IpMorph :

Unification of OS fingerprinting defeating

or, how to defeat common OSFP tools.

Guillaume PRIGENT
Florian VICHOT
DIATEAM - Brest

IpMorph : “How to defeat common OSFP tools”

Context

Reason for creating IpMorph :

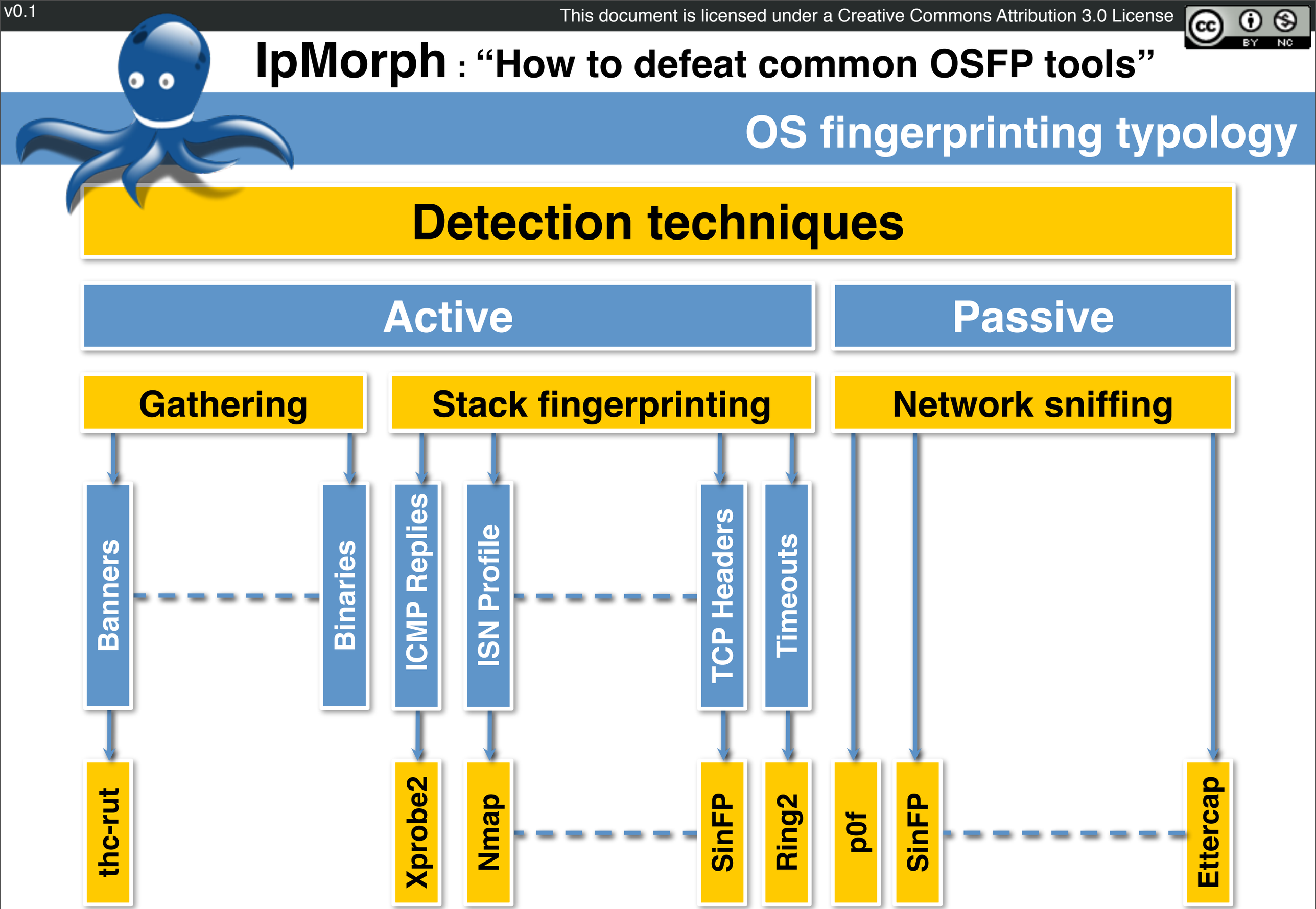
- *Hynesim Project: We needed a way to disguise low-interaction guests (OpenVZ) as different, non-unix OSes.*
- *We already had some software components from previous projects*
- *It seemed fun (at the time)*

Guiding principles :

- *Complete software, not just proof-of-concept*
- *Unification of spoofing mechanisms*
- *No network disruptions*

IpMorph : “How to defeat common OSFP tools”

OS fingerprinting typology



IpMorph : “How to defeat common OSFP tools”

OSFP Timeline

IpMorph : “How to defeat common OSFP tools”

OSFP Timeline

IpMorph : “How to defeat common OSFP tools”

Fingerprinting principles

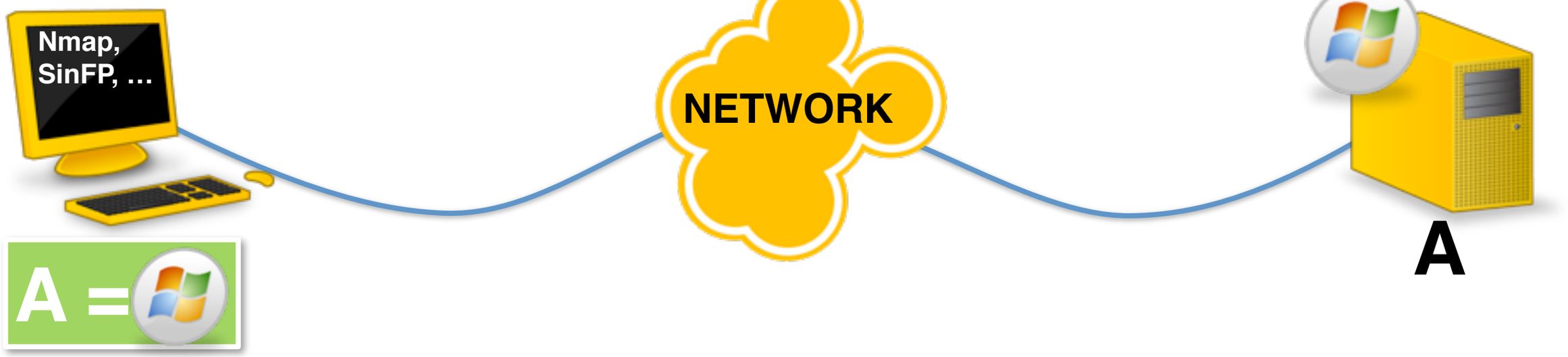
Active stack fingerprinting

Advantages	Drawbacks
Quick, precise	«Noisy», recognisable

IpMorph : “How to defeat common OSFP tools”

Fingerprinting principles

Active stack fingerprinting

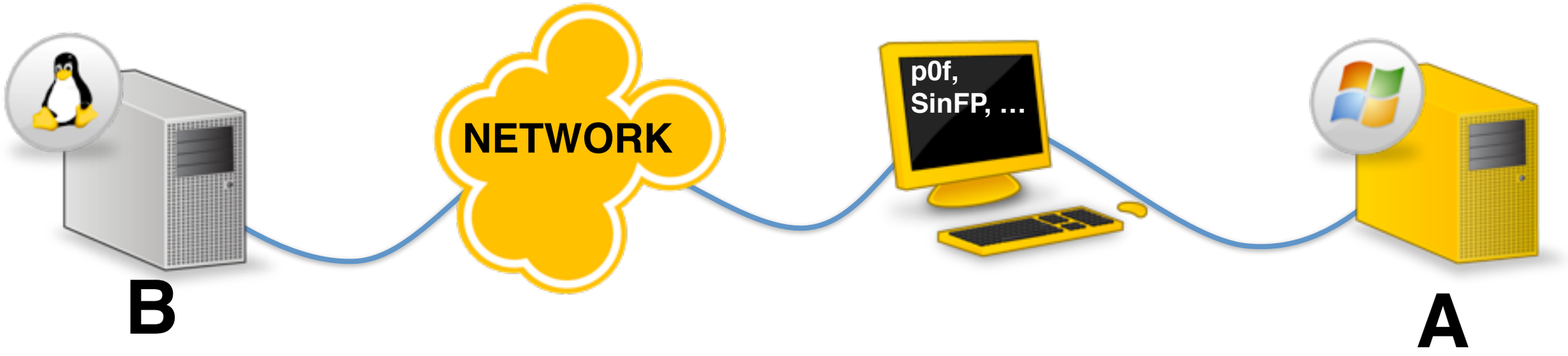


Advantages	Drawbacks
Quick, precise	«Noisy», recognisable

IpMorph : “How to defeat common OSFP tools”

Fingerprinting principles

Passive stack fingerprinting

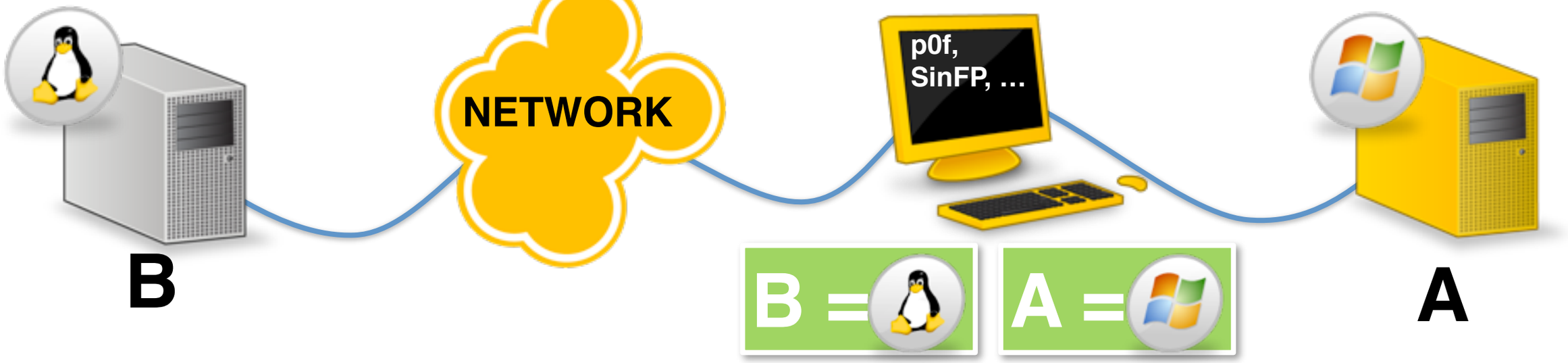


Advantages	Drawbacks
stealthy	slow, vague

IpMorph : “How to defeat common OSFP tools”

Fingerprinting principles

Passive stack fingerprinting

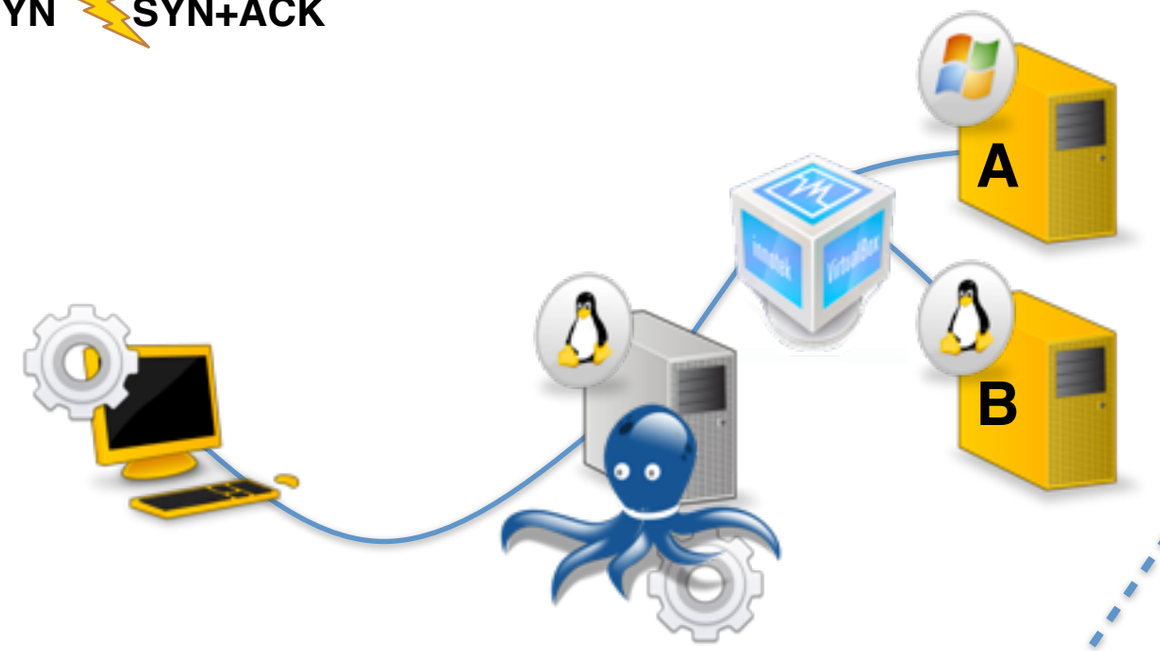


Advantages	Drawbacks
stealthy	slow, vague

IpMorph : “How to defeat common OSFP tools”

