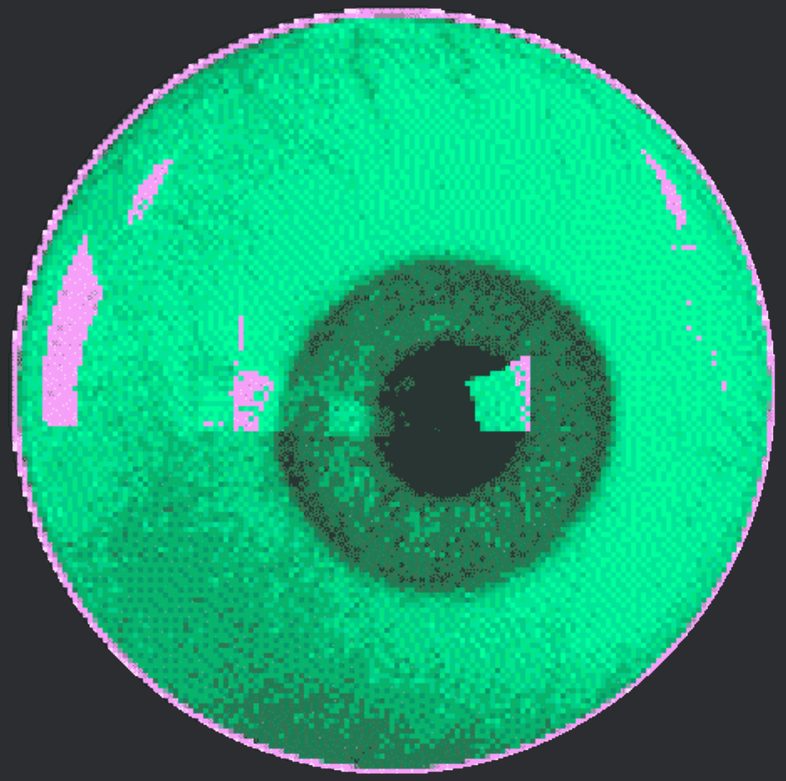
let’s build a



covert **C2 channel**

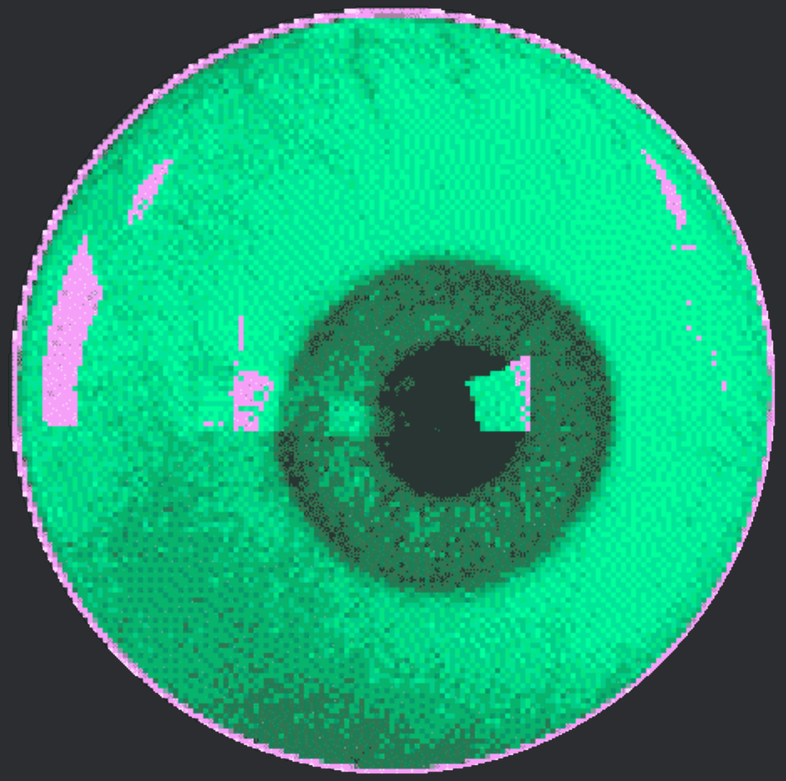
Faan Rossouw

- dad, husband, researcher
- trail running + MTB'ing
- 💜 Springbok rugby
- 💀 post-exploit maldev
- 💀 covert channels
- 💀 offensive sec tooling
- 💀 ai (for) red teaming

→ faanross.com ← free content + courses


4 hr workshop
18 Sept
Tomorrow!
$25! JOIN US!
UNISYS DIRECT

let’s build a



covert **C2 channel**

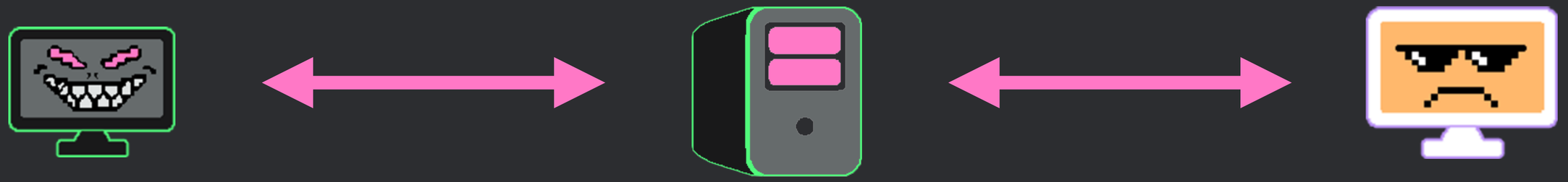
what is a covert channel?