IpMorph use-cases

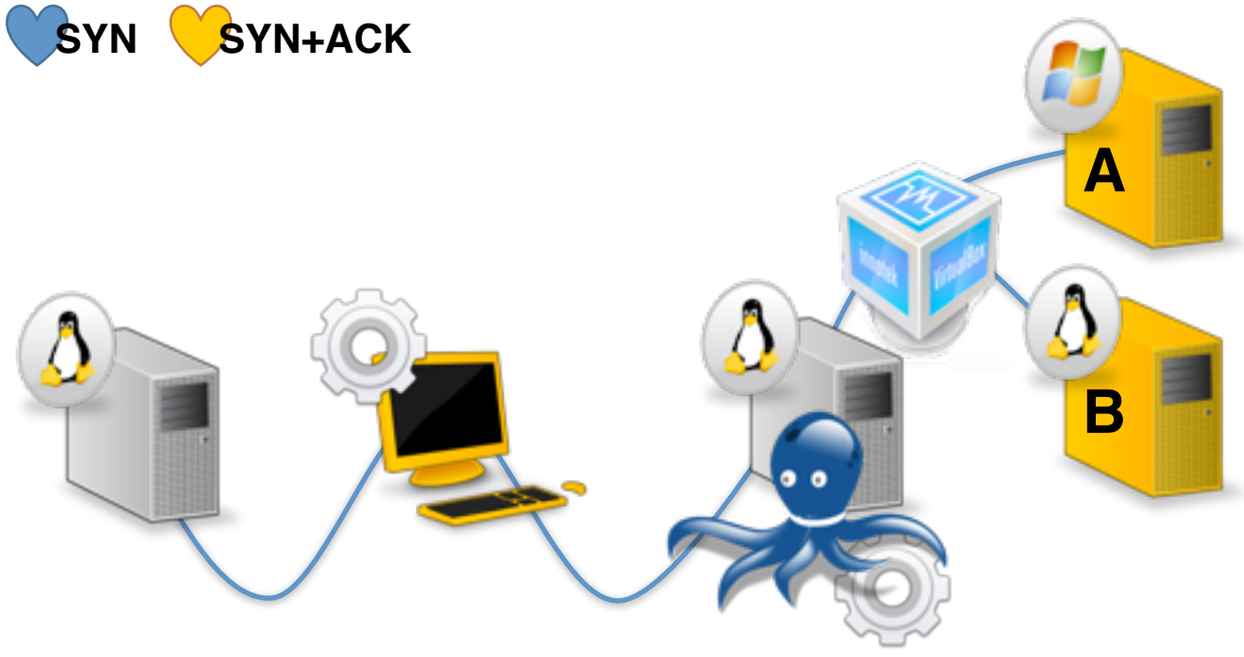
SYN SYN+ACK



Active OSFP + Virtual Machines

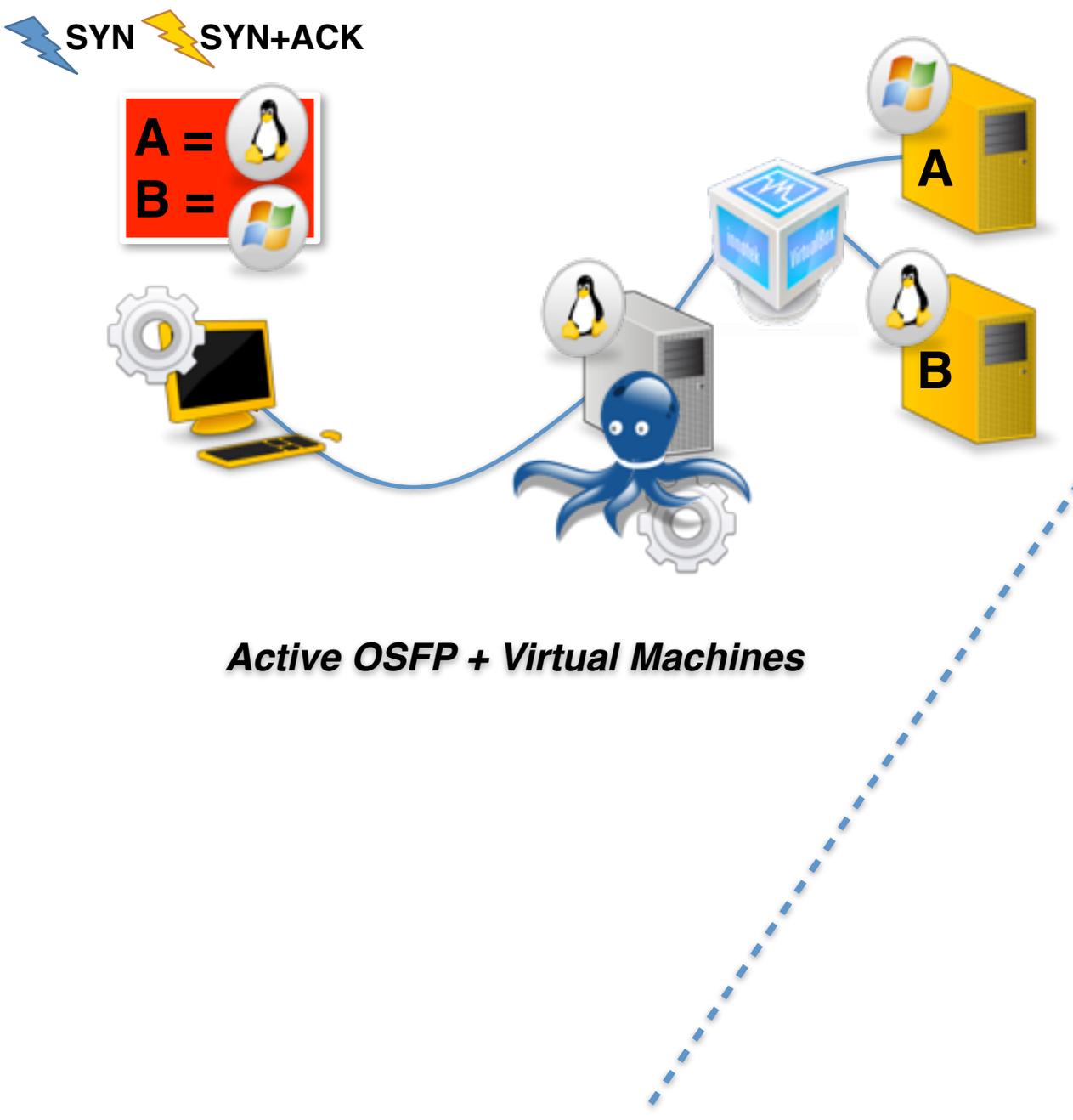
Passive OSFP + Virtual Machines

SYN SYN+ACK



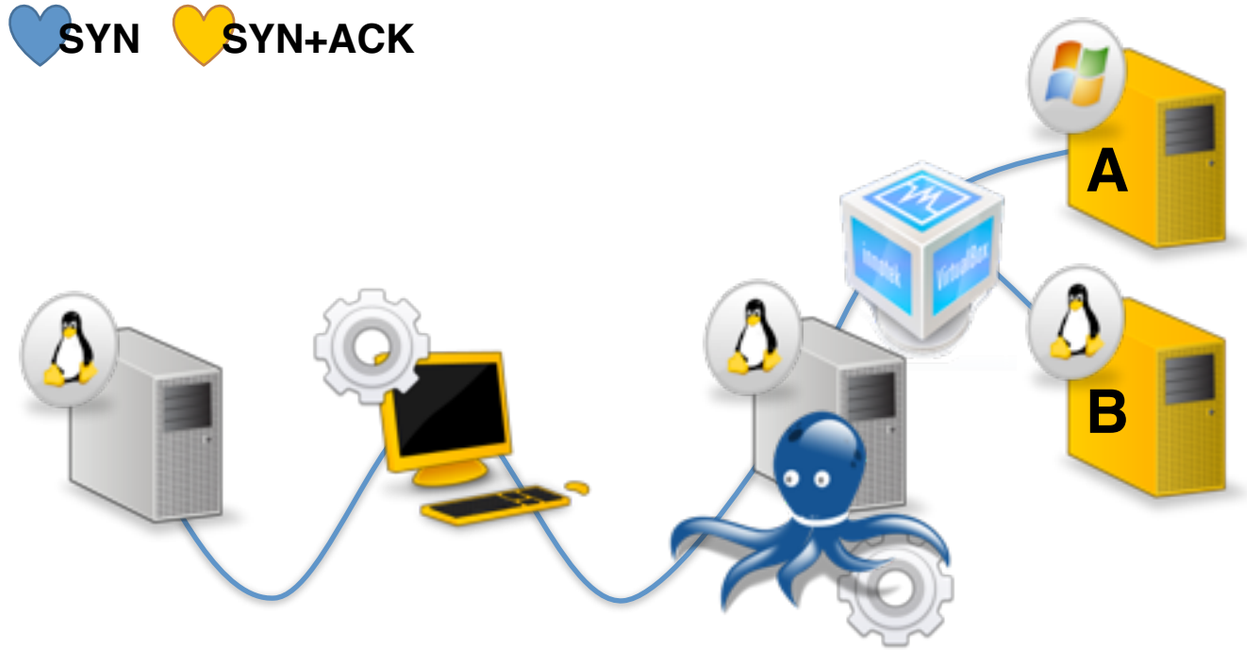
IpMorph : “How to defeat common OSFP tools”