most elementary C2 setup

C2 Client

C2 Server

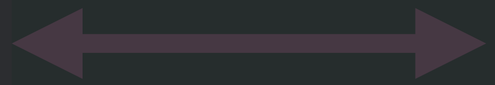
C2 Agent

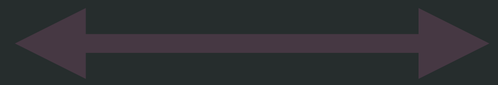


C2 Client

C2 Server

C2 Agent





C2 Client

C2 Server

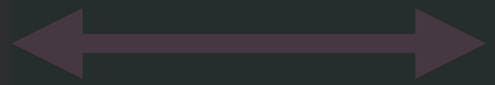
C2 Agent

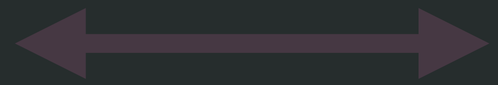
operator

C2 Client

C2 Server

C2 Agent

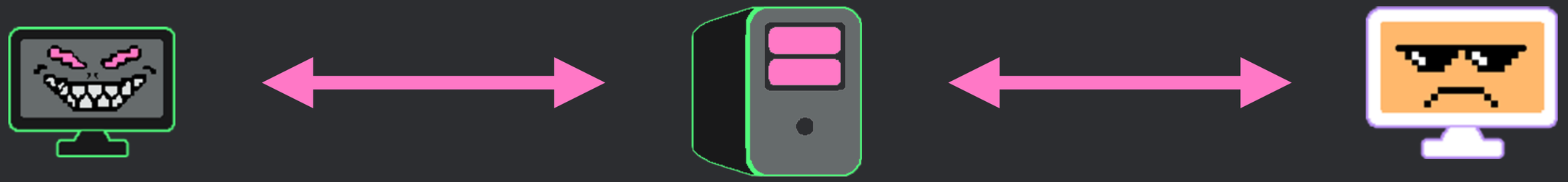


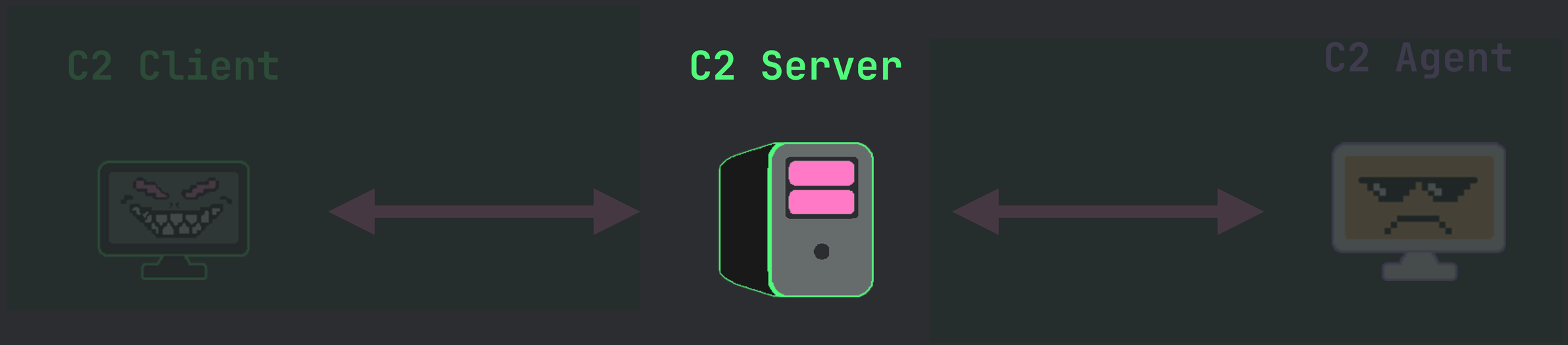


C2 Client

C2 Server

C2 Agent




C2 Client
C2 Server
C2 Agent

C2 Client

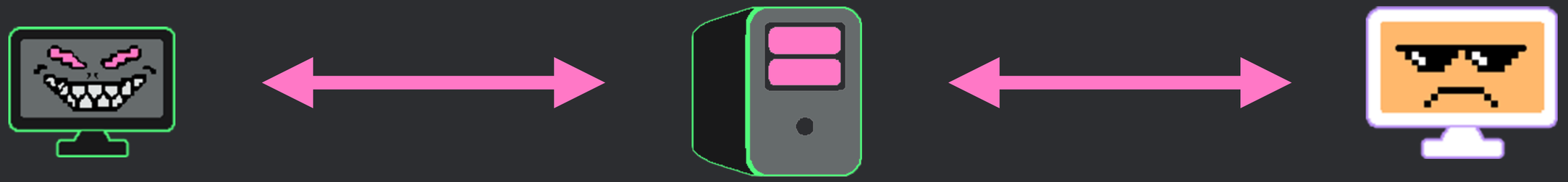
C2 Server

C2 Agent

C2 Client

C2 Server

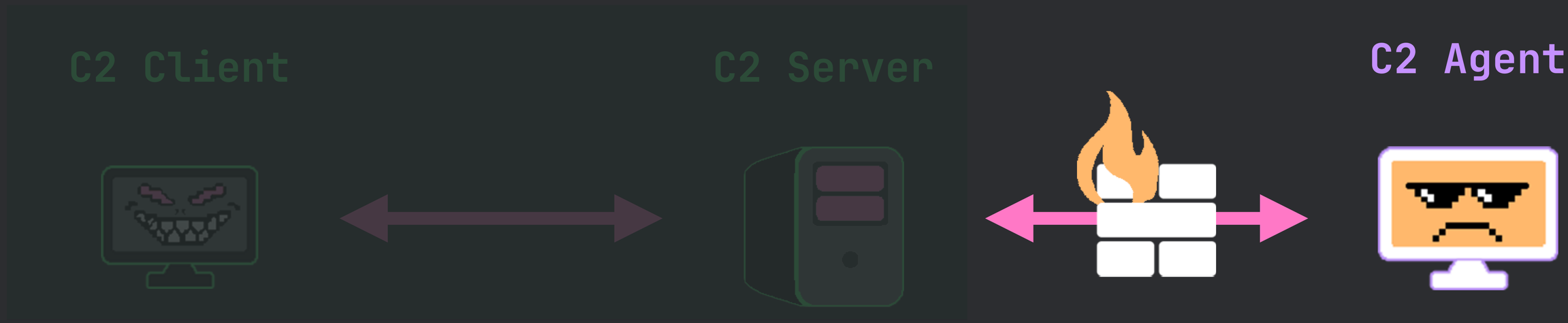
C2 Agent

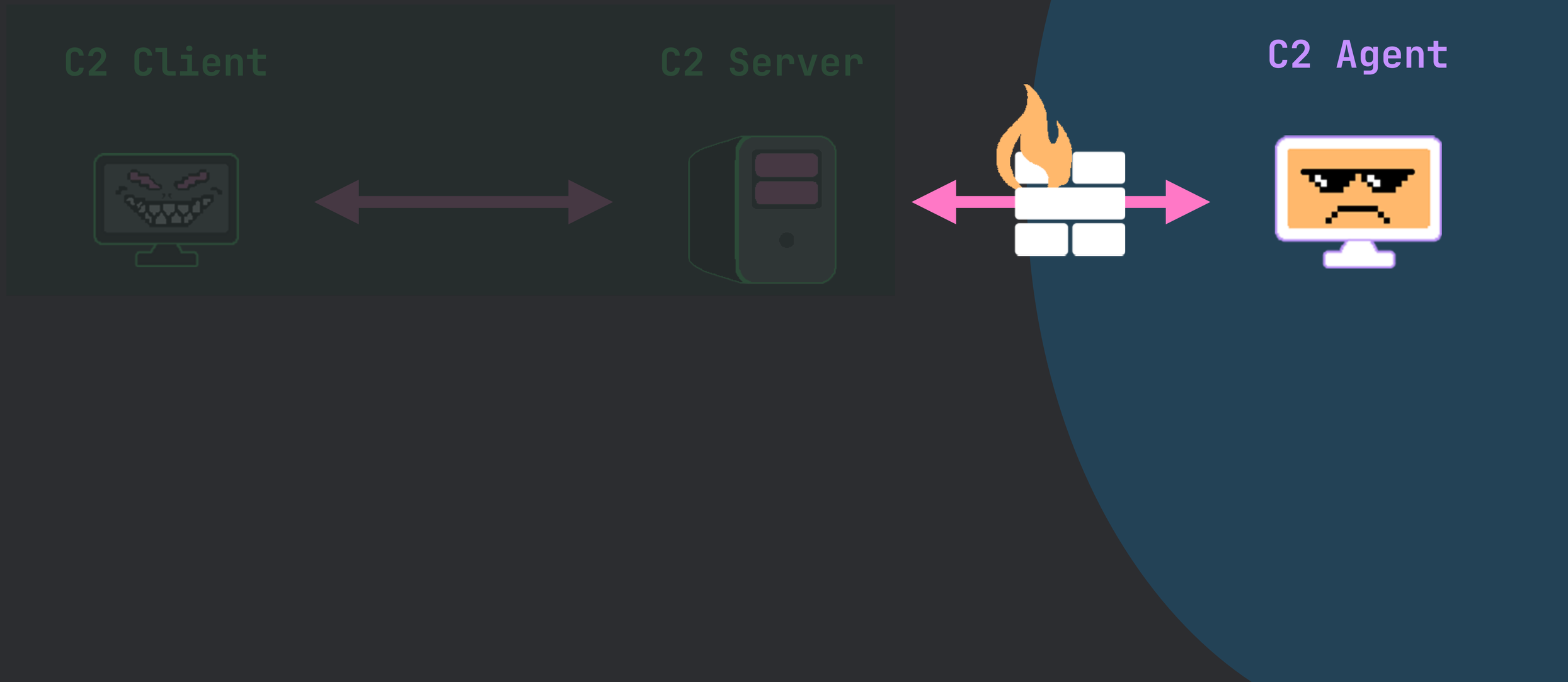


C2 Client

C2 Server

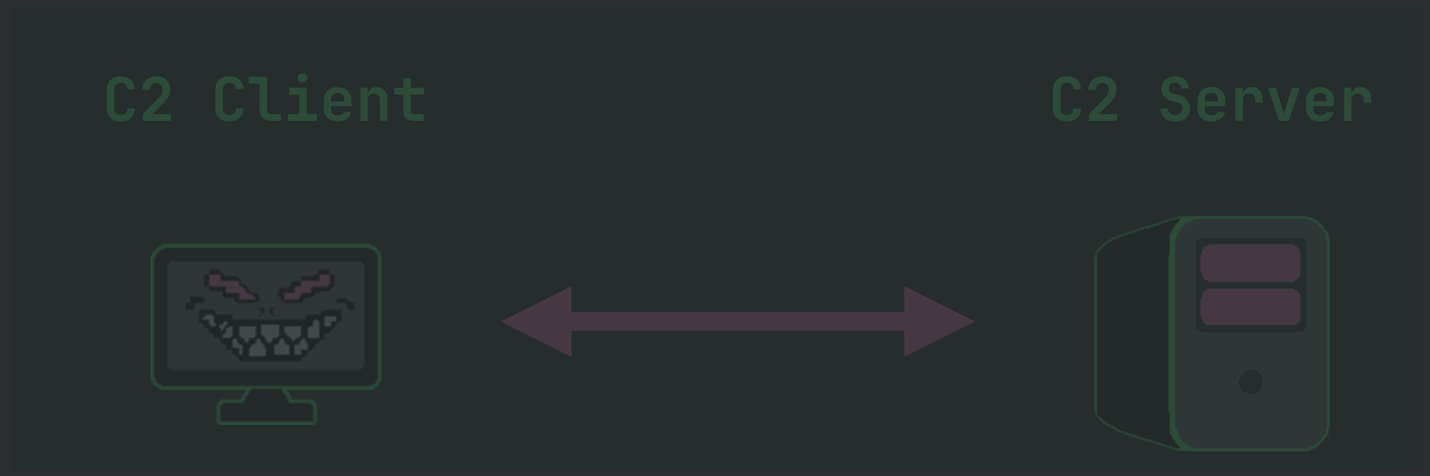
C2 Agent


C2 Client
C2 Server
C2 Agent


C2 Client
C2 Server
C2 Agent

C2 Client

C2 Server



C2 Agent

internal

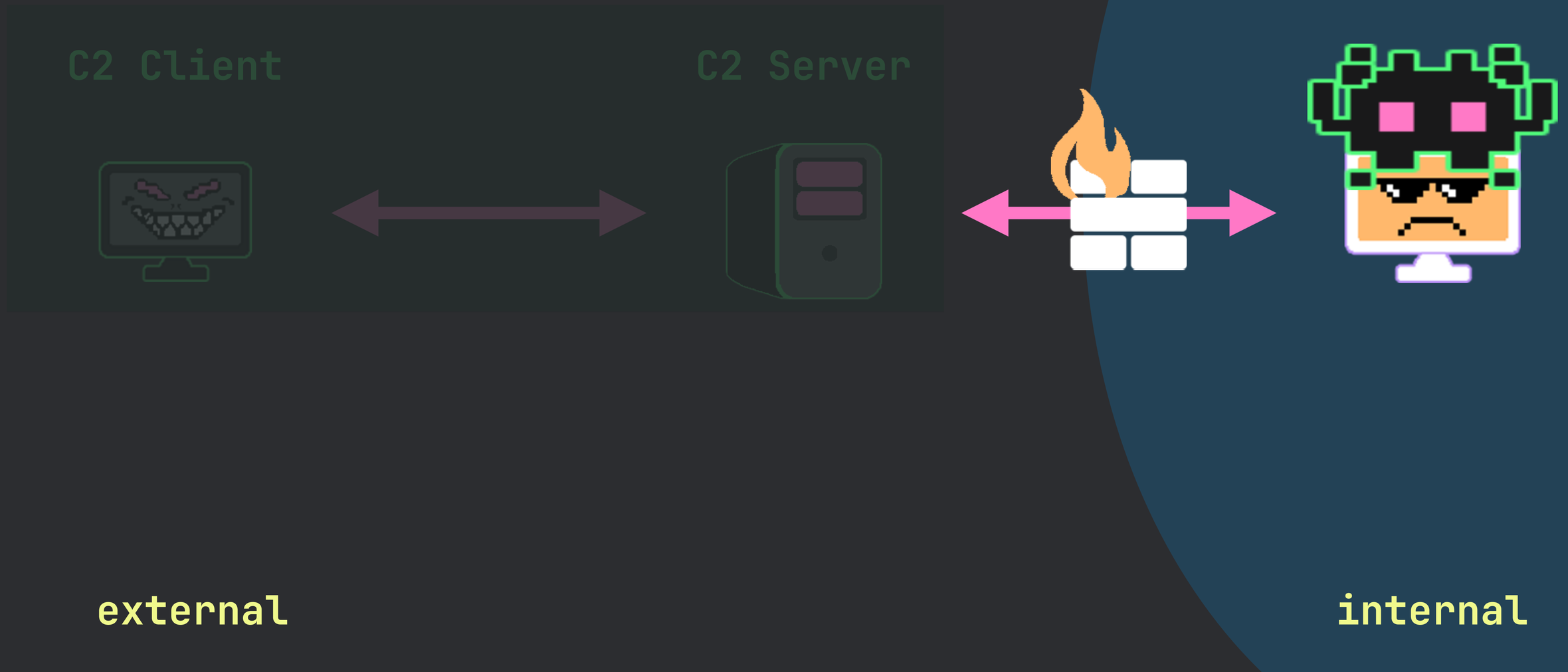
C2 Client

C2 Server

C2 Agent

external

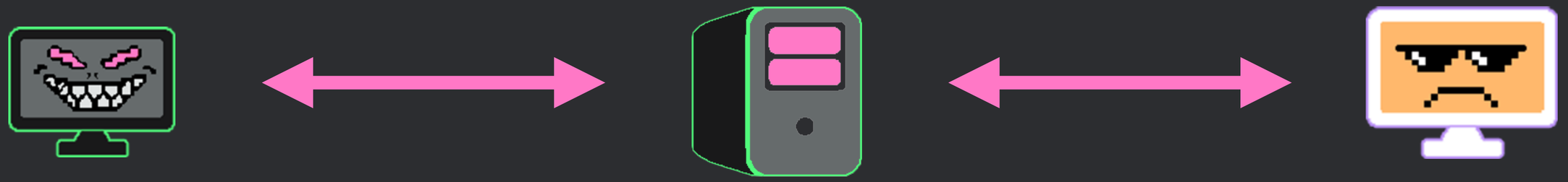
internal


C2 Client
C2 Server
external
internal

C2 Client

C2 Server

C2 Agent



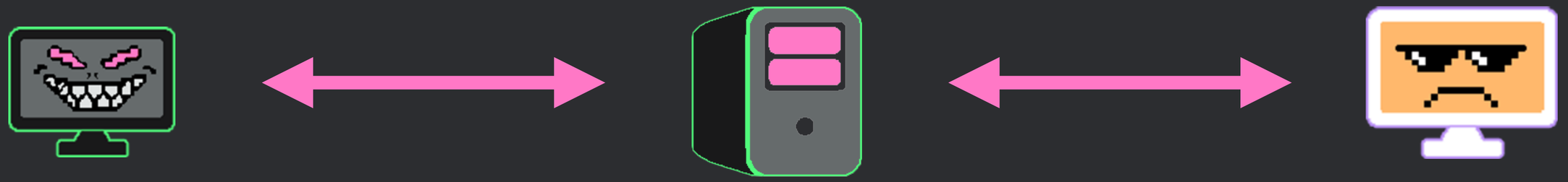
so… about that

covert channel?

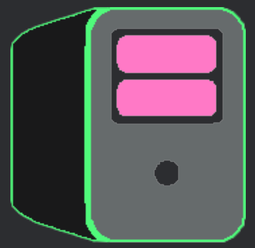
C2 Client

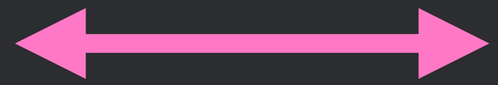
C2 Server

C2 Agent



C2 Server





C2 Agent

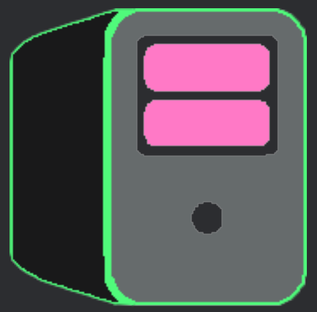
C2 Server

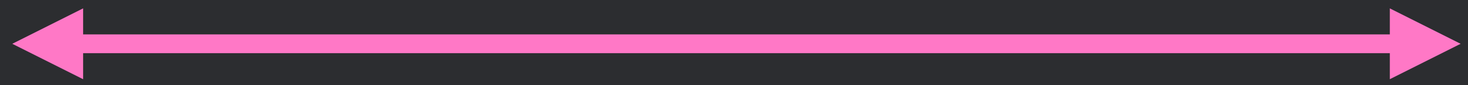
C2 Agent

C2 Server

C2 Server

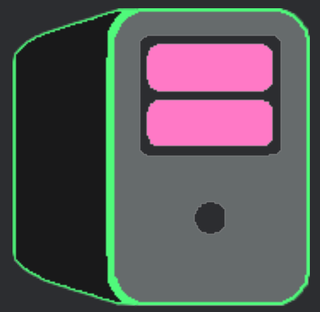
C2 Agent

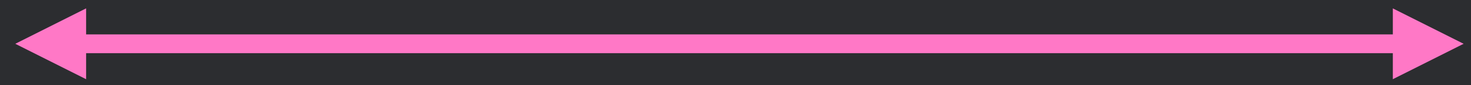




C2 Server

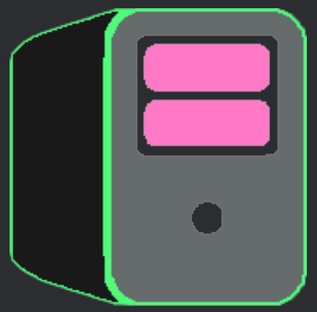
C2 Agent

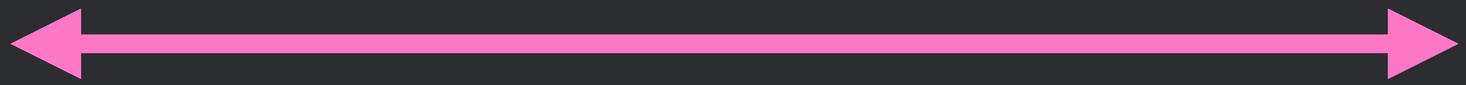


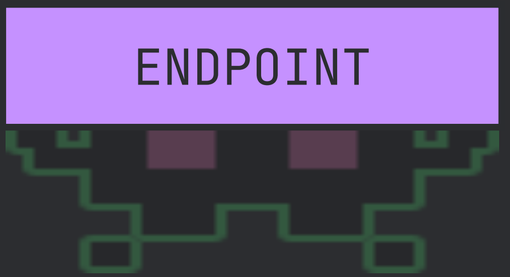


C2 Server

C2 Agent

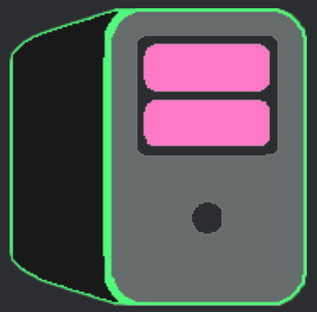


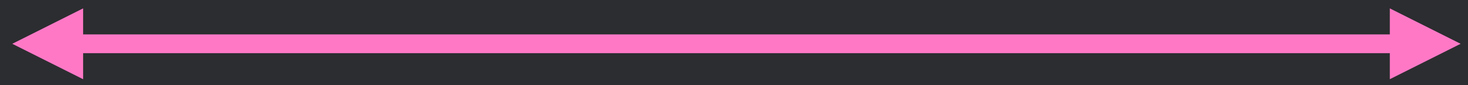




C2 Server

C2 Agent



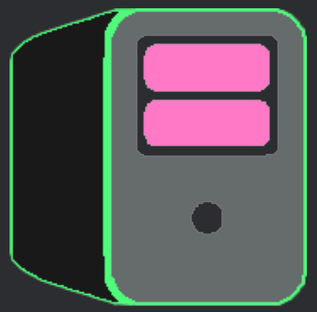


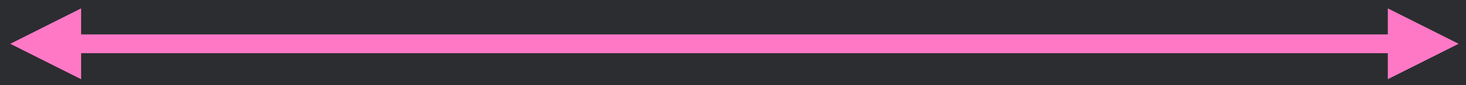
ENDPOINT

MANAGEMENT

C2 Server

C2 Agent





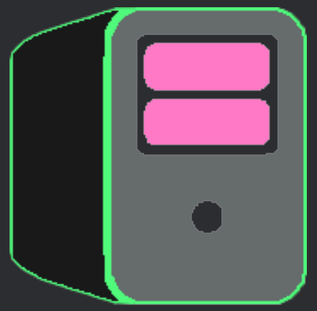
ENDPOINT

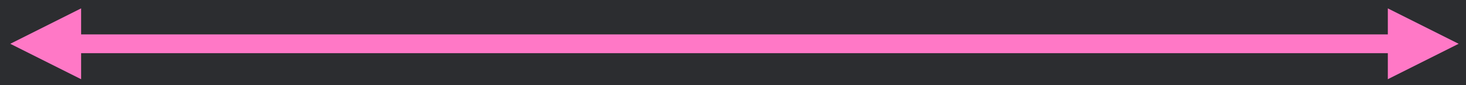
MANAGEMENT

COMMUNICATION

C2 Server

C2 Agent





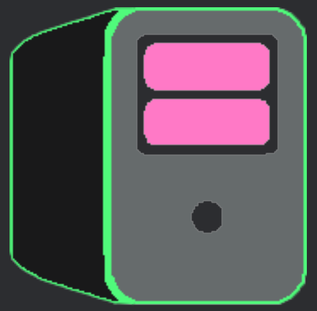
MANY THINGS
(black box)