IpMorph use-cases



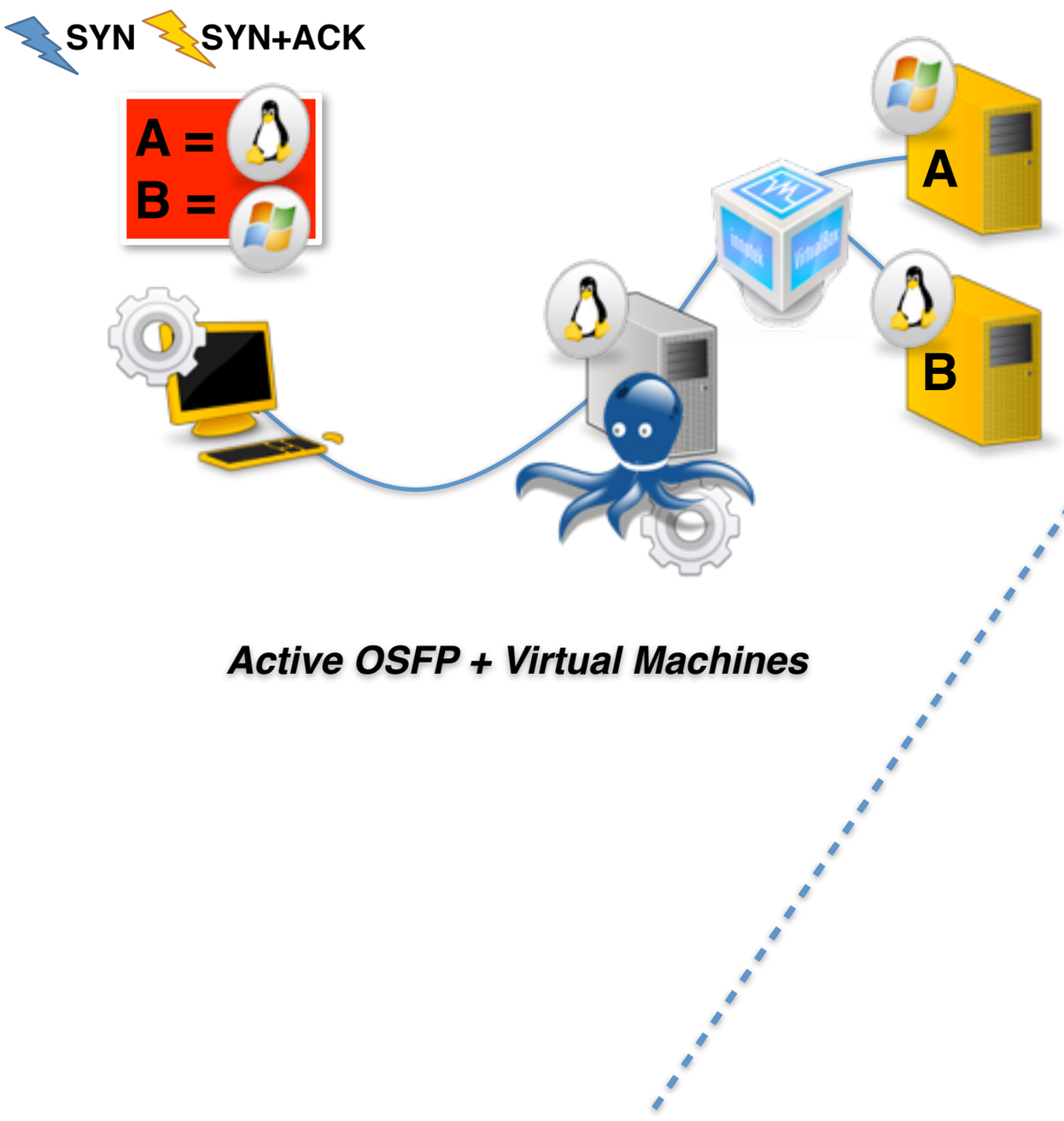
Active OSFP + Virtual Machines

Passive OSFP + Virtual Machines



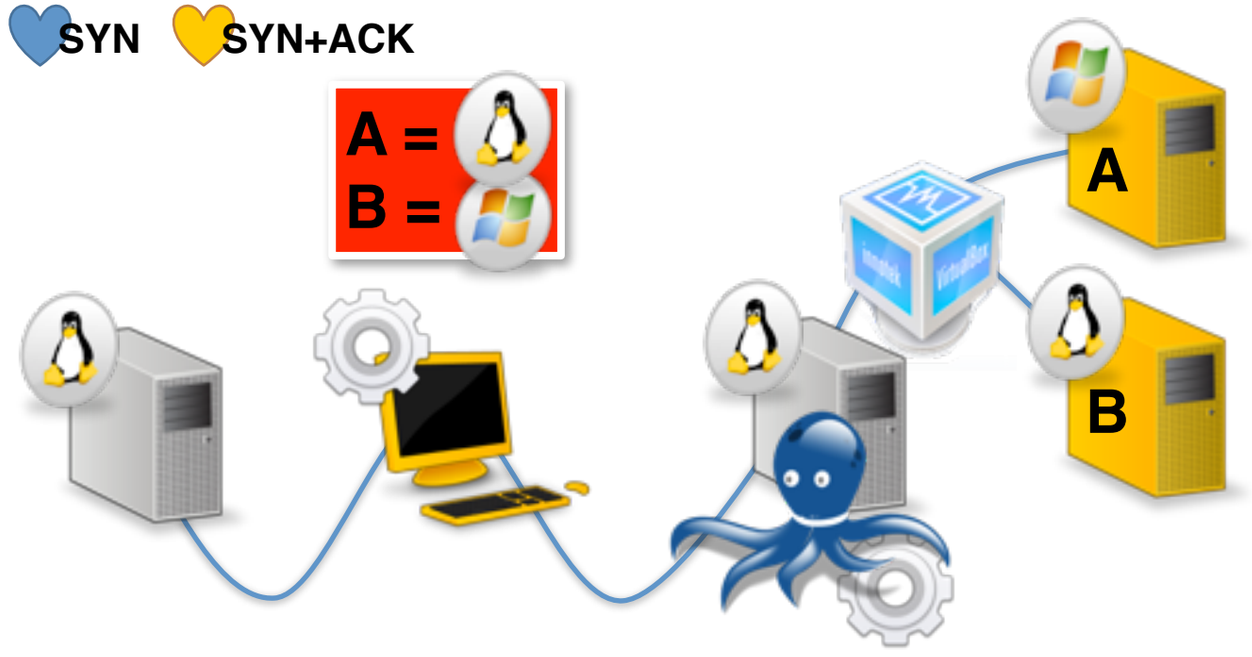
IpMorph : “How to defeat common OSFP tools”

IpMorph use-cases



Active OSFP + Virtual Machines

Passive OSFP + Virtual Machines



IpMorph : “How to defeat common OSFP tools”

Spoofing state of the art

- **Filtering**
  - Stealth patch : **Unmaintained as of 2002, GNU/Linux kernel 2.2-2.4**
  - Blackhole : **FreeBSD, kernel options**
  - IPlog : **Unmaintained as of 2001, *BSD**
  - Packet filter : **OpenBSD**
- **Host TCP/IP stack tweaking**
  - Ip Personality
  - Fingerprint Fucker
  - Fingerprint scrubber
  - OSfuscate
- **Host TCP/IP stack replacement (proxy behaviour)**
  - Honeyd
  - Packet purgatory / Morph

IpMorph : “How to defeat common OSFP tools”

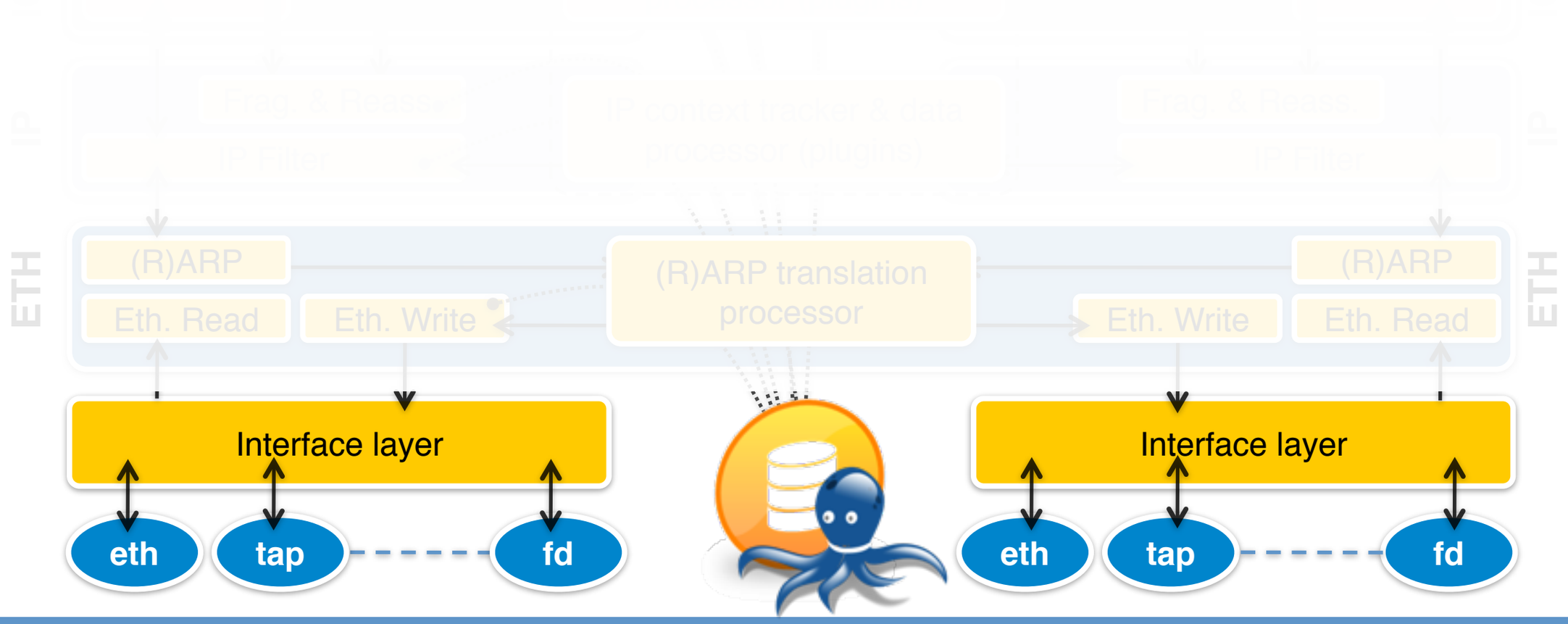
Software bricks

- **Coded in C++**
- **Userland application**
- **Tools:**
  - IpMorph (Core)
  - IpMorph Controller
  - IpMorph Personality Manager
  - IpView (IpMorph GUI)
- **Portability :**
  - GNU/Linux
  - *BSD, Mac OS
- **GPLv3 License**

IpMorph : “How to defeat common OSFP tools”

General architecture

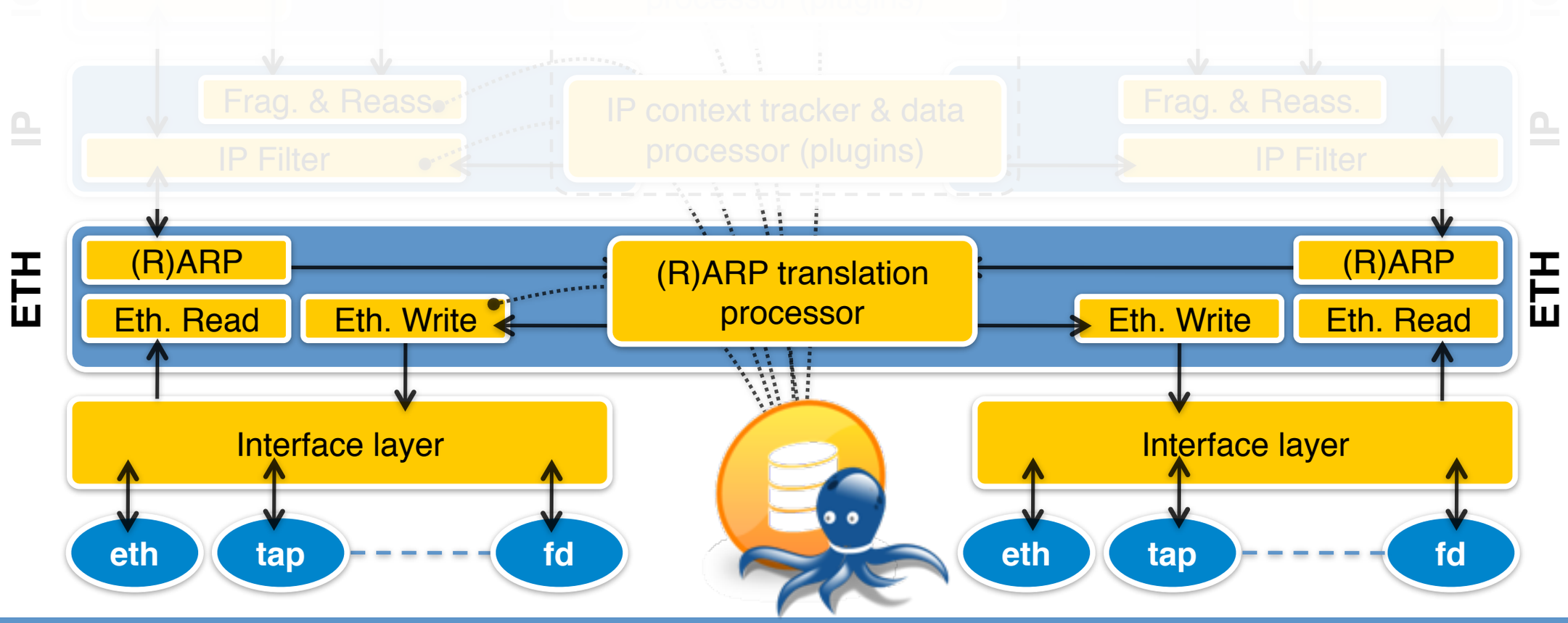
Exposed IP stack **Context queue** **Protected IP stack**



IpMorph : “How to defeat common OSFP tools”

General architecture

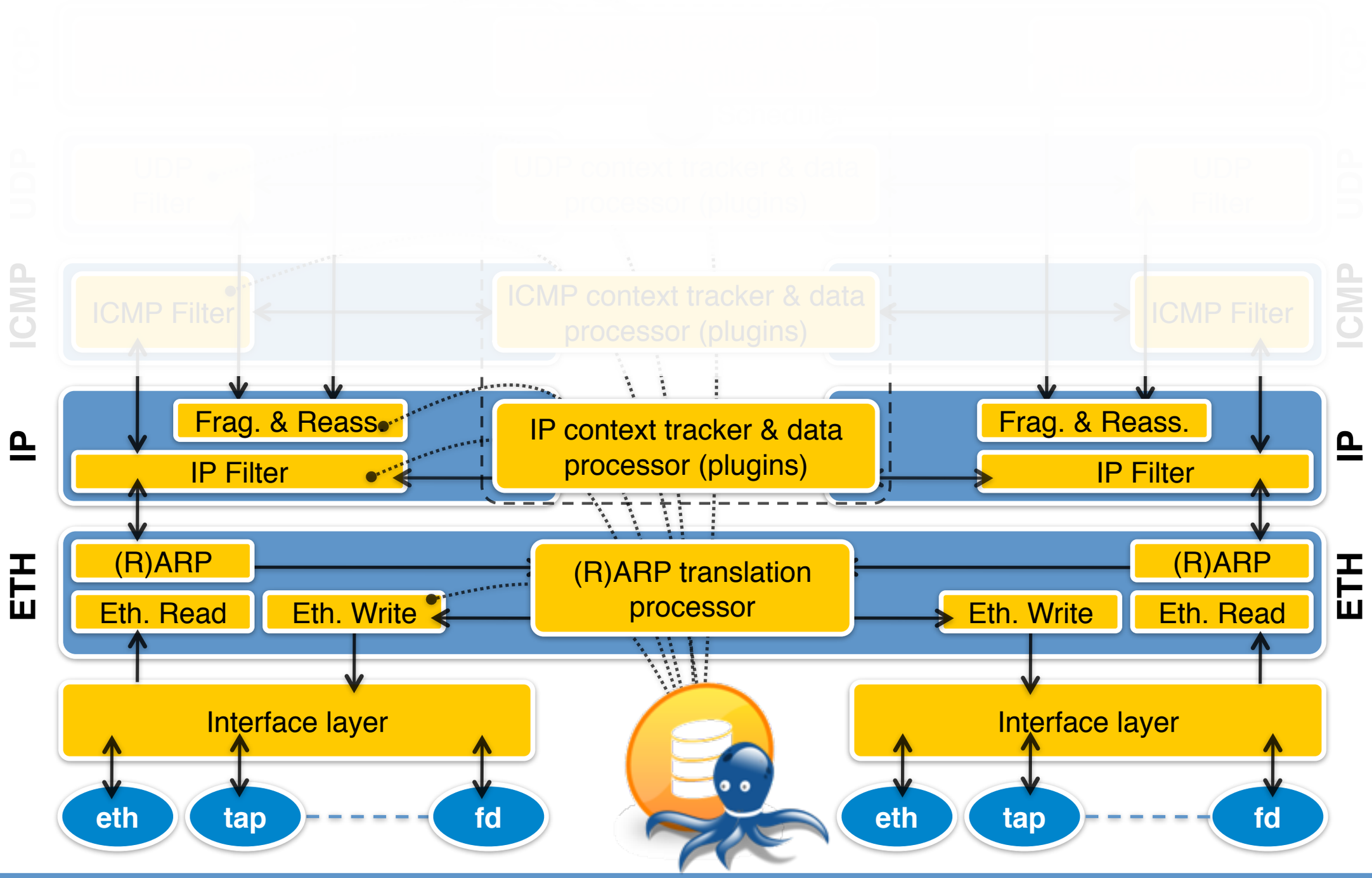
Exposed IP stack **Context queue** **Protected IP stack**



IpMorph : “How to defeat common OSFP tools”

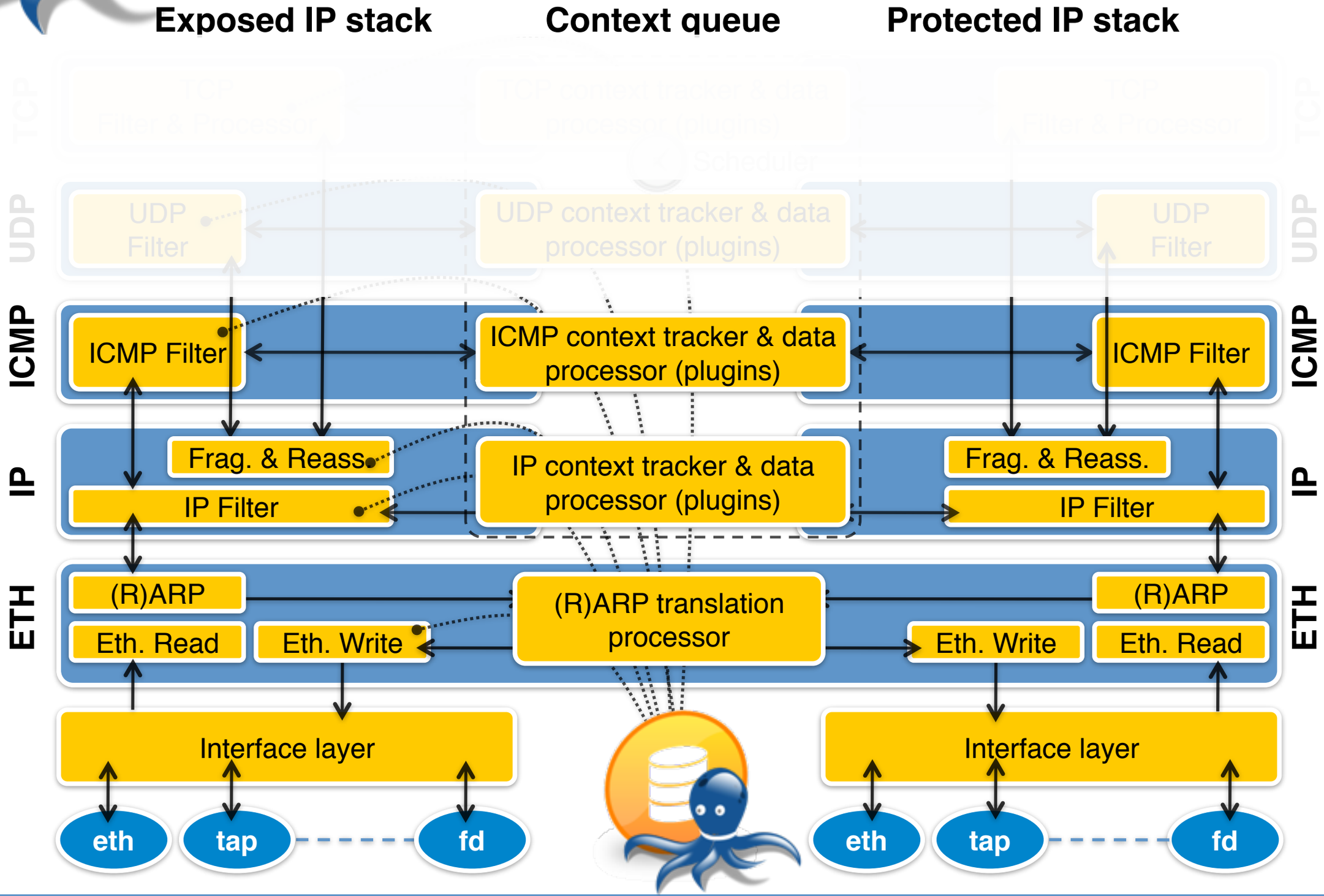
General architecture

Exposed IP stack **Context queue** **Protected IP stack**



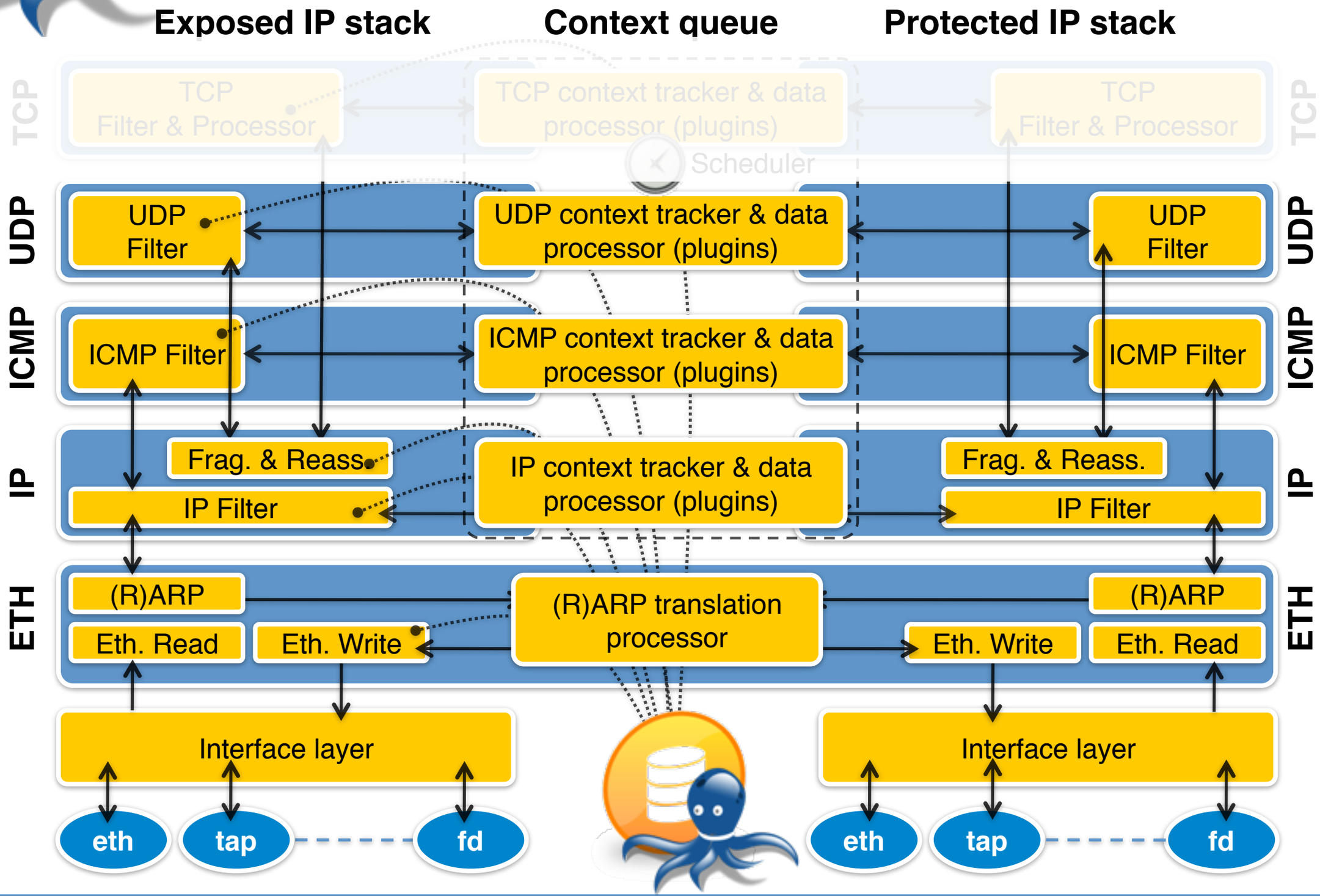
IpMorph : “How to defeat common OSFP tools”

General architecture



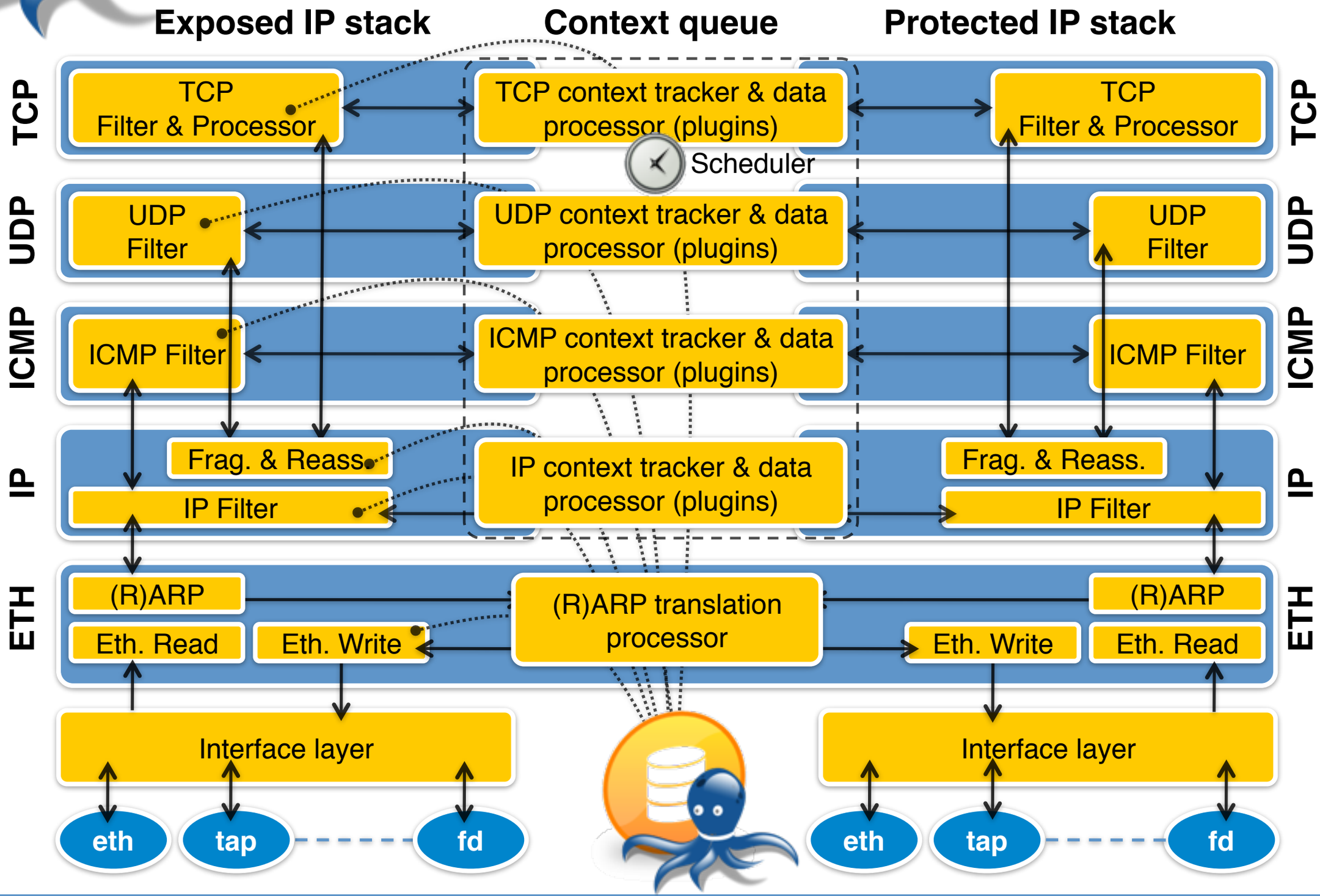
IpMorph : “How to defeat common OSFP tools”

General architecture



IpMorph : “How to defeat common OSFP tools”

General architecture



IpMorph : "How to defeat common OSFP tools"

Nmap : Signature Format

SP : TCP ISN Predictability

GCD : TCP ISN Greatest Common Divisor

ISR : TCP ISN counter Rate

TI : TCP IP ID sequence generation algorithm

II : ICMP IP ID sequence generation algorithm

SS : Shared IP ID sequence Boolean

TS : TCP timestamp option algorithm

```
Fingerprint FreeBSD 7.0-CURRENT
Class FreeBSD | FreeBSD | 7.X | general purpose
SEQ(SP=101-10D%GCD=<7%ISR=108-112%TI=RD%II=RI%TS=20|21|22)
OPS(O1=M5B4NW8NNT11%O2=M578NW8NNT11%O3=M280NW8NNT11%O4=M5B4NW8NNT11%O5=M218NW8NNT11%O6=M109NNT11)
WIN(W1=FFFF%W2=FFFF%W3=FFFF%W4=FFFF%W5=FFFF%W6=FFFF)
ECN(R=Y%DF=Y%T=40%TG=40%W=FFFF%O=M5B4NW8%CC=N%Q=)
T1(R=Y%DF=Y%T=40%TG=40%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=N)
T3(R=Y%DF=Y%T=40%TG=40%W=FFFF%S=O%A=S+%F=AS%O=M109NW8NNT11%RD=0%Q=)
T4(R=Y%DF=Y%T=40%TG=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T5(R=Y%DF=Y%T=40%TG=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=Y%T=40%TG=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T7(R=Y%DF=Y%T=40%TG=40%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)
U1(DF=N%T=40%TG=40%TOS=0%IPL=38%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUL=G%RUD=G)
IE(DFI=S%T=40%TG=40%TOSI=S%CD=S%SI=S%DLI=S)

...
```