ENDPOINT

MANAGEMENT

COMMUNICATION

C2 Server



MANY THINGS
(black box)

COMMUNICATION

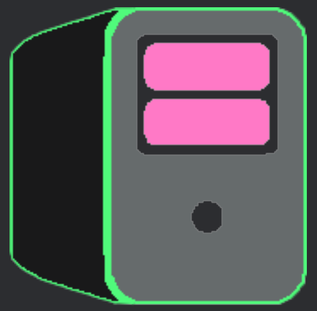
C2 Agent

ENDPOINT

MANAGEMENT

COMMUNICATION

C2 Server



MANY THINGS
(black box)

COMMUNICATION

C2 Agent

ENDPOINT

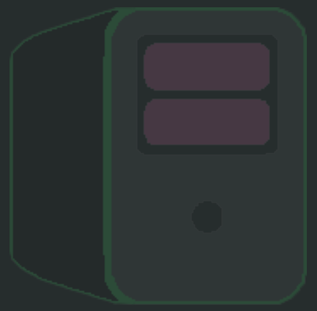
MANAGEMENT

COMMUNICATION

covert channel

C2 Server

C2 Agent



MANY THINGS
(black box)

COMMUNICATION

ENDPOINT

MANAGEMENT

COMMUNICATION

covert channel

specifics → rest of the lecture

covert channel

| goal

| communicate (data) between server + agent

| maximize operational efficiency + reliability

| maximize stealth and network evasion

stealth ⟷ operational efficiency

stealth ⟷ operational efficiency

↑ stealth ↓ operational efficiency

↑ operational efficiency ↓ stealth

goal as malware designers

optimize the
relationship
between stealth
+ operational
efficiency

goal as malware designers

optimize the relationship between stealth + operational efficiency

today’s mantra:

- rules create constraints to help us learn
- as we progress, this can limit us
- discernment re: what rules can be bent

→ can HTTP carry encrypted data? ←

→ can UDP be reliable? ←

→ can a channel use multiple protocols? ←

a major paradigm shift occurred for me when learning about…

SUNBURST

SUNBURST

- well-documented
- complex

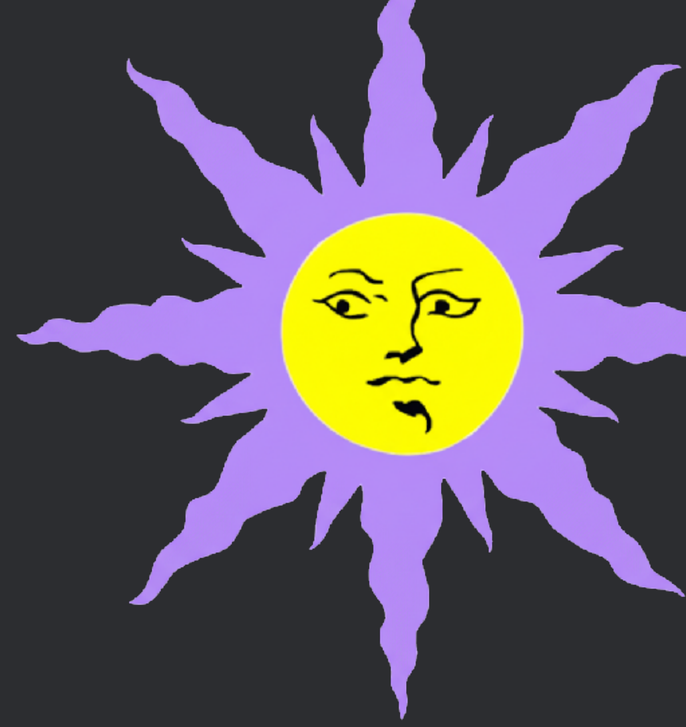
SUNBURST

| network strategy

| simplified

SUNBURST AGENT

NETWORK COMMS



→ multi-stage, multi-protocol design

→ strategically prioritizes stealth

→ each stage: stealth ⟷ operational efficiency?

RISK + REWARD

following initial access…

stage	description
0	pre-flight checks

stage	description
0	pre-flight checks

→ pre-communication

→ automated assessment

RISK + REWARD

stage	description
0	pre-flight checks
1	DNS

stage	description
1	DNS

stage	description
1	DNS

stage	description
1	DNS

→ state-driven channel

→ semantic meaning of response = signal

stage	description
1	DNS

RISK + REWARD →

→ high stealth

→ low bandwidth

→ vs 0: more freedom to assess

stage	description
1	DNS
2	HTTPS

stage	description
2	HTTPS

stage	description
2	HTTPS

stage	description
2	HTTPS

→ less stealth, high bandwidth

→ gamut of typical C2 functions

stage	description
2	HTTPS

→ in most cases, ends here

→ but, for the chosen few…

RISK + REWARD →

stage	description
2	HTTPS
3	TEARDROP → mod. CS Beacon

RISK + REWARD

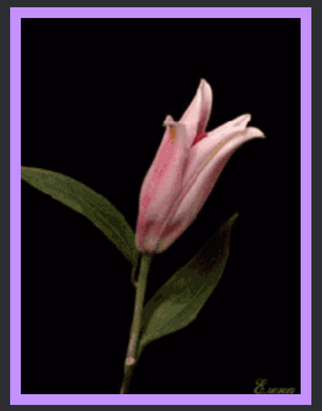
stage	description
3	TEARDROP → mod. CS Beacon

stage	description
3	TEARDROP → mod. CS Beacon

stage	description
3	TEARDROP → mod. CS Beacon

→ less stealth → sigs + aggression

→ access CS’s post-exploit toolbox



FINAL + FULL EXPRESSION

stage	description
3	TEARDROP → mod. CS Beacon

stage	description
3	TEARDROP → mod. CS Beacon

stage	description
0	pre-flight checks
1	DNS
2	HTTPS
3	TEARDROP → mod. CS Beacon

sunburst design

optimize the relationship between stealth + operational efficiency by using multi-stage, multi-protocol design.

risk + reward

each transition to a higher stage is based on calculating risk + reward, using 4 degrees of variable stealth ⟷ operational efficiency

the story

technical guide WIP

www.faanross.com/deep_dives/malware/sunburst

feeling inspired

Let’s start with the most important part…


What is your name?

bobbejaan

bobbejaan-spinnekop

bobbejaan-spinnekop

bobbejaan-spinnekop

communication protocol

communication protocol

- HTTP, HTTPS, WS/S
- DNS, DoH, DoT
- ICMP, NTP, IRC, SSH
- RAW TCP, UDP, QUIC
- TOR, RatNet, VPN
- Write your own

The Skimask Principle

”conspiciously inconspicuous”

Sometimes the best way to not be noticed is to not try too hard to not be noticed

don’t think about

hiding

it’s more about

blending in

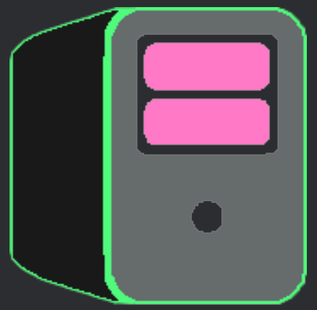
“Wait a minute! Isn’t anyone here a real sheep?”

for this reason (+ others)

I like DNS and HTTPS

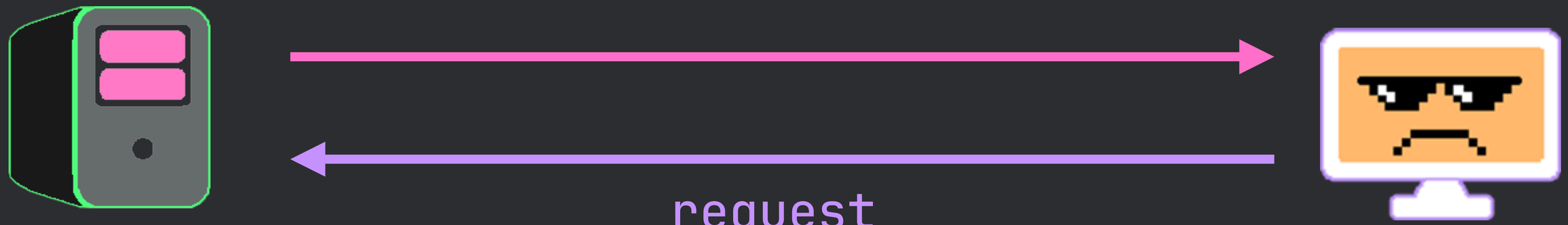
dns as covert channel

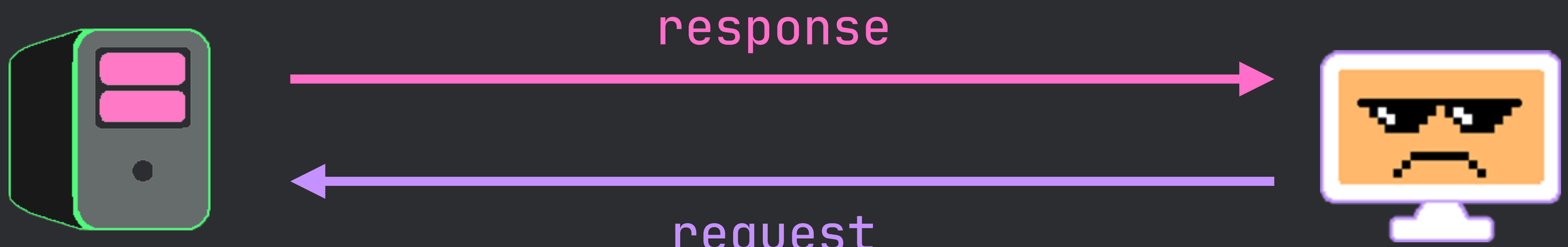
Server

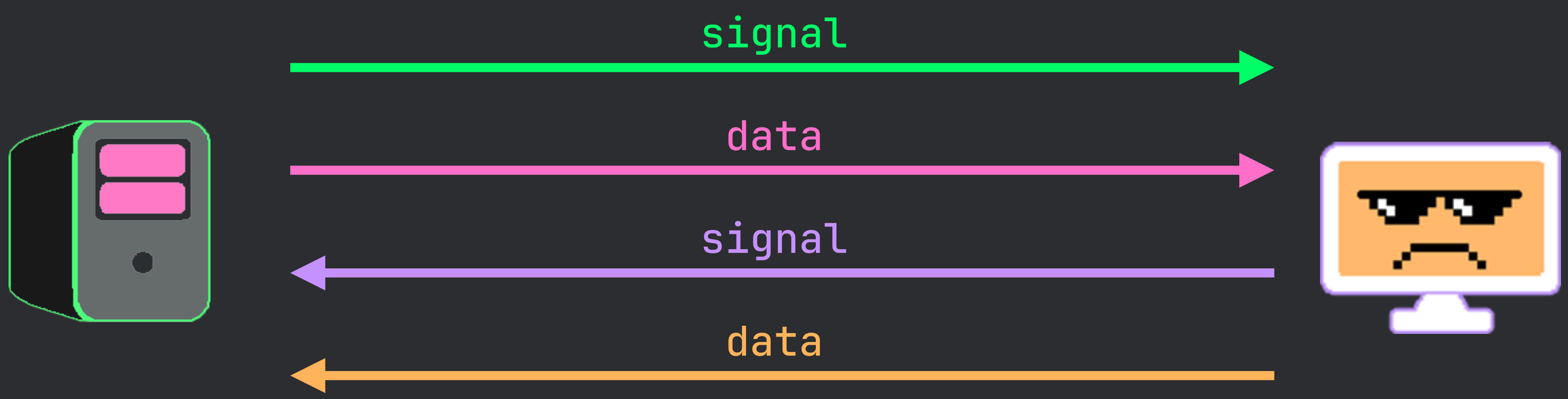


Agent

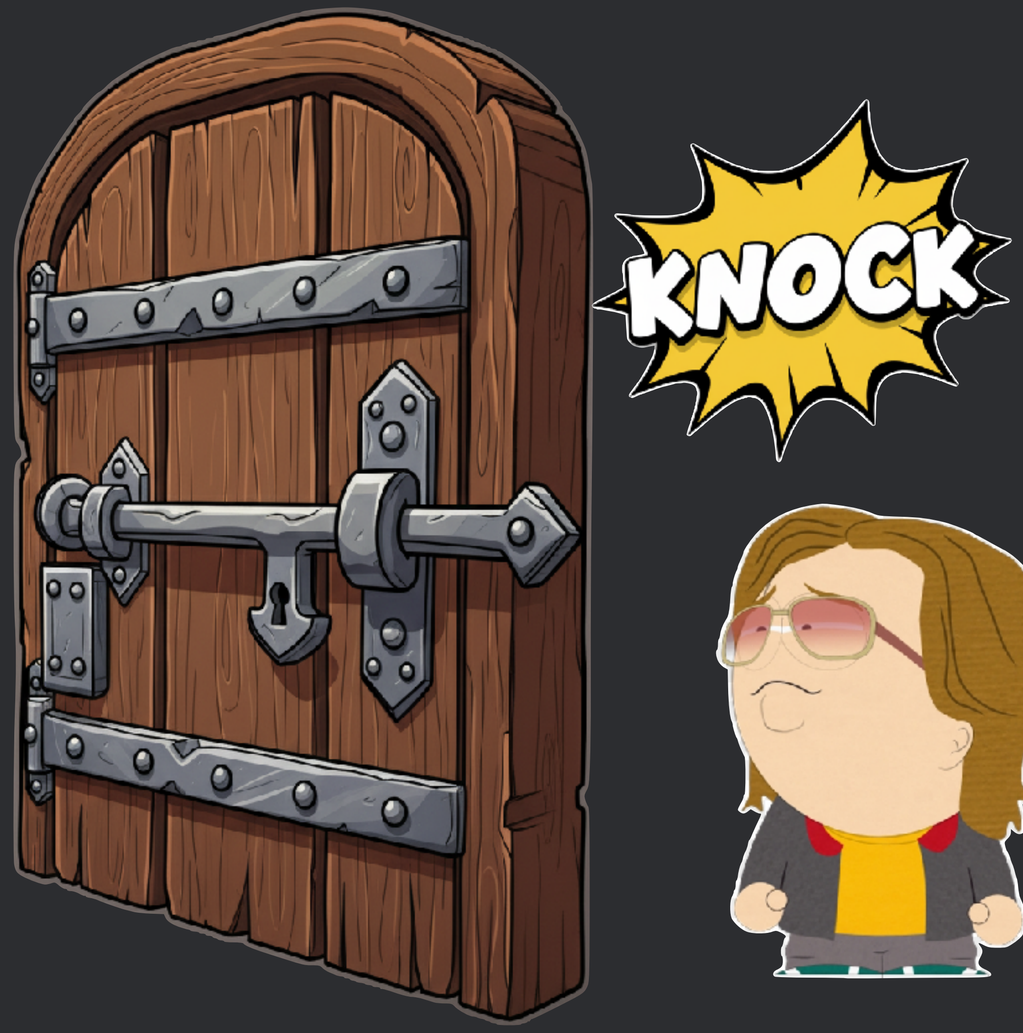

Server
Agent


request

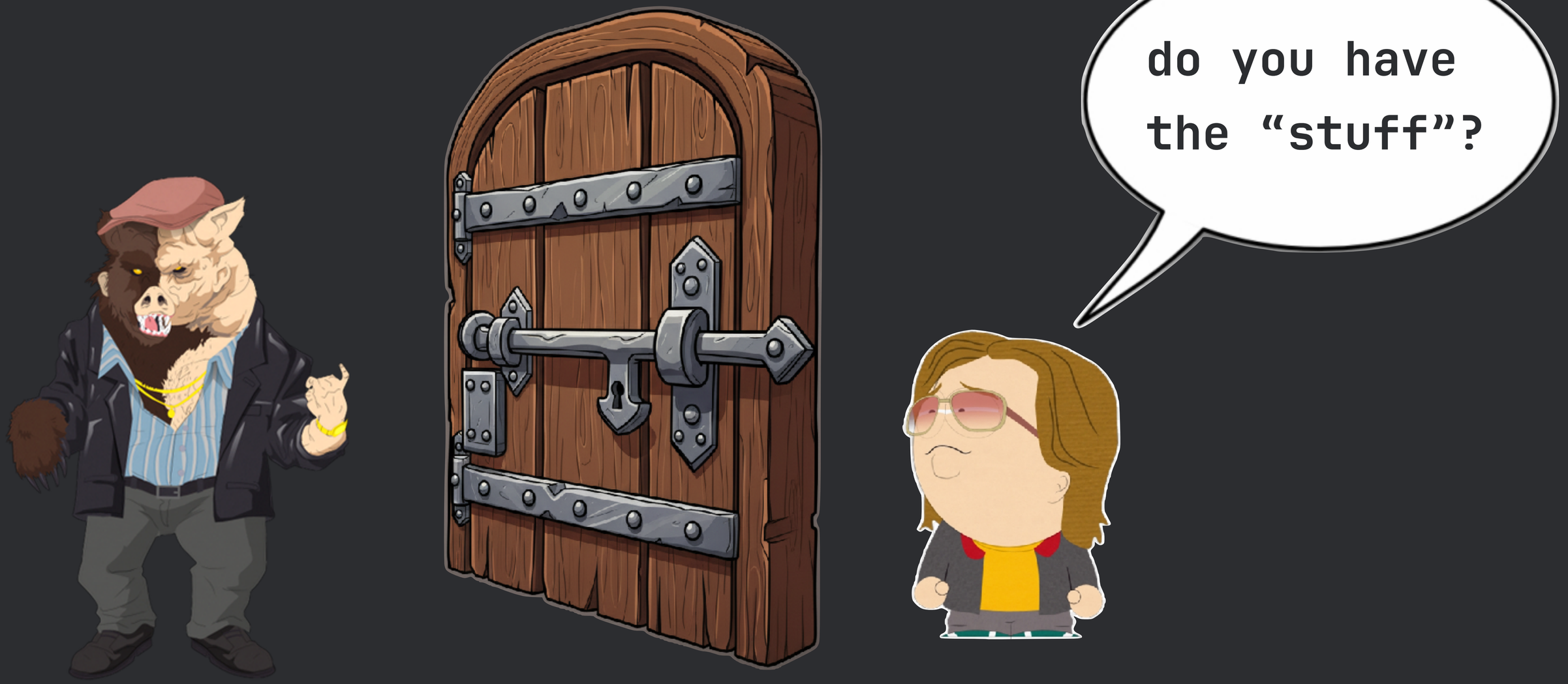

response
request


signal
data
signal
data

signal vs data


KNOCK


KNOCK


do you have
the “stuff”?