O1-06: TCP Options (ordering & values)

W1-W6 : TCP initial win size

DF: IP don't fragment bit

W: TCP initial win size

T: IP initial time-to-live

S: TCP seq. number

TG: IP initial time-to-live guess

A: TCP ack. number

F: TCP Flags

RD: TCP RST data checksum

Q: TCP misc. quirks

RIPCK: Returned probe IP checksum value

RUCK: Returned probe UDP checksum

TOS: IP type of service

IPL: IP total length

UN: Unused port unreach. field nonzero

RIPL: Returned probe IP total length value

RID: Returned probe IP ID value

RUL: Returned probe UDP length

IpMorph : “How to defeat common OSFP tools”

Nmap : Spoofing test RD

SP : TCP ISN Predictability

GCD : TCP ISN Greatest Common Divisor

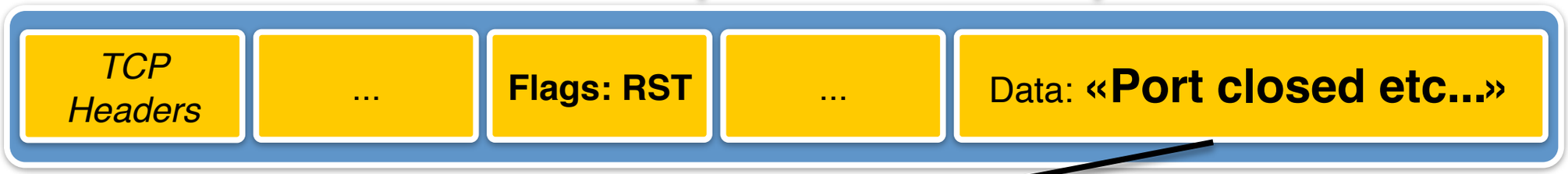
ISR : TCP ISN counter Rate

TI : TCP IP ID sequence generation algorithm

II : ICMP IP ID sequence generation algorithm

RD: TCP RST data checksum

TCP SYN+RST packet on closed port



CRC32(«Port closed etc...») = 0x0af1a1cb

Reverse_CRC32(0x0af1a1cb) = 0x201111b8

IpMorph : “How to defeat common OSFP tools”

Nmap : Spoofing test RD

SP : TCP ISN Predictability

GCD : TCP ISN Greatest Common Divisor

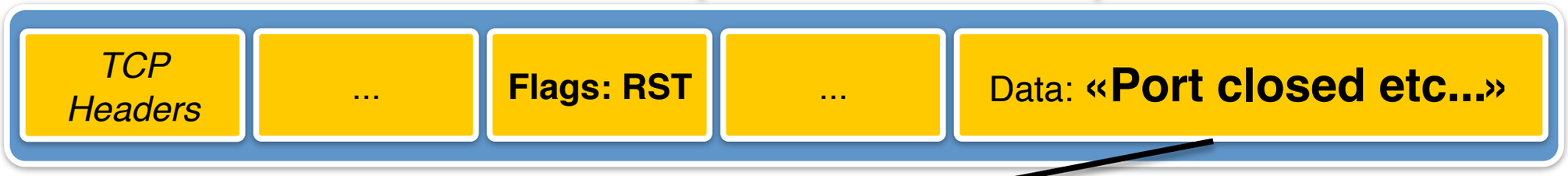
ISR : TCP ISN counter Rate

TI : TCP IP ID sequence generation algorithm

II : ICMP IP ID sequence generation algorithm

RD: TCP RST data checksum

TCP SYN+RST packet on closed port



CRC32(«Port closed etc...») = 0x0af1a1cb

Reverse_CRC32(0x0af1a1cb) = 0x201111b8

IpMorph : "How to defeat common OSFP tools"

Nmap : Spoofing tests TI / II

SP : TCP ISN Predictability

GCD : TCP ISN Greatest Common Divisor

ISR : TCP ISN counter Rate

RD: TCP RST data checksum

TI : TCP IP ID sequence generation algorithm

II : ICMP IP ID sequence generation algorithm

Typical IP packet

Possible algorithms:

- **Always zero**
- **Constant**
- **Constant increment**
- **...**

We keep 2 counters:

- `_lastIpIdTcp`
- `_lastIpIdOther`

IpMorph : “How to defeat common OSFP tools”

Nmap : Spoofing tests SP / GCD / ISR

RD: TCP RST data checksum

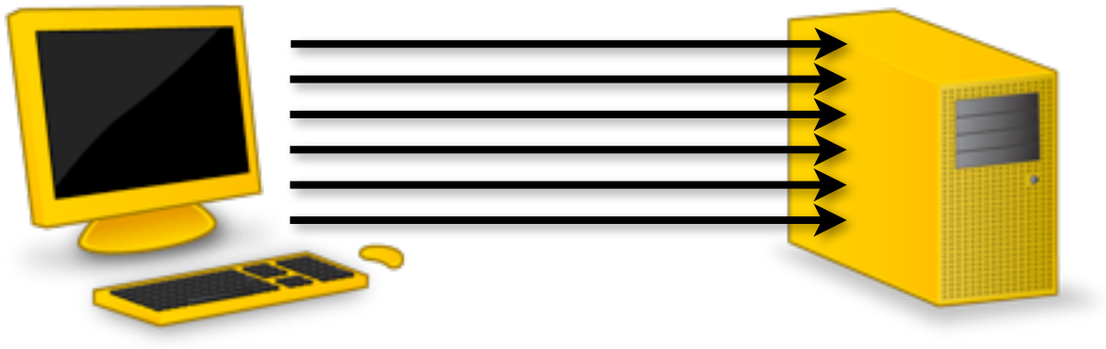
TI : TCP IP ID sequence generation algorithm

II : ICMP IP ID sequence generation algorithm

SP : TCP ISN Predictability

GCD : TCP ISN Greatest Common Divisor

ISR : TCP ISN counter Rate

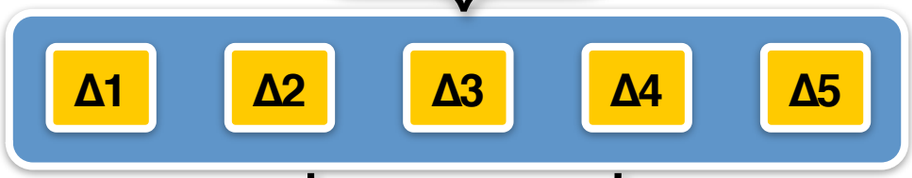


Nmaps send 6 TCP probes, at a 550ms interval.

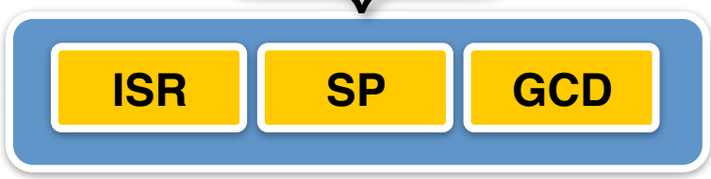
Typical TCP Header

Get all 6 ISNs :

Compute differences :



Compute mean, stddev and GCD :



IpMorph : “How to defeat common OSFP tools”

Nmap : Spoofing tests SP / GCD / ISR

RD: TCP RST data checksum

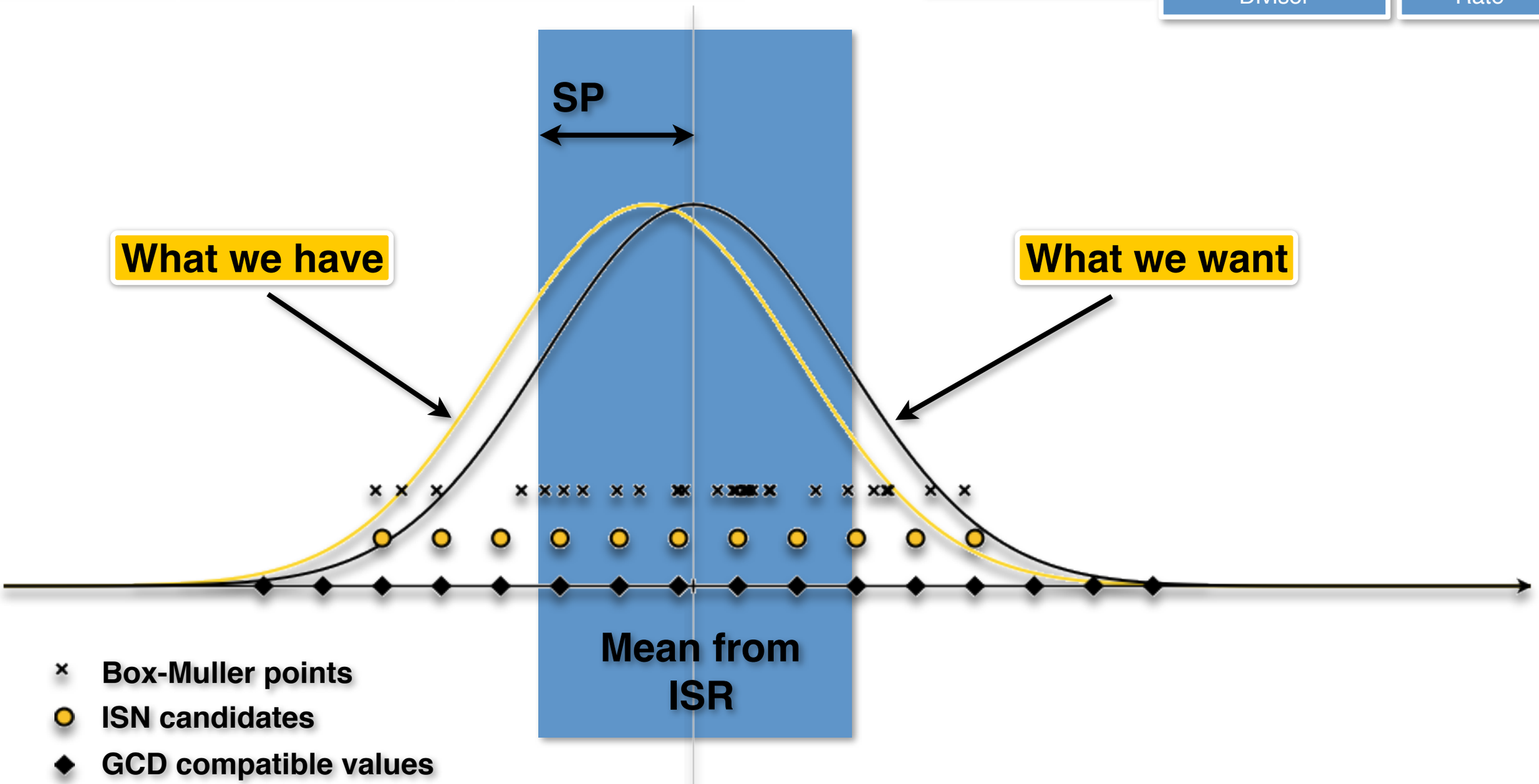
TI : TCP IP ID sequence generation algorithm

II : ICMP IP ID sequence generation algorithm

SP : TCP ISN Predictability

GCD : TCP ISN Greatest Common Divisor

ISR : TCP ISN counter Rate



IpMorph : “How to defeat common OSFP tools”

Nmap : Spoofing tests SP / GCD / ISR

RD: TCP RST data checksum

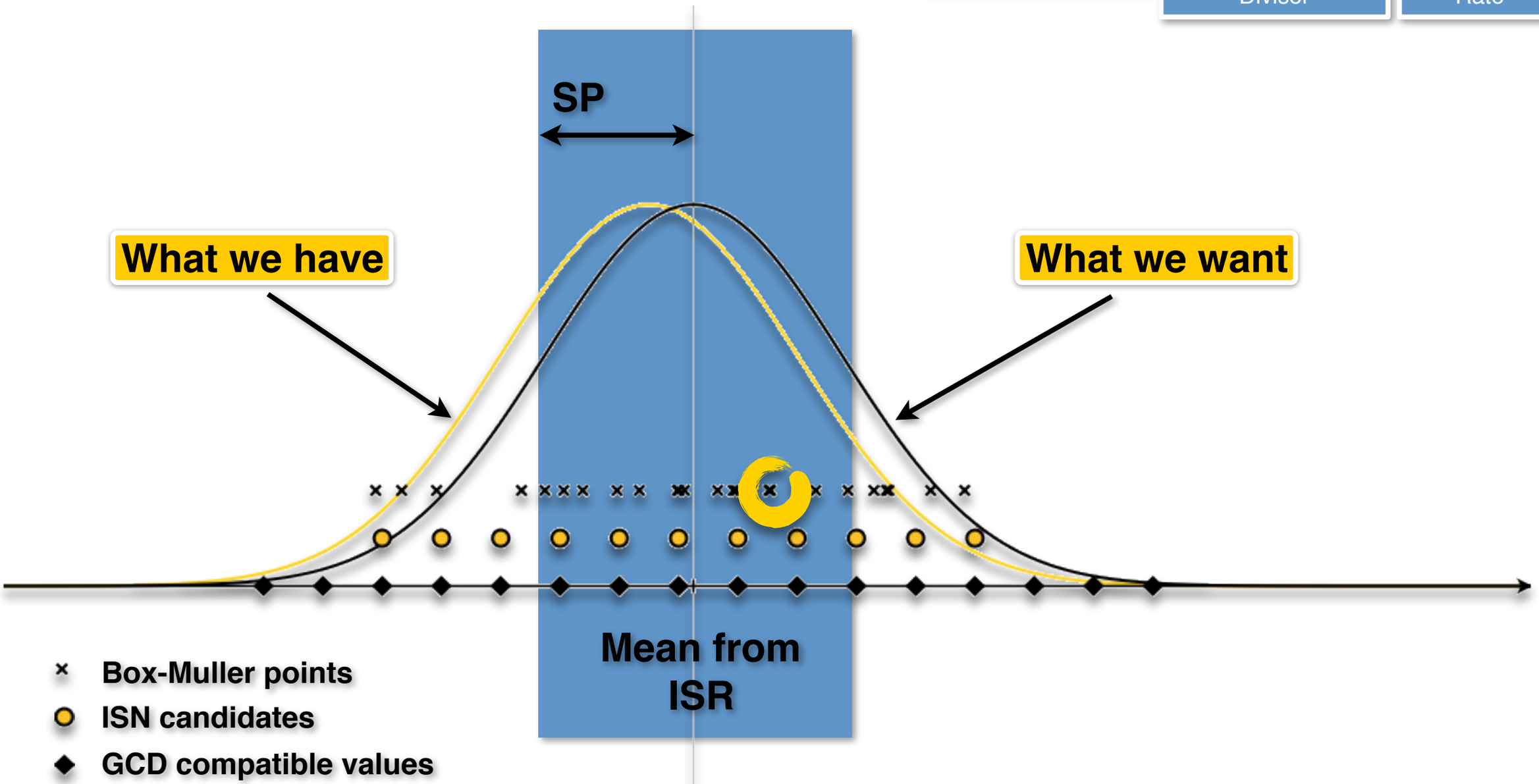
TI : TCP IP ID sequence generation algorithm

II : ICMP IP ID sequence generation algorithm

SP : TCP ISN Predictability

GCD : TCP ISN Greatest Common Divisor

ISR : TCP ISN counter Rate



IpMorph : "How to defeat common OSFP tools"

SinFP : How is the spoofing done ?

Binary : heuristic0, heuristic1, heuristic2

Constants:
- **TTL, ID, DF**
- **seq and ack**

TcpFlags : heuristic0, heuristic1, heuristic2

Offset	0-3	4-7	8-11	12-15	16-19	20-23	24-27	28-31
0	Source port				Destination Port			
32	Sequence number							
64	Acknowledgment number							
96	Data Offset	*Reserved*	Flags		Window Size			
128	Checksum				Urgent Pointer			
160 ...	Options ...							

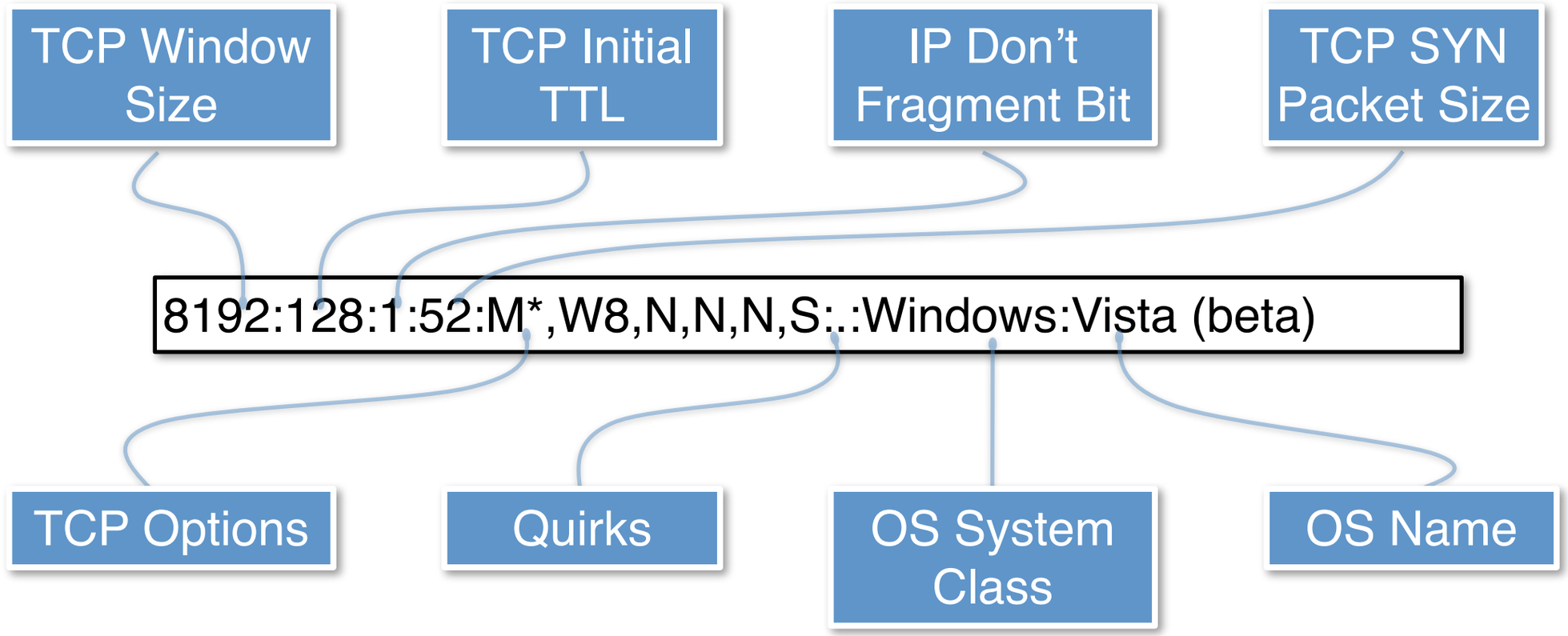
TcpWindow : heuristic0, heuristic1, heuristic2

TcpMss : heuristic0, heuristic1, heuristic2

TcpOptions : heuristic0, heuristic1, heuristic2

IpMorph : “How to defeat common OSFP tools”

p0f : Signature format

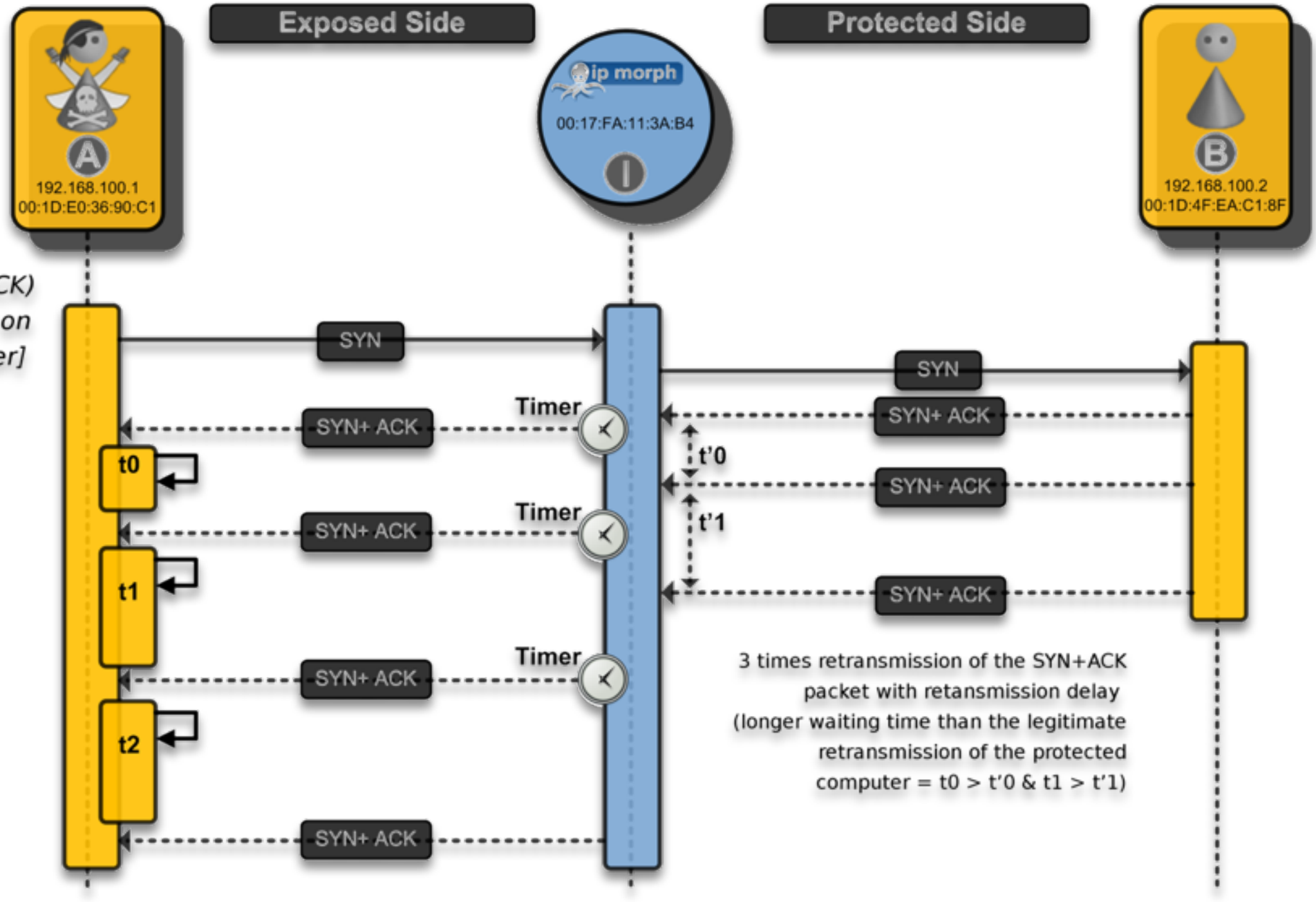


- Version 2.0.8 (2006)
- 6 parameters analyzed
- Only on SYN packets (default DB = p0f.fp)
- Actually can analyze other type of packets, but very few signatures available (experimental)

IpMorph : “How to defeat common OSFP tools”

Ring2 - congestion timeouts spoofing

[Connection (SYN+ACK) congestion spoofing on behalf of the attacker]



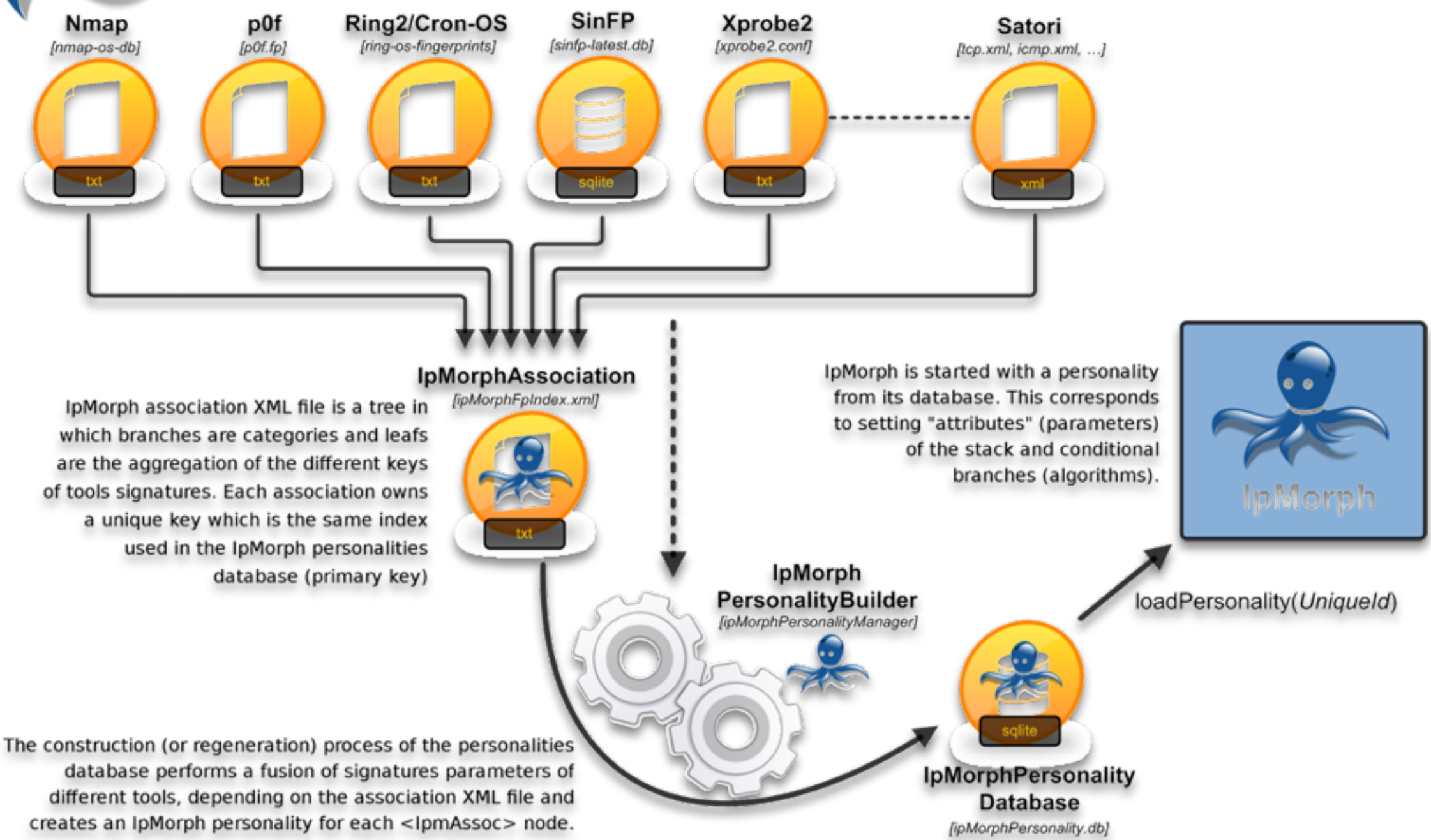
IpMorph : “How to defeat common OSFP tools”

Future

- **June 2009 – SSTIC 2009**
  - First demo
  - « *Beta release* » 0.1 (available on website)
- **End 2009 – Beginning 2010**
  - Refactoring (leaner, faster, easier to use)
  - PersonalityManager, various tools...
  - Version 0.2 will be available on website too
  - Documentation...
  - Possibly, integrate application-level scrubbers (DNS, SMB, DHCP, …) ?