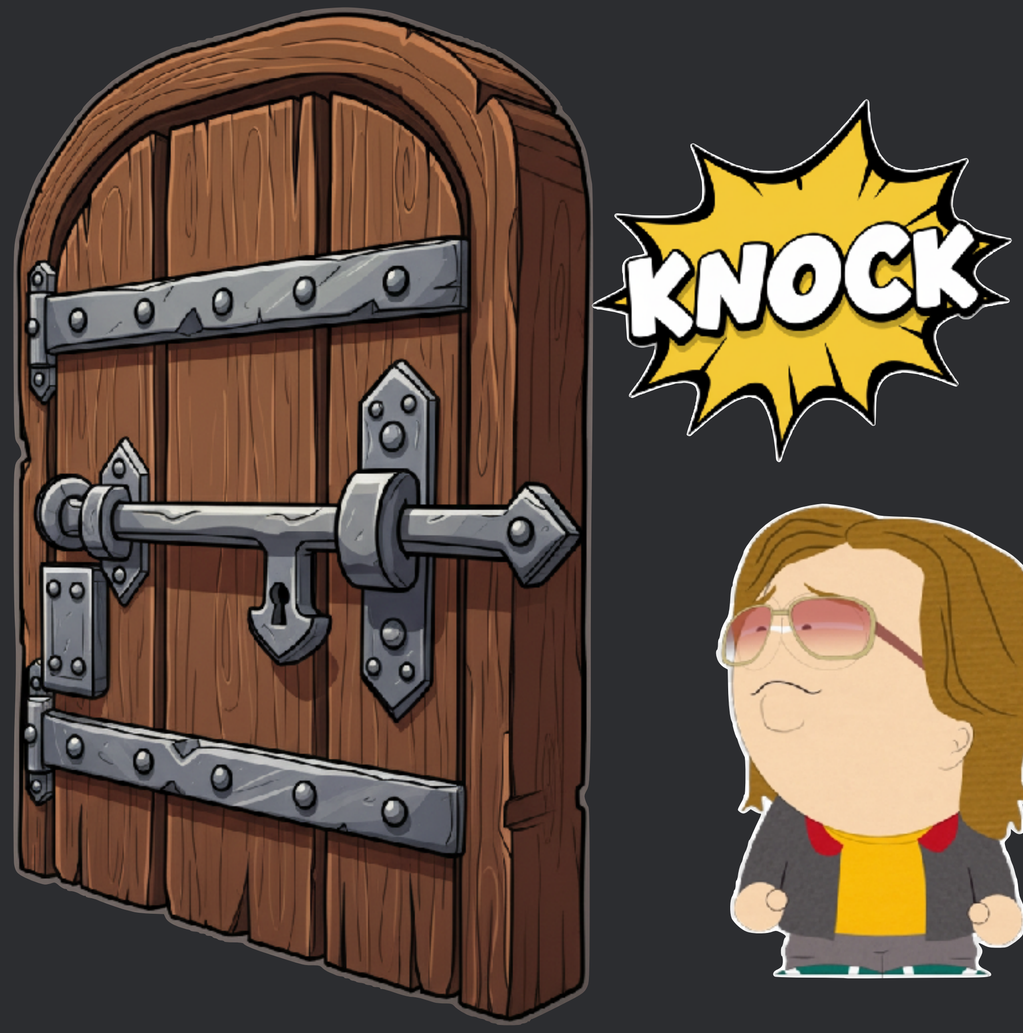

Stuff


Stuff


Stuff


Stuff

DATA → Meaning/Intention Explicit


KNOCK


KNOCK


KNOCK


KNOCK


KNOCK


KNOCK


KNOCK


Stuff


Stuff


Stuff

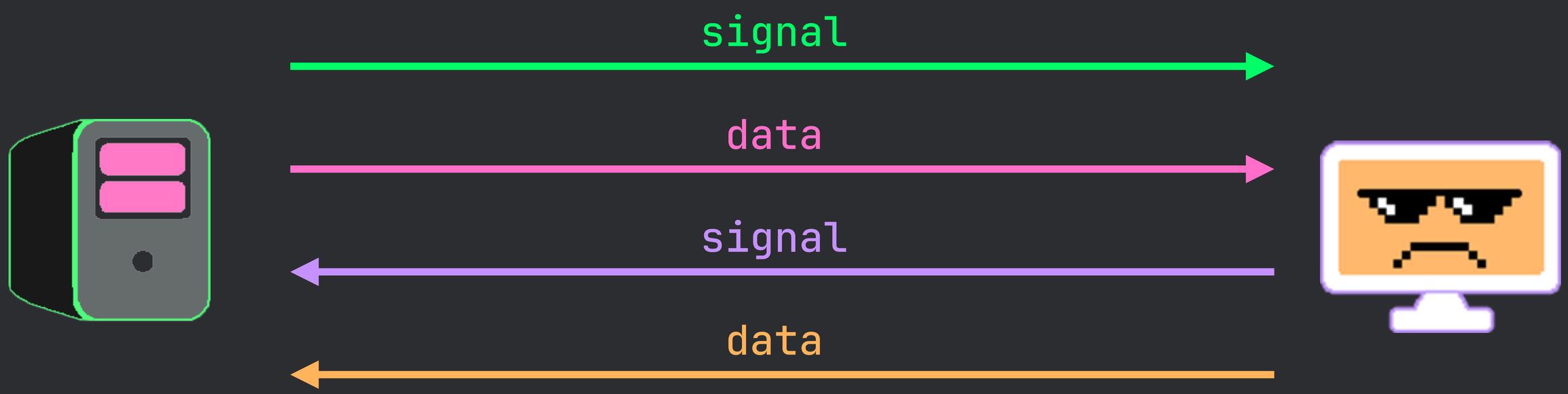

Stuff

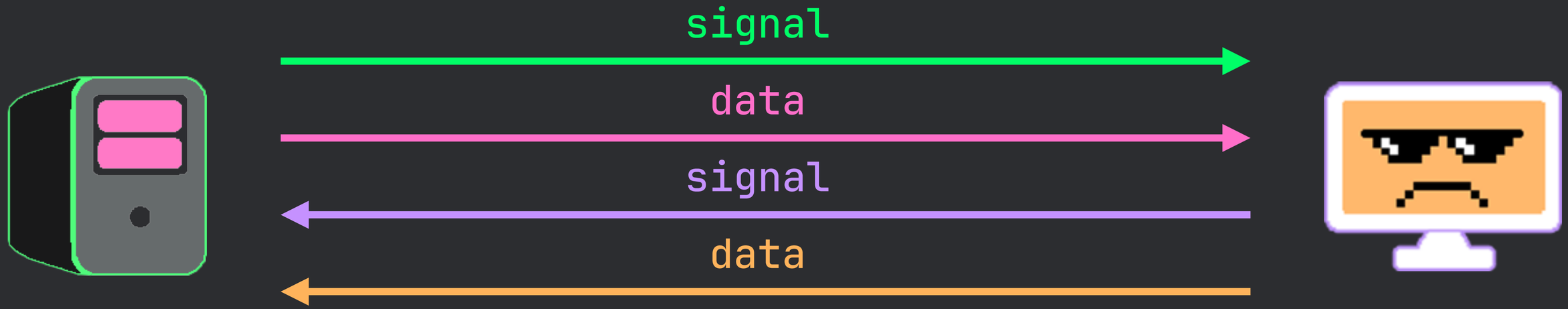

Stuff

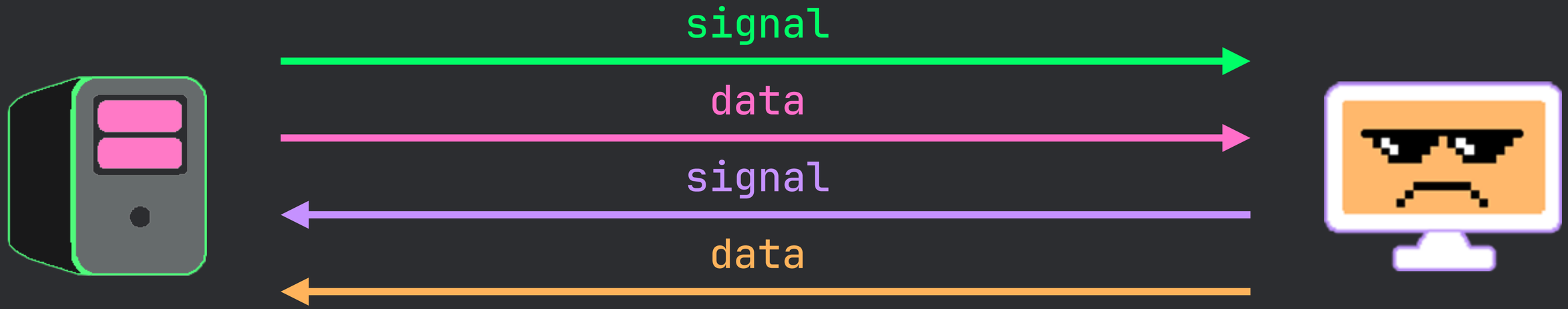
SIGNAL → Meaning/Intention Implicit

| can be form of subterfuge

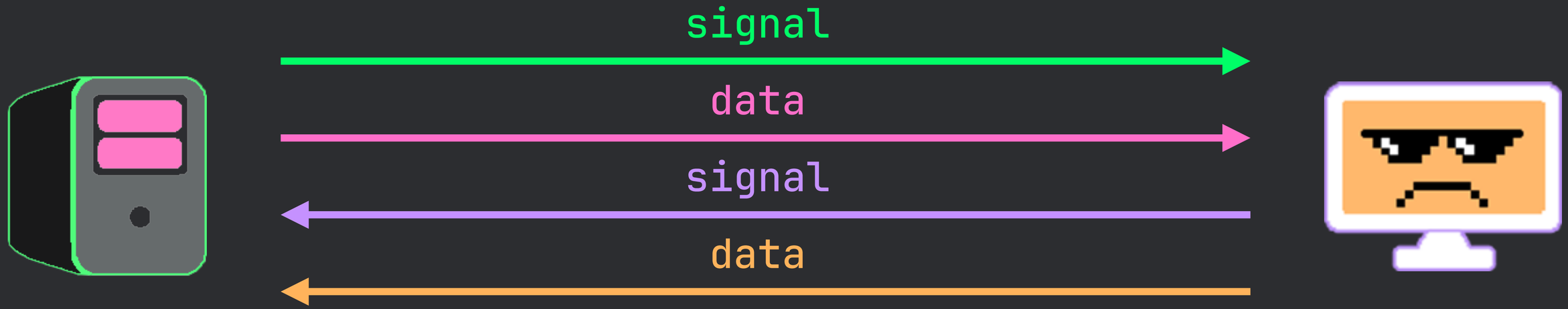
| can be due to constraint

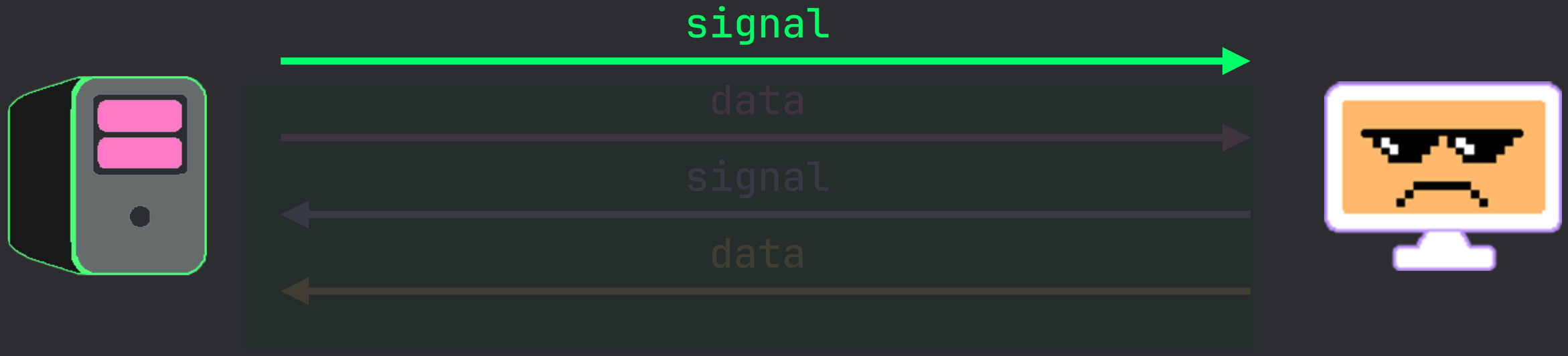

signal
data
signal
data

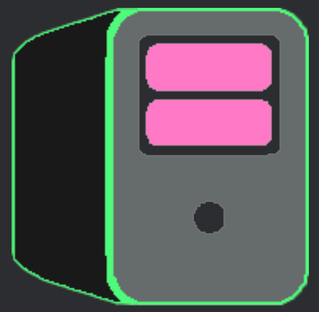

signal
data
signal
data



DNS


signal
data
signal
data


signal
data
signal
data



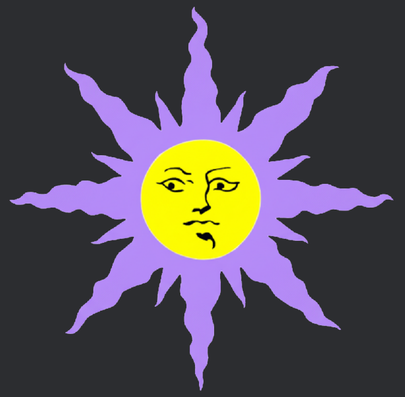
signal

data

signal

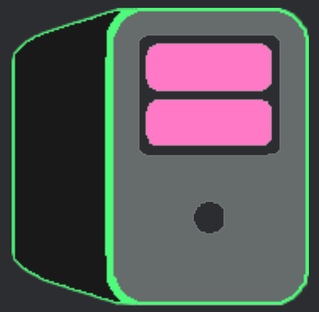
data

sunburst → DNS Reply A Record IP



→ Different Ranges = Signals

21.0.0.0/8 - 30.0.0.0/8	→ Command 1 (enumerate)
31.0.0.0/8 - 40.0.0.0/8	→ Command 2 (sleep 1 hr)
41.0.0.0/8 - 50.0.0.0/8	→ Command 3 (HTTPS)