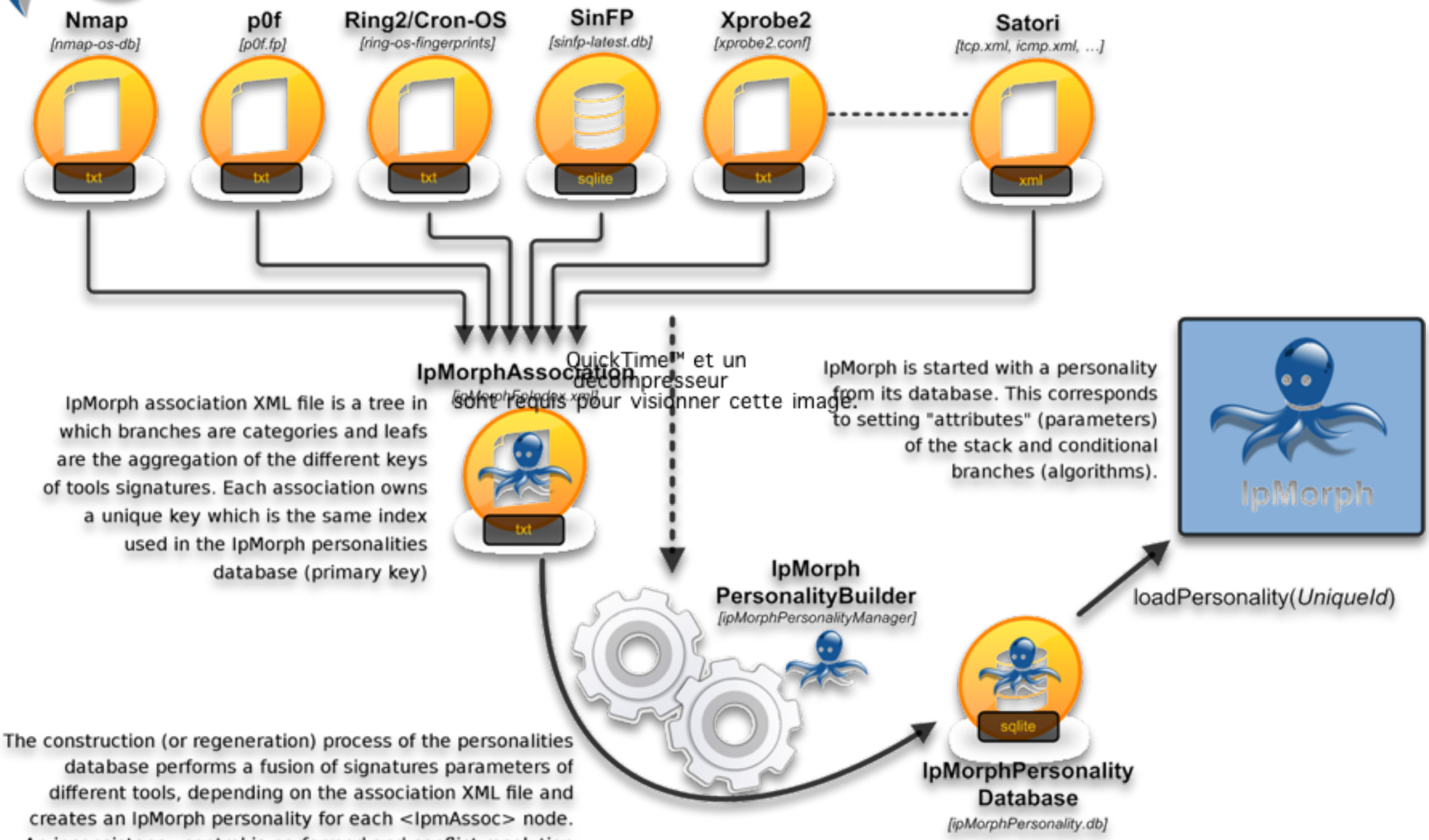
IpMorph : “How to defeat common OSFP tools”

Personality Manager



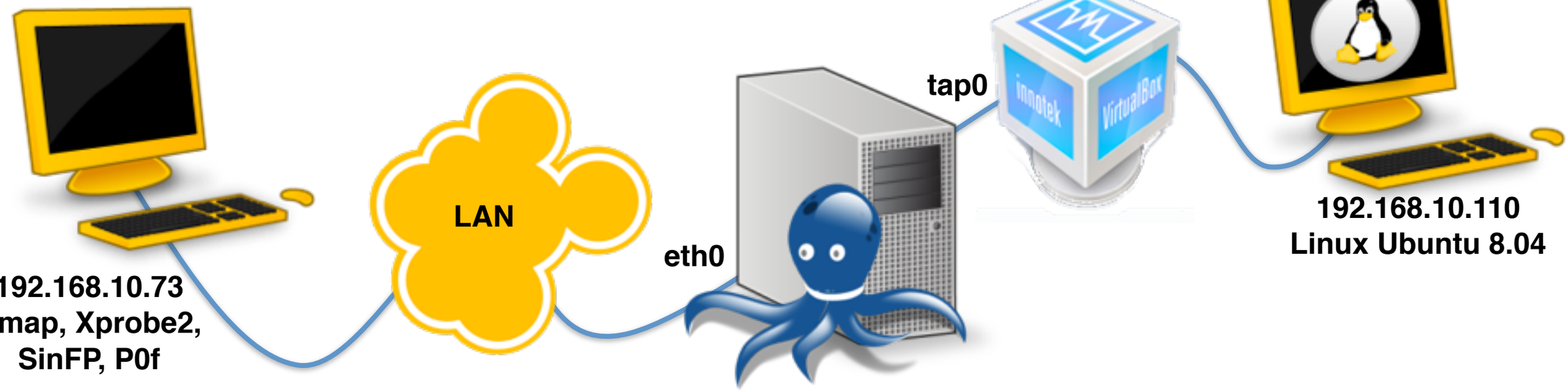
IpMorph : “How to defeat common OSFP tools”

Personality Manager



IpMorph : “How to defeat common OSFP tools”

Demo



Demo’s scenario

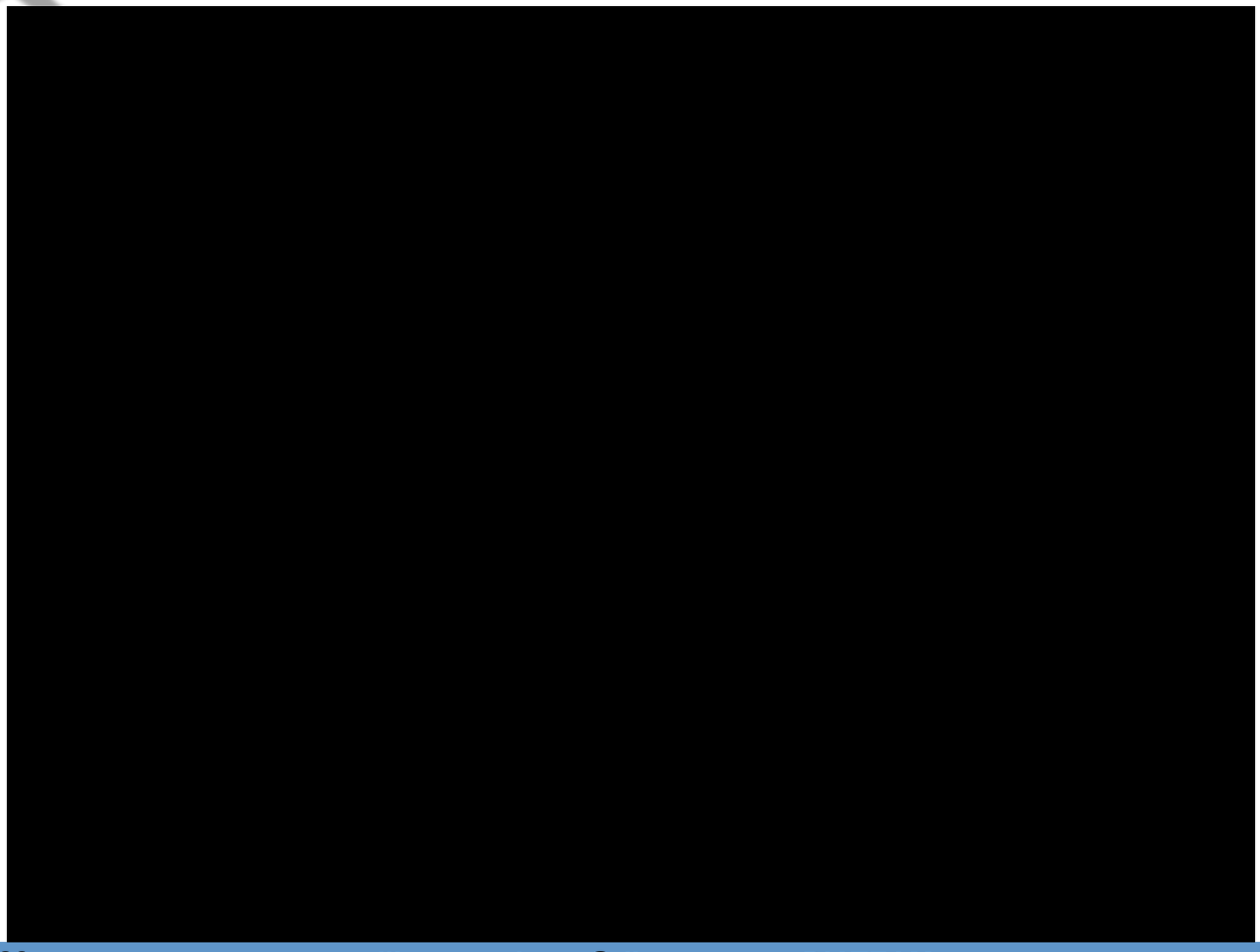
Configuration	Active Fingerprinting	Passive Fingerprinting
1 - Interface tap0	4 - Xprobe2	7 - SinFp as passive
2 - VirtualBox	5 - Nmap	8 - p0f
3- IpMorph	6 - SinFp as active	

IpMorph : “How to defeat common OSFP tools”

Demo XProbe2

IpMorph : “How to defeat common OSFP tools”

Demo XProbe2

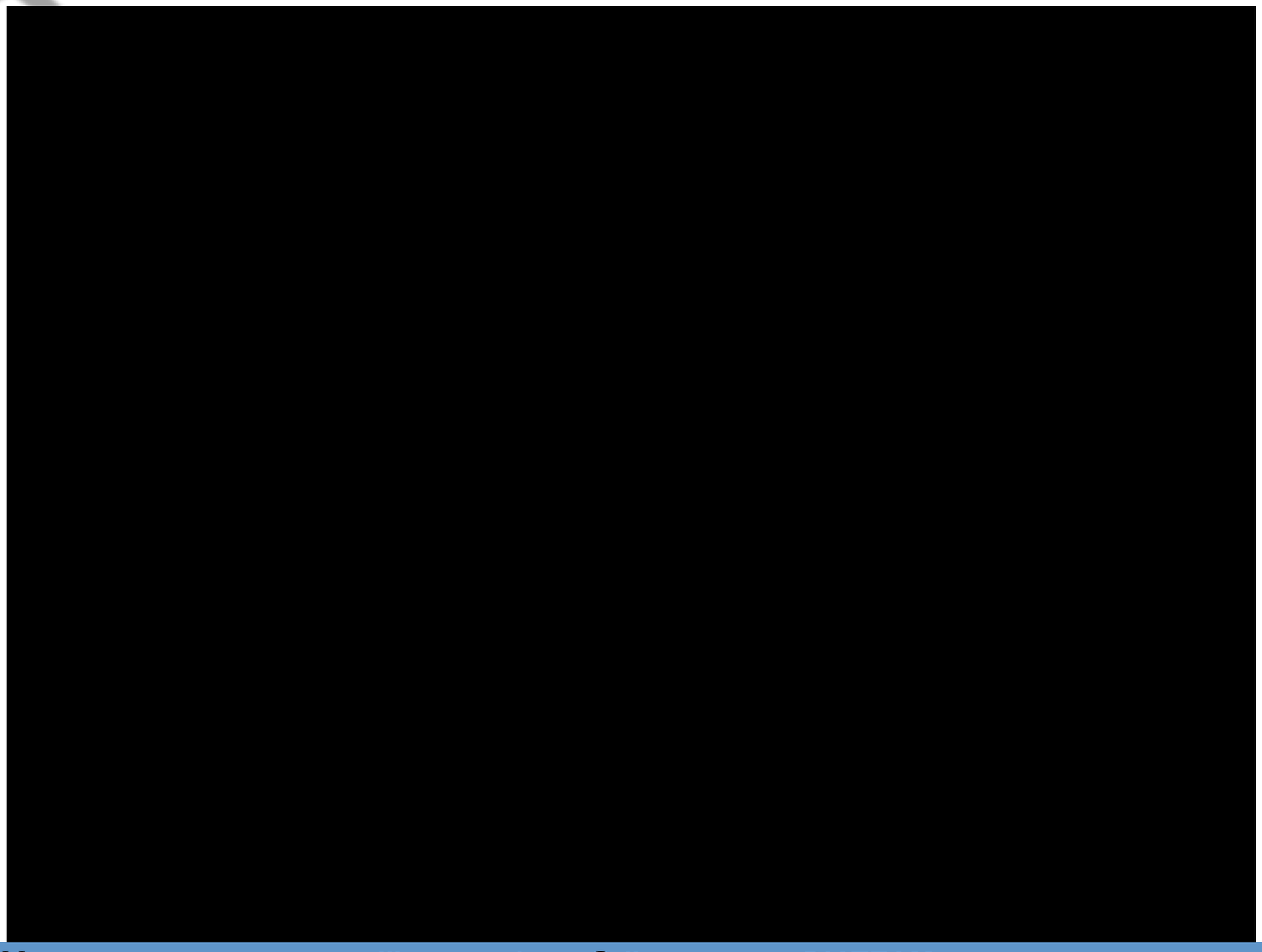


IpMorph : “How to defeat common OSFP tools”

Demo Nmap

IpMorph : “How to defeat common OSFP tools”

Demo Nmap

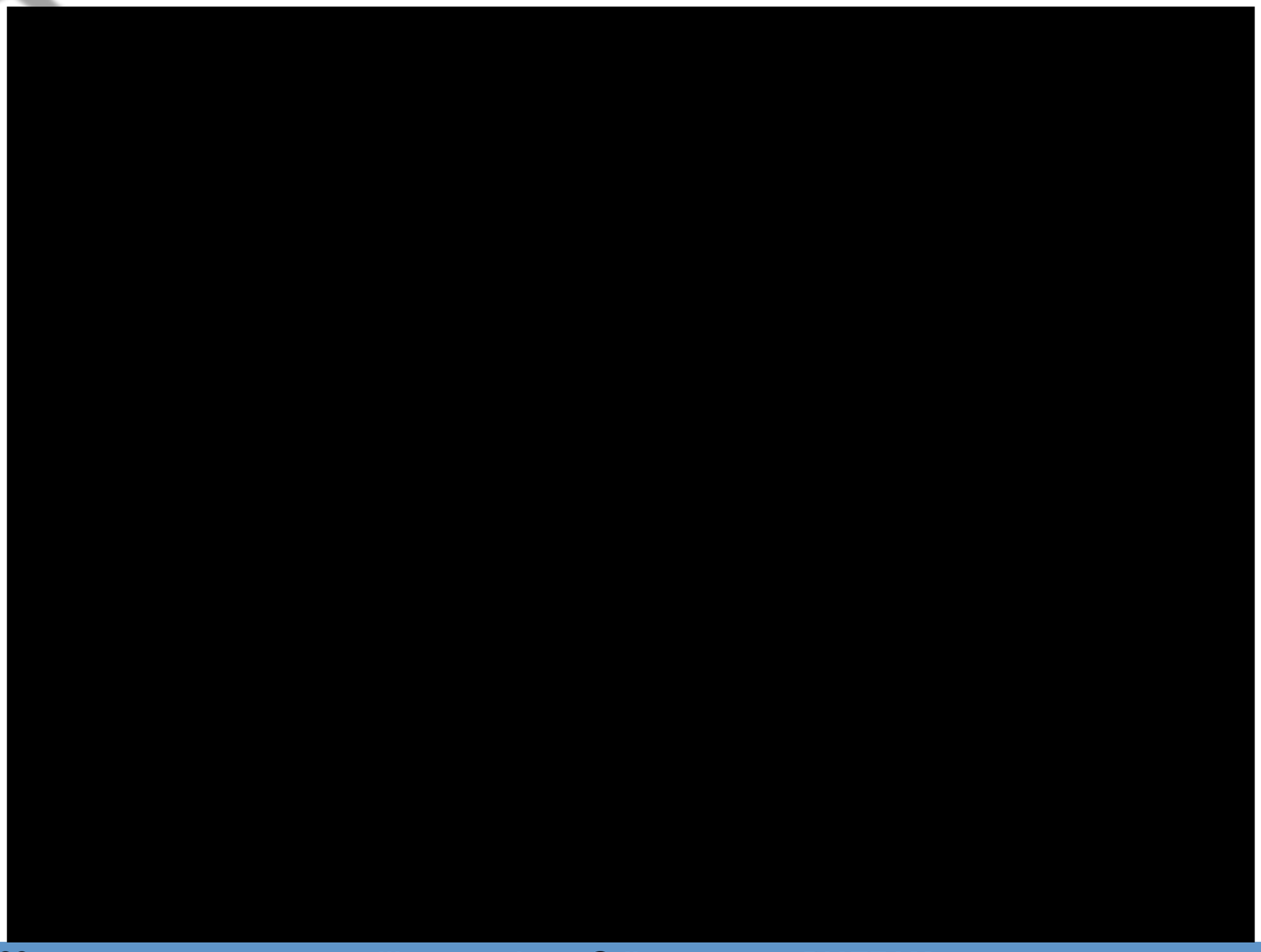


IpMorph : “How to defeat common OSFP tools”

Demo SinFP active

IpMorph : “How to defeat common OSFP tools”

Demo SinFP active

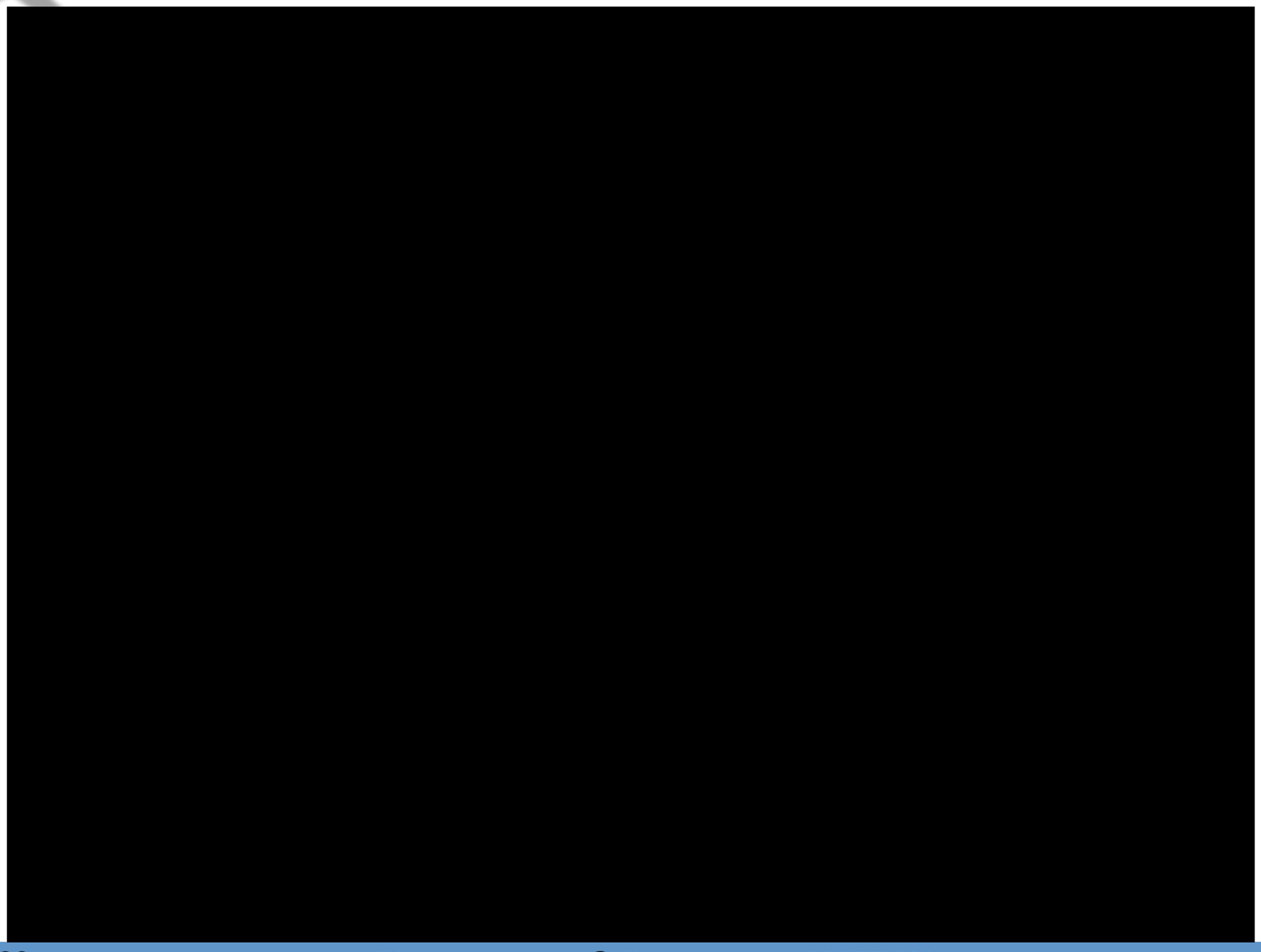


IpMorph : “How to defeat common OSFP tools”

Demo SinFP passive

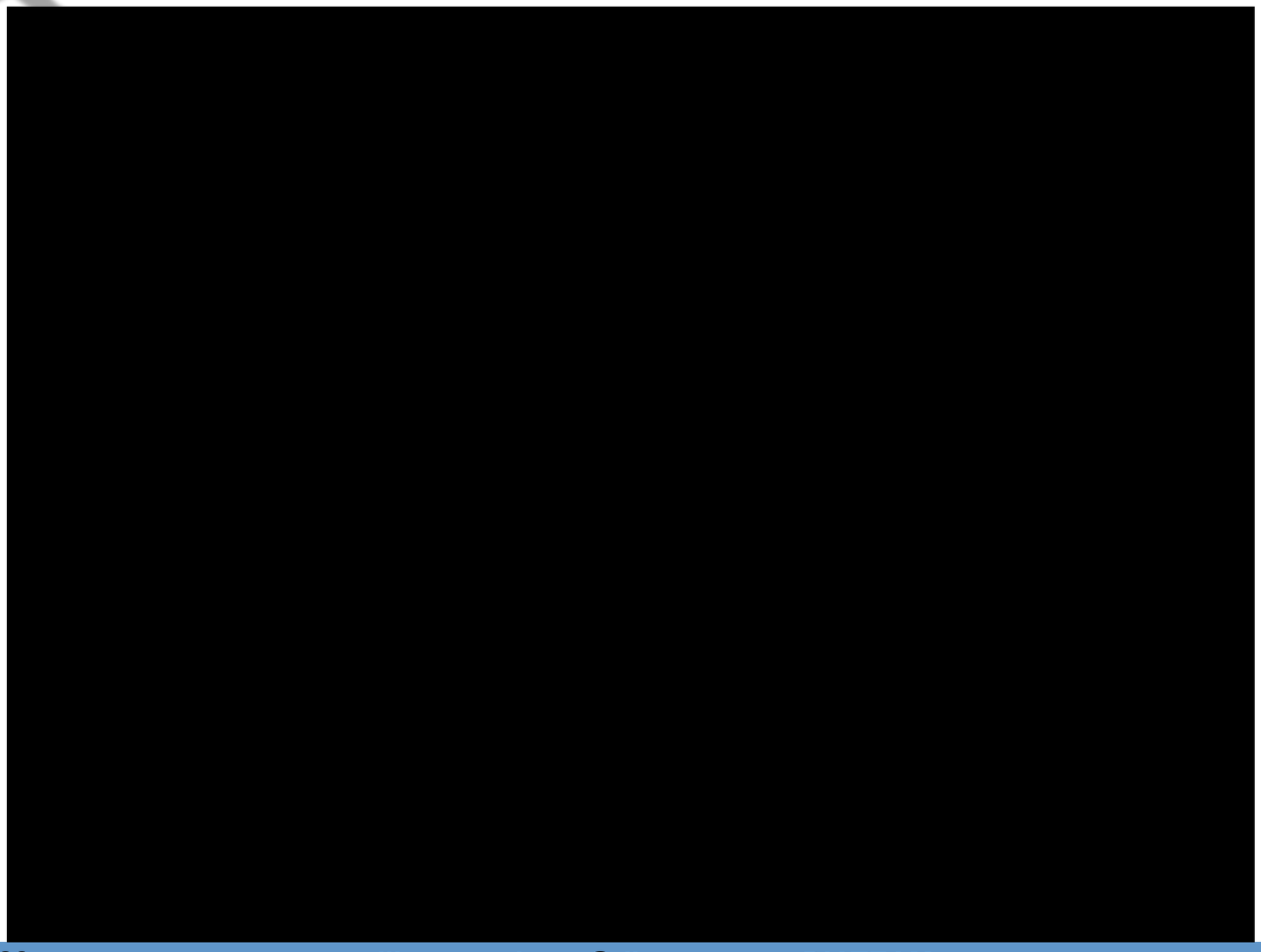
IpMorph : “How to defeat common OSFP tools”

Demo SinFP passive



IpMorph : “How to defeat common OSFP tools”

Demo p0f



Thank you for listening! Questions ?