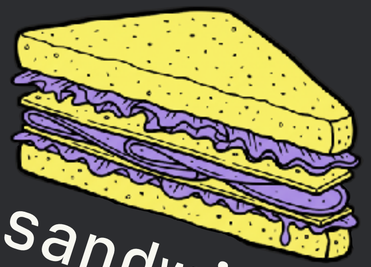
signal

data

signal

data

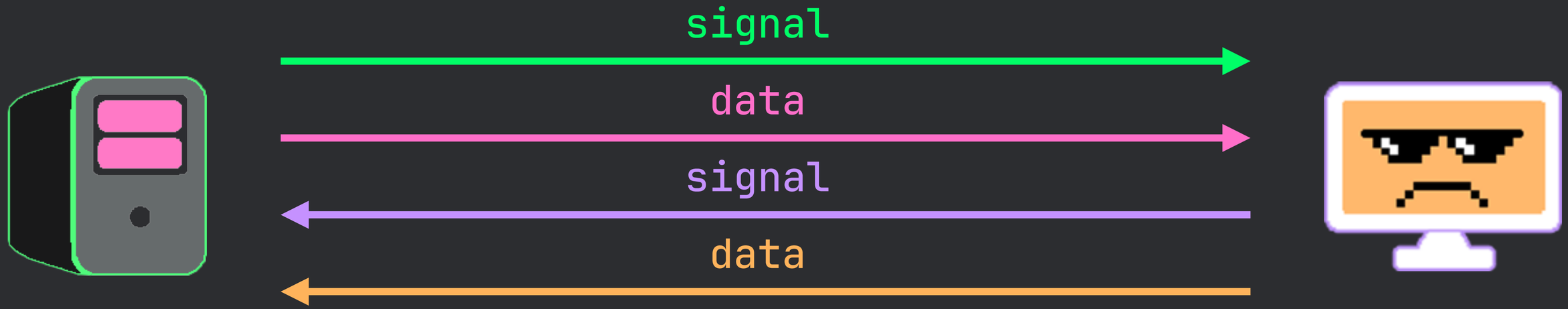
DNS

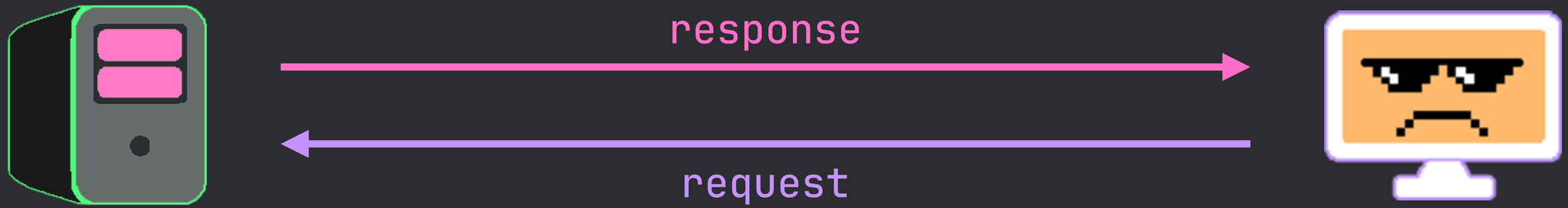


sandwich

→ 2 fields we can co-opt

→ let’s look at structure


signal
data
signal
data


response
request

HEADER

QUESTION

HEADER

QUESTION

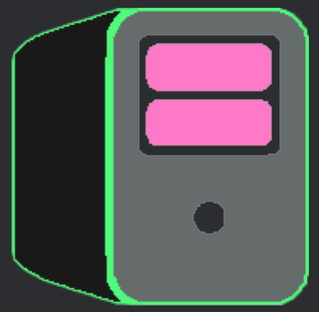
HEADER

QUESTION

HEADER

QUESTION

ANSWER



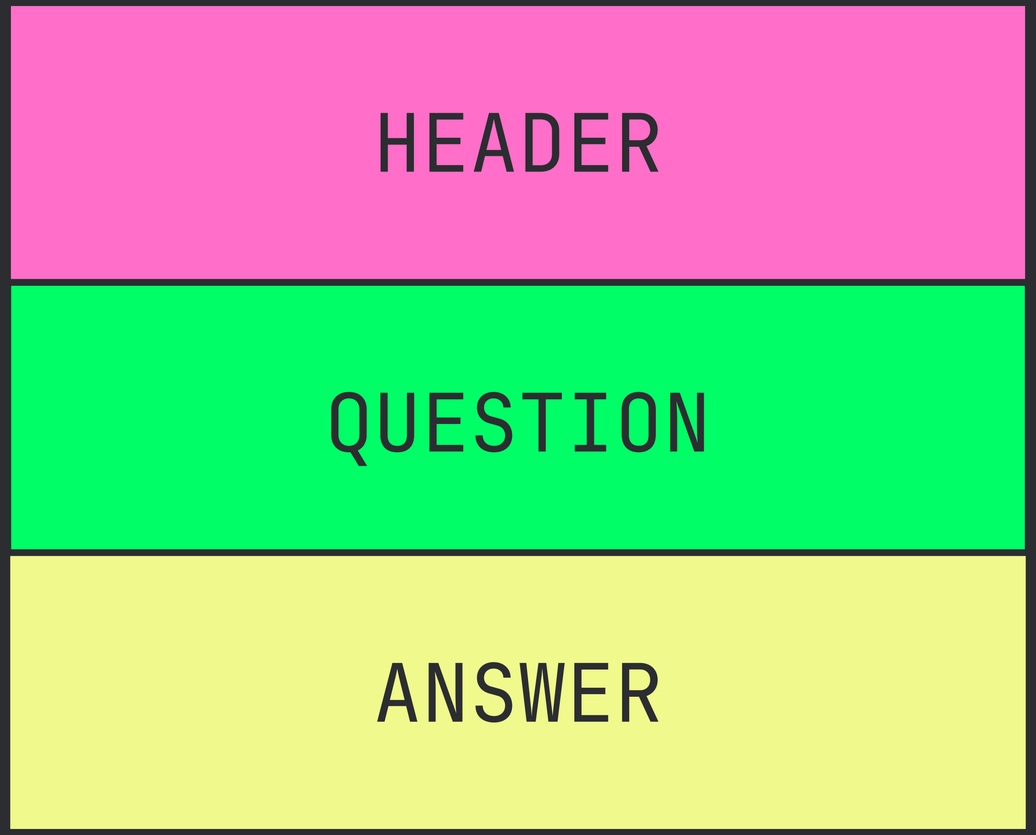
response

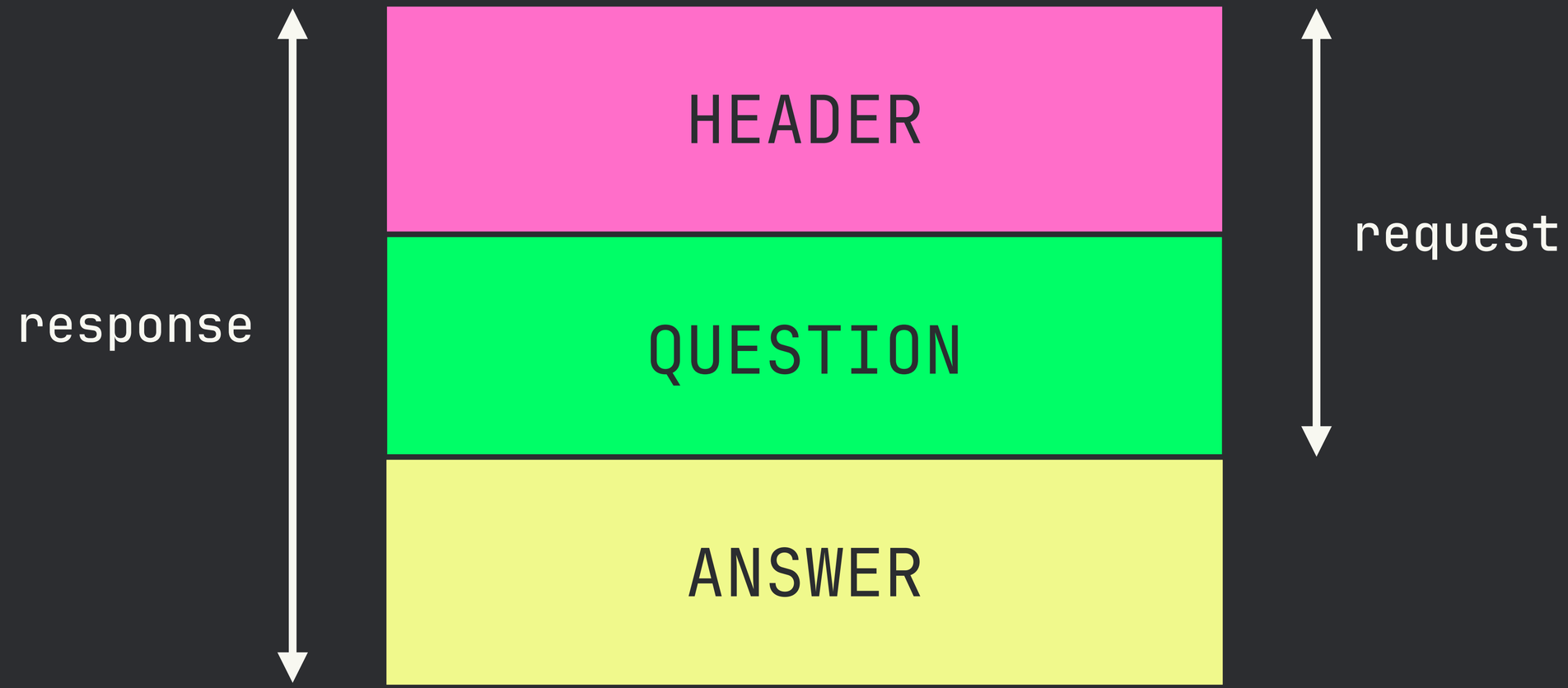
request

HEADER

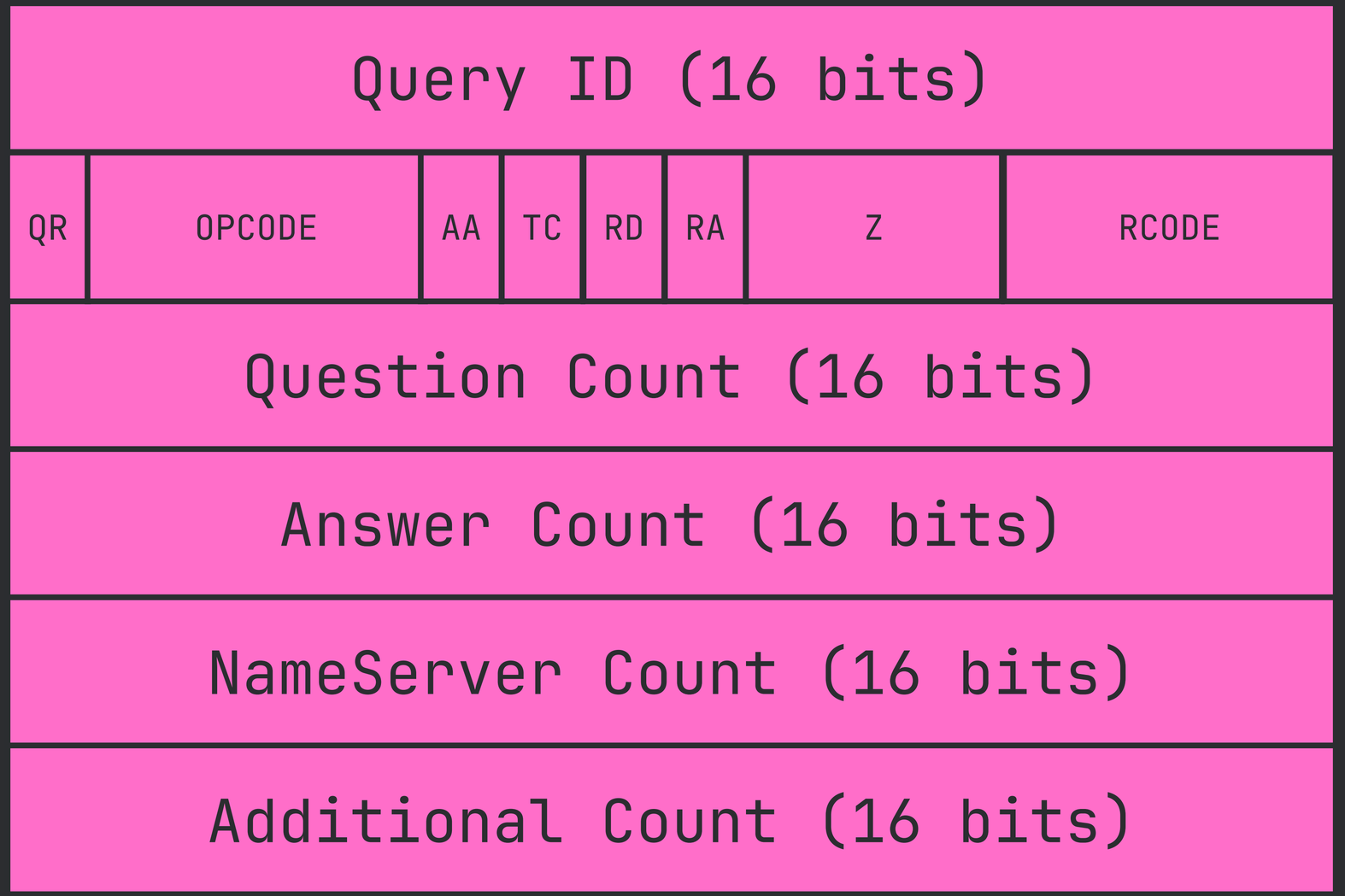
QUESTION

ANSWER


HEADER
QUESTION
ANSWER


HEADER
QUESTION
ANSWER
response
request

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

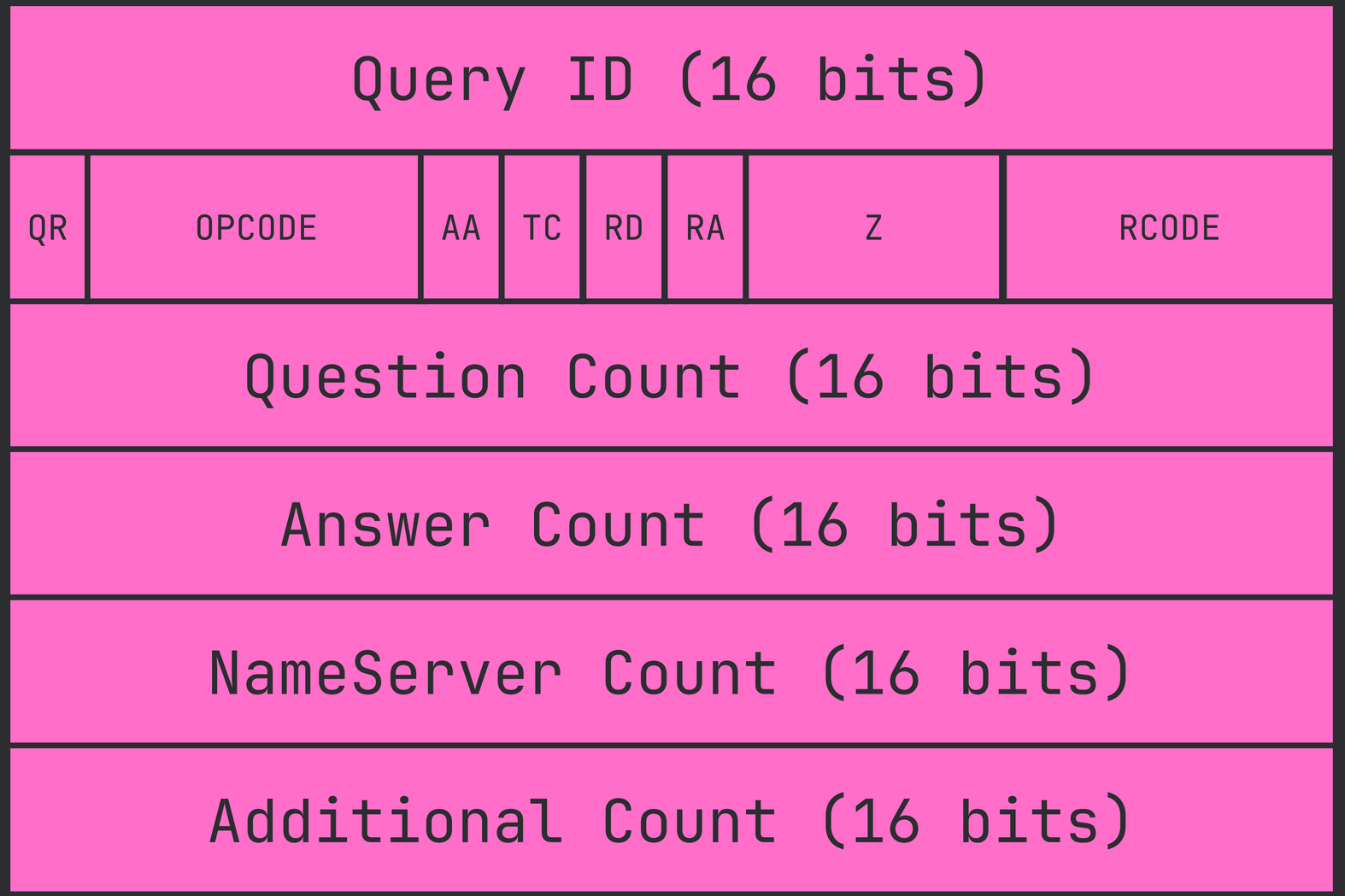
Z Value

→ 3 bits reserved for future use

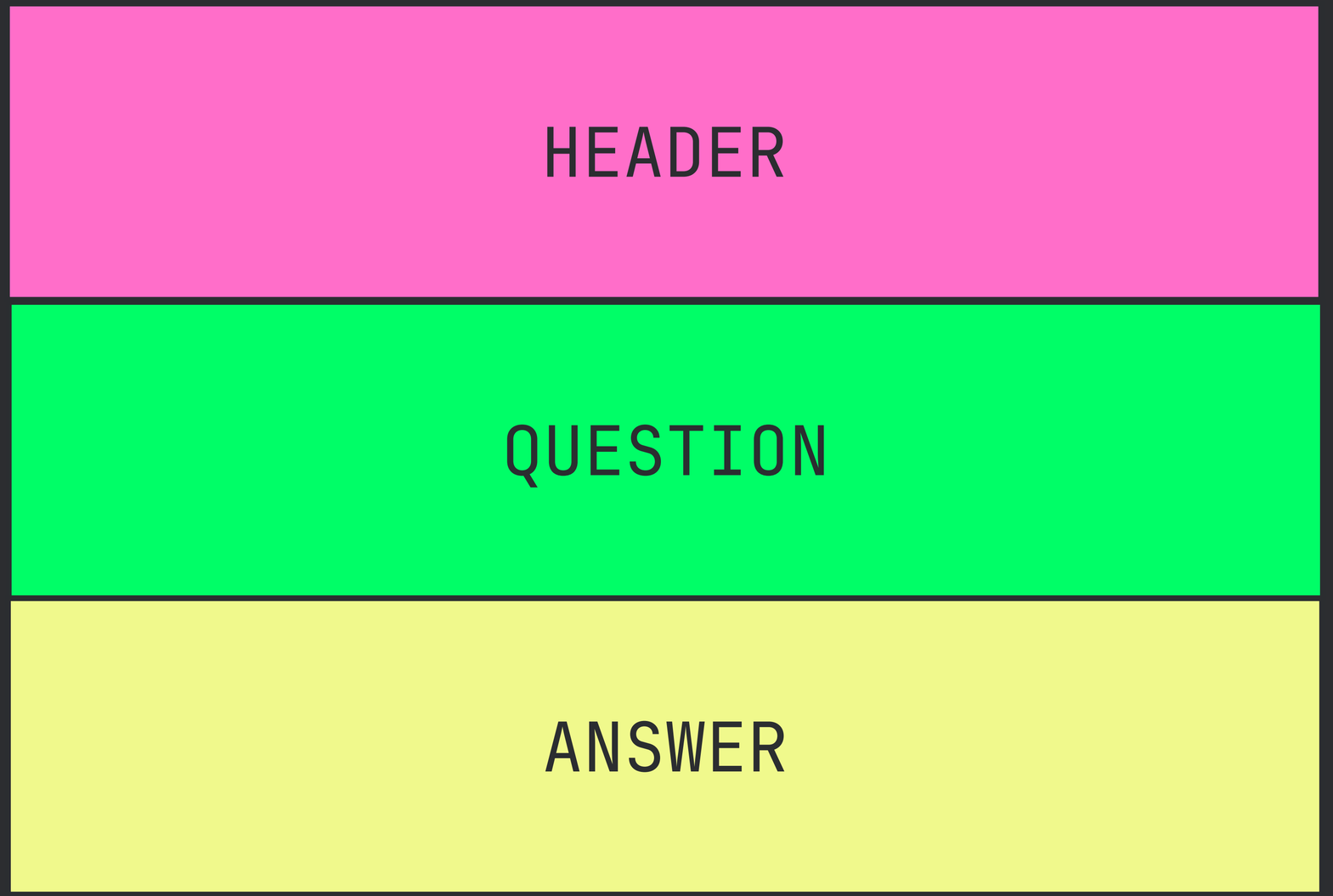
→ according to RFC - “must be 0”

→ 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

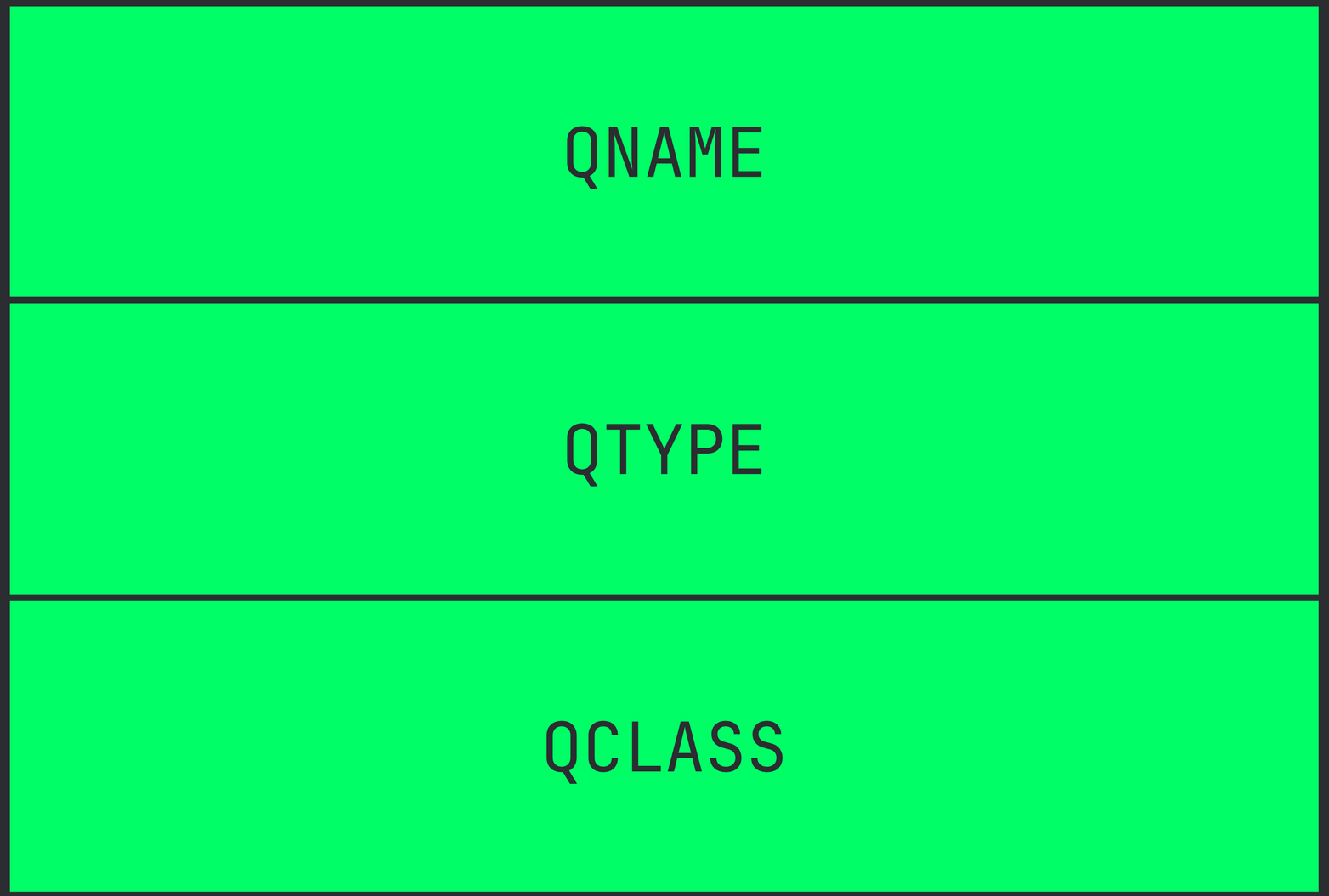

HEADER
QUESTION
ANSWER

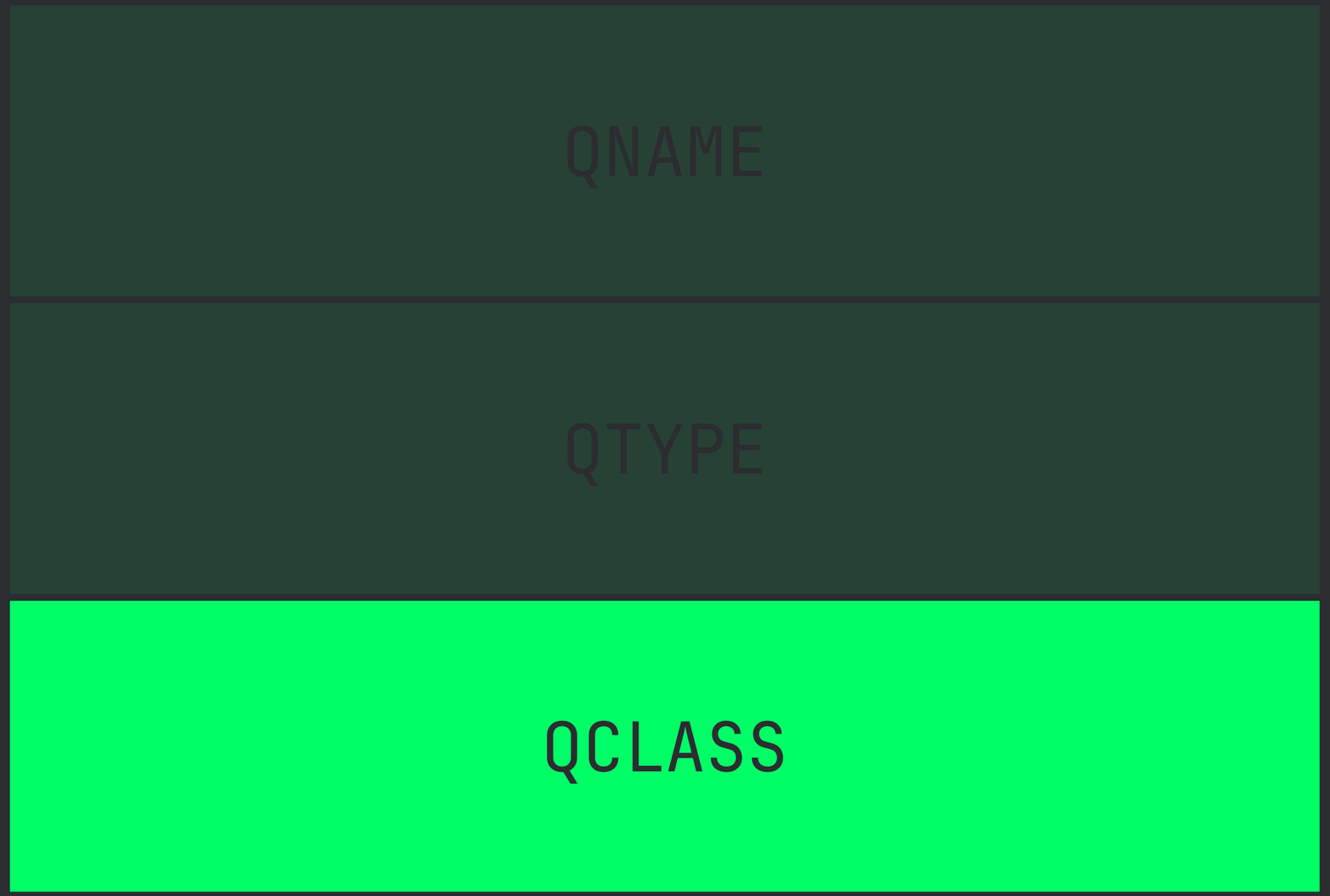
HEADER

QUESTION

ANSWER

QUESTION


QNAME
QTYPE
QCLASS

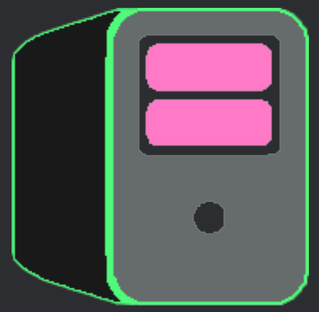

QNAME
QTYPE
QCLASS

QCLASS

→ 16 bit int, 0 - 65535 options

→ it’s “always” IN(ternet) (1)

→ most middlebox ignore (test!)



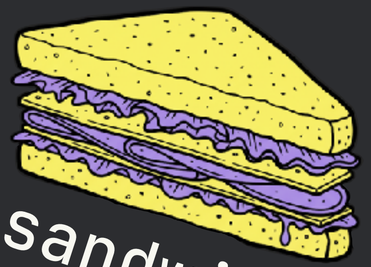
signal

data

signal

data

DNS

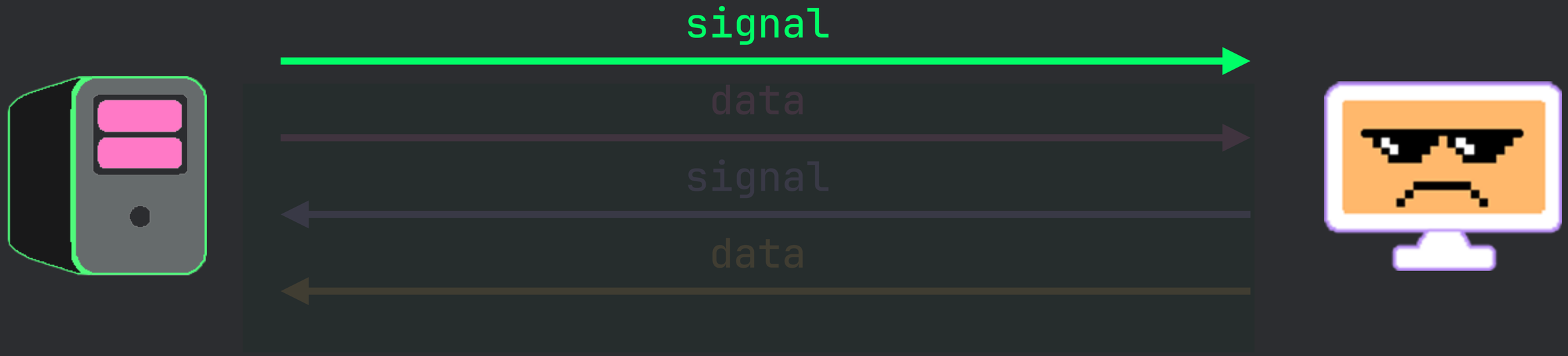


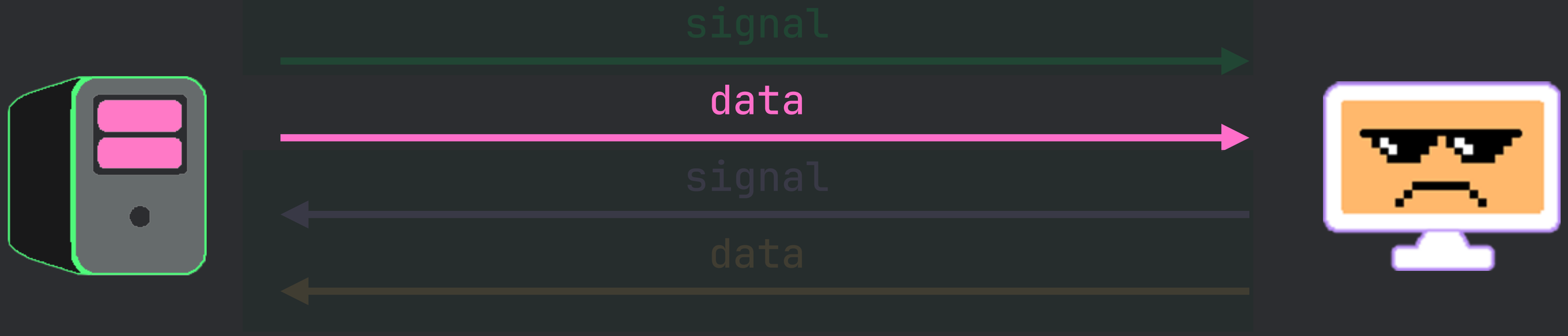
sandwich

→ Z gives us 8 options

→ qClass gives us 65,536

→ if we combine, < 0.5M


signal
data
signal
data



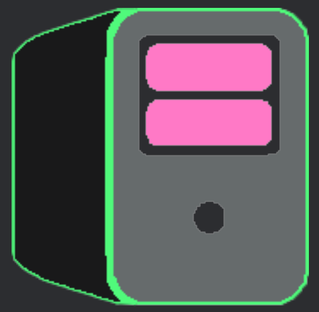
→ Limited, but options

→ Data carried in ANSWER

→ NULL Records

→ CNAME/NS/MX Records

→ TXT Records



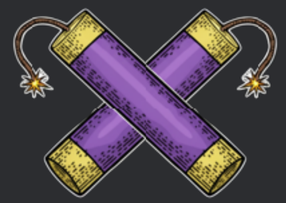
signal

data

signal

data

TXT



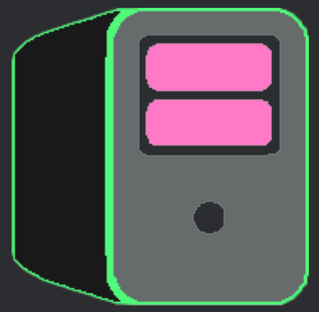
records

→ 255 chars per string

→ 1+ strings/record (1kB+)

→ direct encoding - blends in

→ iodine, dns2tcp, dnscat2



signal

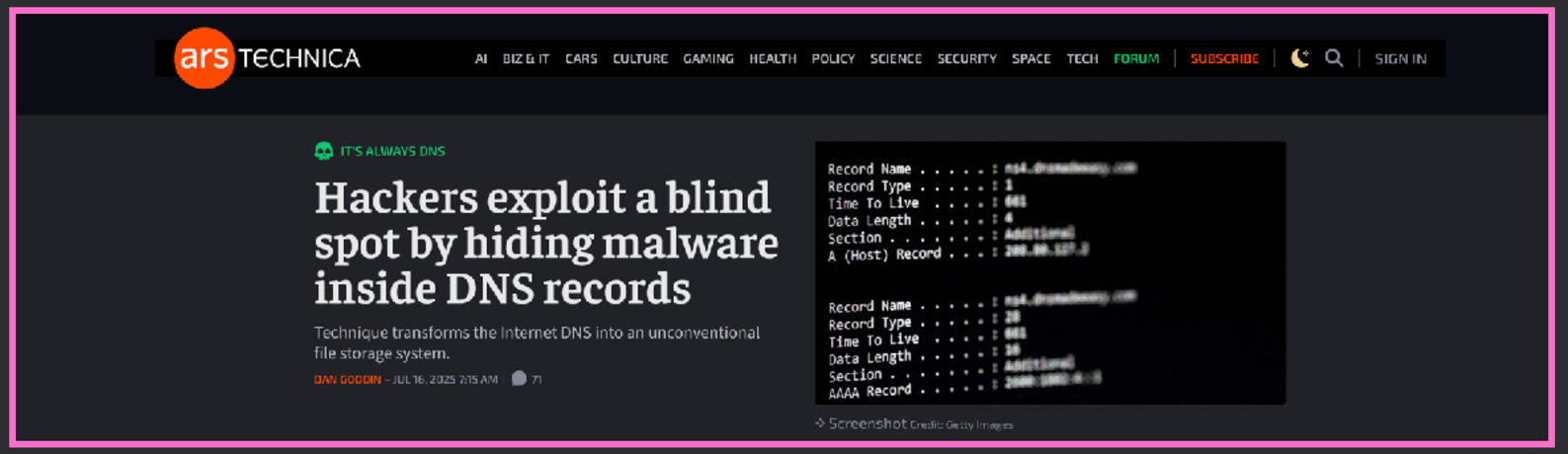
data

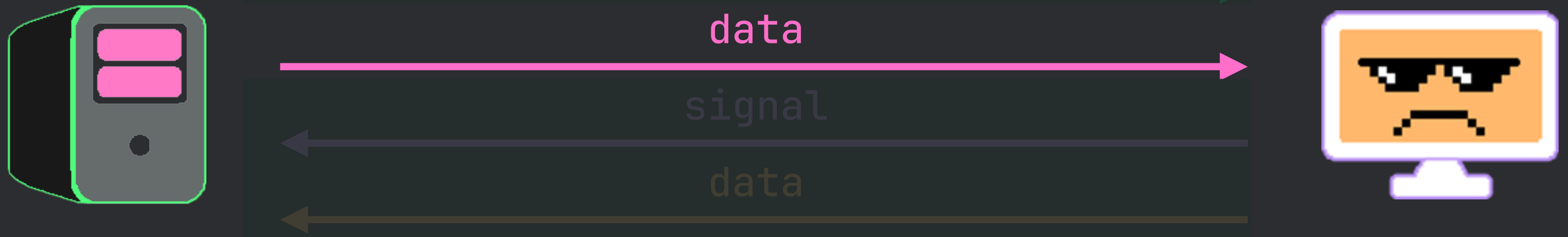
signal

data

joker

screenmate

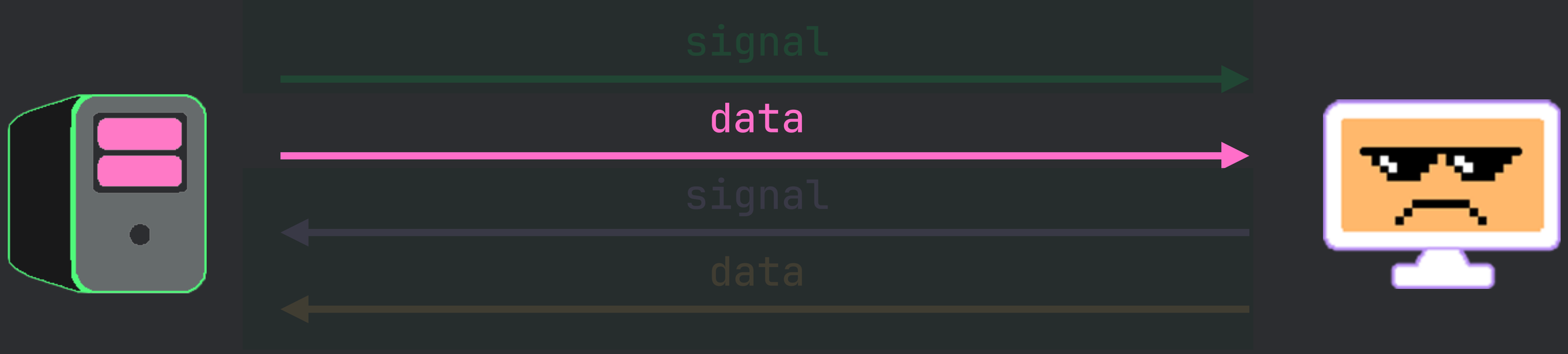


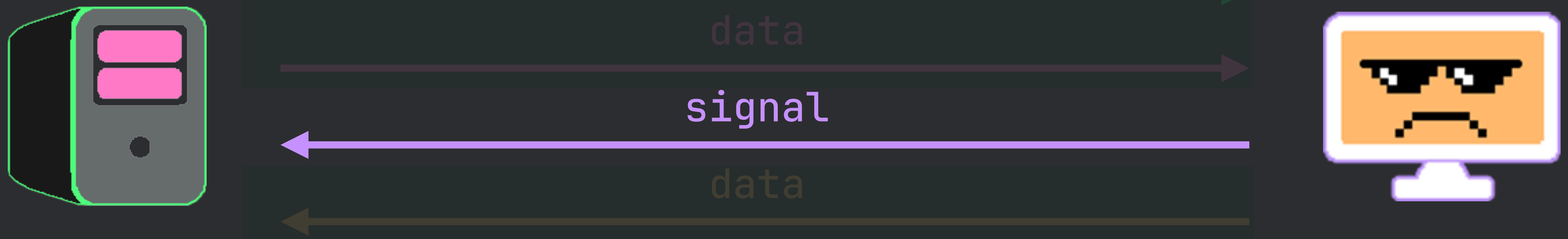


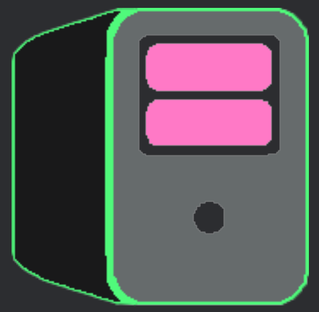
→ Limited, but options

→ TXT Records prob best

→ 1000s of TXT sus af


signal
data
signal
data


signal
data
signal
data



signal

data

signal

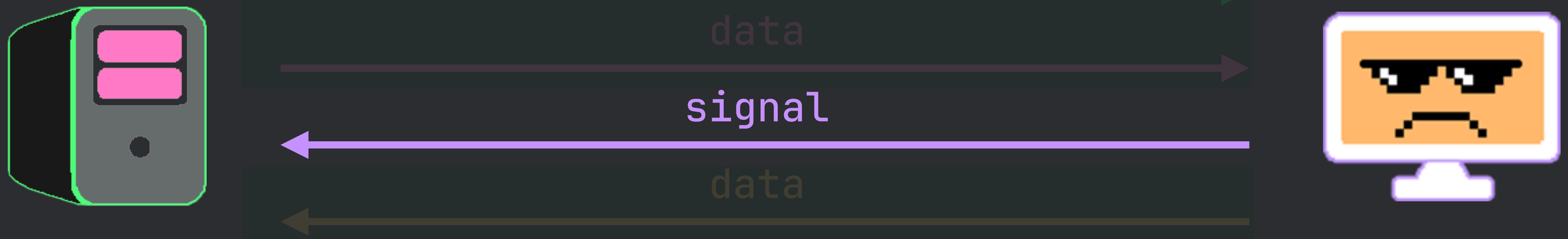
data

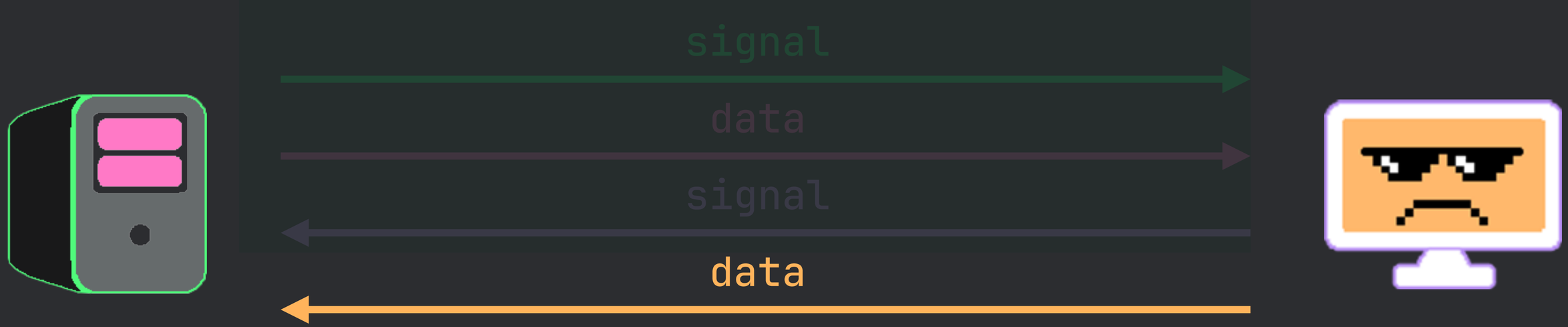
→ Similiar to SERVER → AGENT

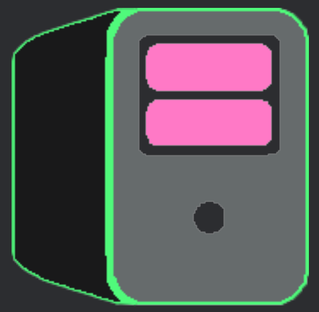
→ Just lacking ANSWER

→ We have Z + qClass

→ Enough to work with


signal
data
signal
data


signal
data
signal
data



signal

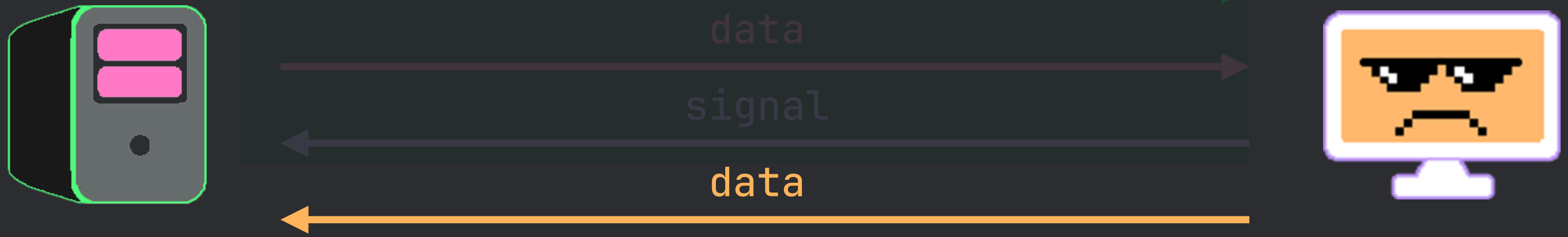
data

signal

data

This is where we encounter

real constraints…



→ conventional MO is using encoded subdomains

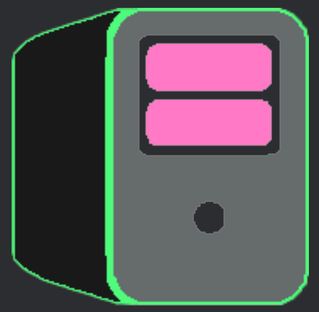
→ run `whoami` → **longtim**

→ HEX → 6c6f6e6774696d

→ 6c6f6e6774696d.derpistan.com

→ max 63 chars, 30/1 chars output

→ large data chunked over 1+ subdomains

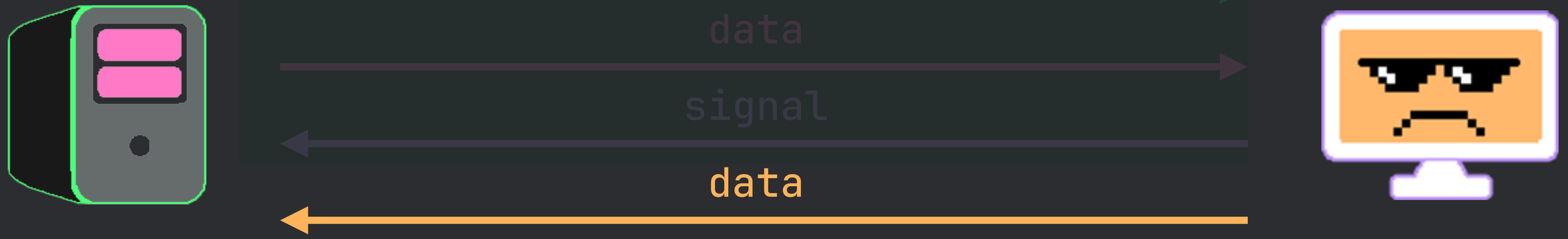


signal

data

signal

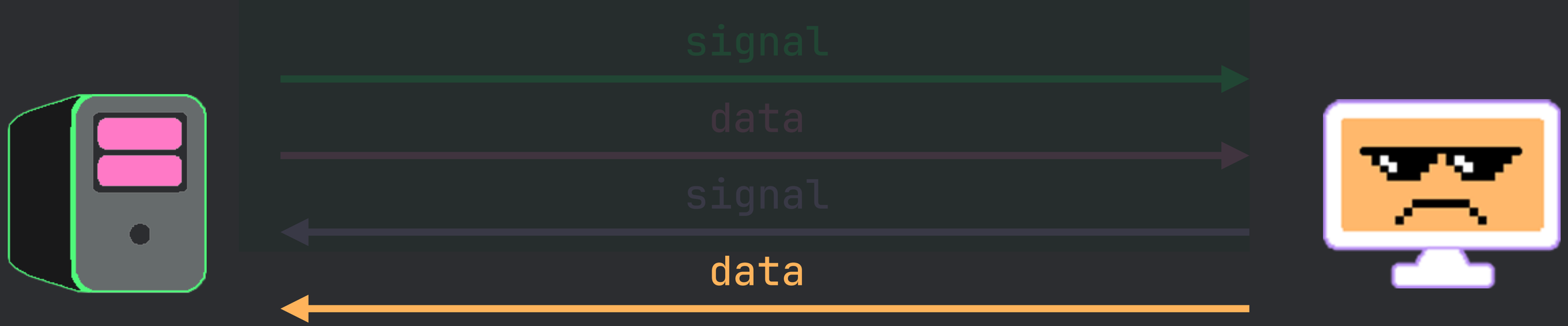
data



→ largest known domains max 100s

→ this technique generates 100s, 1000s, 10ks

→ unknown domain + 1000s of garbled subdomains…



→ encoded subdomains: use SPARINGLY

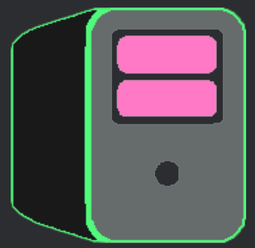
→ Joff Thyer: Query ID (16 bits)

DNS as Covert Channel

- DNS not designed for high-bandwidth data
- Use as state machine (signals)
- Use for: check-ins, enumerate, profile
- RISK + REWARD → Upgrade to HTTPS

https as covert channel

C2 Server



C2 Agent

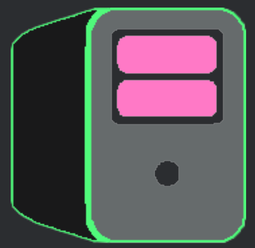
C2 Server

→ keep 4-component model in mind

→ model HTTPS using R+R cycle

C2 Agent

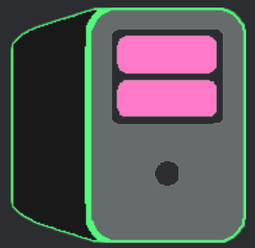
C2 Server



C2 Agent

REQUEST

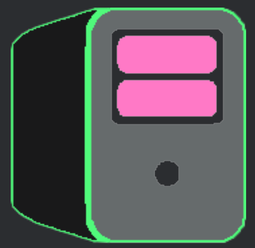
C2 Server



C2 Agent

REQUEST - Heartbeat/Check-in

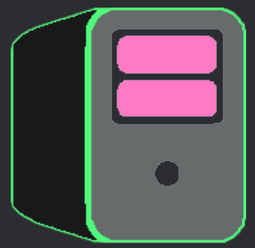
C2 Server



C2 Agent

Hit /ep using method GET (SIGNAL)

C2 Server

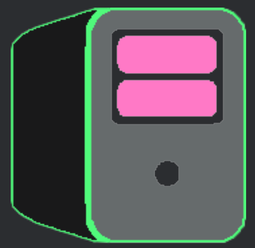


C2 Agent

Hit /ep using method GET (SIGNAL)

RESPONSE

C2 Server

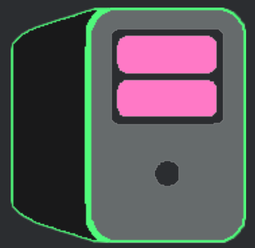


C2 Agent

Hit /ep using method GET (SIGNAL)

No job (FALSE) or job (TRUE) | DATA

C2 Server



C2 Agent

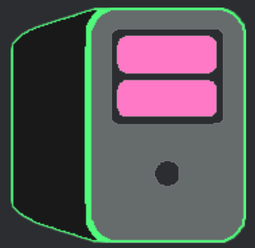
Hit /ep using method GET (SIGNAL)

No job (FALSE) or job (TRUE) | DATA

C2 Server

C2 Agent

IF TRUE → REQUEST



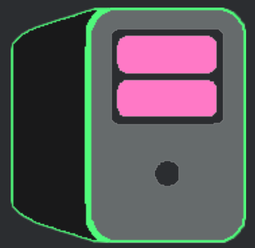
Hit /ep using method GET (SIGNAL)

No job (FALSE) or job (TRUE) | DATA

C2 Server

C2 Agent

POST body with encrypted JSON blob

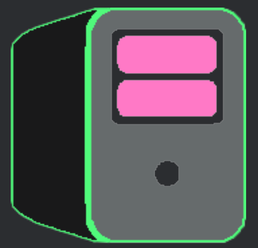


Hit /ep using method GET (SIGNAL)

No job (FALSE) or job (TRUE) | DATA

C2 Server

C2 Agent



POST body with encrypted JSON blob

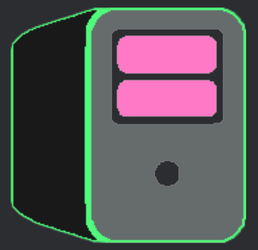
200 OK || 204 No Content

Hit /ep using method GET (SIGNAL)

No job (FALSE) or job (TRUE) | DATA

C2 Server

C2 Agent

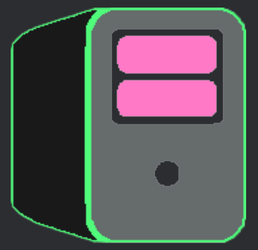


POST body with encrypted JSON blob

200 OK || 204 No Content

→ SLEEP = Delay (k) + Jitter (var) ←

C2 Server



Hit /ep using method GET (SIGNAL)

No job (FALSE) or job (TRUE) | DATA

POST body with encrypted JSON blob

200 OK || 204 No Content

C2 Agent

→ DISCONNECT (beacon) ←

→ SLEEP = Delay (k) + Jitter (var) ←

HTTPS as Covert Channel

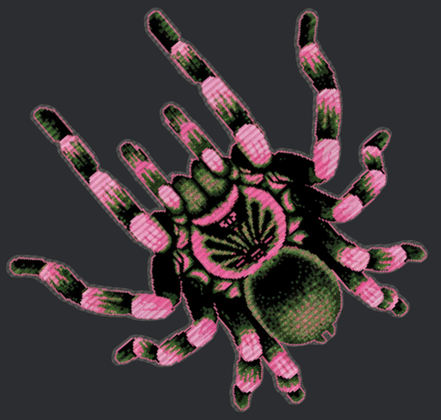
- Designed for high bandwidth (expected)
- Designed to be encrypted (expected)
- Ubiquitous + high-volume (like DNS)
- Signal/Data in both directions → Endless

it seems like it can do
everything DNS can, and more.

the question becomes…

DNS is STEALTHIER

- Less expected, less inspected
- Ability to resolve locally
- HTTPS headache - certificate and domain



MODERN C2 and DATA EXFILTRATION

Kyle Avery

BLACK HILLS | Information Security

WEBCA

1:38.

there’s more to covert channels…

→ Infrastructure Architecture

→ Timing Characteristics

→ Data Jitter/Padding

→ Data Encoding/Obfuscation

→ 3rd Party Service Integration

→ Authentication + Access Control

→ etc…

tomorrow's workshop

→ create DNS/HTTPS hybrid covert channel in Go

→ COMPLETE OVERHAUL from Version 01

→ Specifics AND transcendent value

→ Idiomatic Go Design Architecture/Patterns

www.faanross.com

thank you!


NOW, ARE THERE
ANY QUESTIONS?