#sf21veu

Scapy Turned 18

Boy They Grow Up Fast, Don’t They!

Guillaume Valadon
Quarkslab

#sf21veu

Hello!

I am Guillaume Valadon

a network and security enthusiast.

I am here to share part of Scapy history. So far,
it is only known by its maintainers.

You can find me at @guedou

#sf21veu

Get the slides at https://tknk.io/F5LN

#sf21veu

What is Scapy?

In a Nutshell

simplified network interactions in Python

- interactive shell & Python module

define a packet and sent it with a single line

```
srp1(Ether() / IP(dst="sharkfesteurope.wireshark.org") / ICMP())
```

easily add a new protocol

- define a list of fields to parse and construct a header

Batteries Included

PCAP manipulations

rdpcap() & wrpcap()

many supported protocols

802.11, IPv6, DNS, TLS, BLE, ZigBee, HTTP/2...

multi-platform

Linux, macOS, *BSD, Windows

#sf21veu

```
$ git clone https://github.com/secdev/scapy
$ cd scapy/
$ sudo ./run_scapy

                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.4.5
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | We are in France, we say Skappee.
       scccccp///pSP///p          p//Y   | OK? Merci.
      sY/////////y  caa           S//P   |             -- Sebastien Chabal
       cayCyayP//Ya              pY/Ya   |
        sY/PsY////YCc           aC//Yp
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps

>>>
```

#sf21veu

```
>>> r, u = sr(IP(dst="8.8.8.0/24", ttl=3) / TCP(flags="R"))
>>> [p for p in r if not ICMP in p.answer]
```

Some Numbers

#sf21veu

18 years old

developed by Philippe Biondi, since 2003
maintained by Gabriel, Guillaume & Pierre, since 2012

300 contributors

5 regular ones

50k PyPi installation per day

still a lot of Python2

Project Management

best effort

volunteer work only

our employers are really supportive

annual release

using master is recommended

#sf21veu

Before Scapy

Forging Packets in C

complete control on packet manipulations

- set any value to any field

hundred of lines of code

- read / parse & forge / send
- routing, source address selection, checksums computation...

dnet & pcap libraries

- simplified common tasks
- code portability

hping

command-line based network interactions

```
hping3 --icmp sharkfesteurope.wireshark.org
```

Tcl scripting

```
hping3> hping send "ip(daddr=sharkfesteurope.wireshark.org)+icmp()"
```

pyrat.py

Scapy ancestor developed in January 2003

- https://github.com/secdev/pyrat

validated ideas

- protocol stacking
- default values
- simple packet injection

#sf21veu

```
$ sudo python2.7 pyrat.py
Welcome to PyRat
>>> send(Ether() + ARP() + "pyrat was here!")
```

#sf21veu

Scapy Concepts

Default Values

default packet always work

```
IP(dst="sharkfesteurope.wireshark.org") / TCP()
```

smart fields values

80 for TCP.dport, 64 for IP.ttl ...

compute values automatically

IP & TCP checksums, source address selection, ...

Protocols Layers

stack layers with the / operator

```
DNS() / IPv6() / ARP()
```

each layer is a Packet object

list of fields and Python methods

#sf21veu

```
class UDP(Packet):
    name = "UDP"
    fields_desc = [ShortEnumField("sport", 53, UDP_SERVICES),
                   ShortEnumField("dport", 53, UDP_SERVICES),
                   ShortField("len", None),
                   XShortField("chksum", None), ]
```

Build & Parse

raw() builds a Packet & converts it to bytes

```
data = raw(IP(ttl=42) / UDP())
```

each layer can parse itself

```
p = IP(data)
p.ttl == 42
```

Ease of Use

no external dependency

clone the repository and you're good to go

plenty of useful functions

```
sniff()      - sniff packets
wireshark()  - view packets in Wireshark
sr()         - send & receive packets
hexdump()    - hexadecimal view
```

#sf21veu

Scapy Take-off

AnsweringMachine

opposite of sr()

- wait for a packet and send an answer

simplify server / client interactions

- simple DNS & DHCP daemons
- ARP & NDP spoofing

#sf21veu

```
class ProbeRequest_am(AnsweringMachine):
  function_name = "pram"

  mac = "00:11:22:33:44:55"

  def is_request(self, pkt):
    return Dot11ProbeReq in pkt

  def make_reply(self, req):

    rep = RadioTap()
    rep /= Dot11(addr1=req.addr2, addr2=self.mac, addr3=self.mac,
                 ID=RandShort(), SC=RandShort())
    rep /= Dot11ProbeResp(cap="ESS", timestamp=int(time.time())
    rep /= Dot11Elt(ID="SSID", info="Scapy !")
    rep /= Dot11Elt(ID="Rates", info='\x82\x84\x0b\x16\x96')
    rep /= Dot11Elt(ID="DSset", info=orb(10))

    return rep
```


●●●○○ Orange F
23:01
34 %
Settings
Wi-Fi
Livebox-c9a0
mouchky
NEUF_999C
orange
Scapy !
SFR WiFi FON
SFR WiFi Mobile
Other…
Ask to Join Networks
Known networks will be joined automatically. If no known networks are available, you will have to manually select a network.

#sf21veu

IPv6

initial support in 2005

- merged in 2008 in Scapy 2.0

many protocols implemented

- NDP, DHCPv6, MIPv6, NIQ...

playground to learn & experiment

- CVE-2007-4285 - Cisco IOS & XR crash with a Routing Header
- RFC5095 - Deprecation of Type 0 Routing Headers in IPv6

UTScapy

regression tests needed

- IPv6 broken several times during its development
- modifying Scapy core impacts protocols

dedicated tool for Scapy campaigns

- written in 2005 when pytest was released
- Python code between markups

#sf21veu

UTScapy tests

Shrink All | Expand All | Expand Passed | Expand Failed

000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 054 055 056 057 058 059 060 061
062 063 064 065 066 067 068 069 070 071 072 073 074 075 076 077 078 079 080 081 082 083 084 085 086 087 088 089 090 091 092 093 094 095 096 097 098 099 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123
124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185
186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239

Regression tests for Scapy

CRC=6343C3EA SHA=6E86F6642B0302B9E619059873692618FE221396
Run Tue Jun 15 11:16:38 2021 from [test/regression.uts] by UTscapy

PASSED=237 FAILED=3

B6BC775E Information on Scapy

- +000+ DA599690 Setup
- +001+ D18177CF Get conf
- +002+ C7B4EFF2 Test module version detection
- +003+ 19EC7768 List layers
- +004+ FE0F029B List layers - advanced
- +005+ CA149C8C List packet fields - ls
- +006+ B614219C List commands
- +007+ AD124E63 List contribs
- +008+ B7884B09 Test automatic doc generation
- +009+ DE8CCBA1 Check that all contrib modules are well-configured
- -011- 3F6538B8 Configuration conf.use_* LINUX

```
>>> try:
...     conf.use_bpf = True
...     assert False
... except:
...     True
...
True
>>> assert not conf.use_bpf
Traceback (most recent call last):
  File "<input>", line 2, in <module>
AssertionError
```

- +012+ 3F6538B8 Configuration conf.use_* WINDOWS
- +013+ 37C97003 Configuration conf.use_pcap
- +014+ A123BCF3 Test layer filtering
- +015+ 970EB61E UTscapy route check

B87462A3 Scapy functions tests

- +016+ FFEF292B Interface related functions

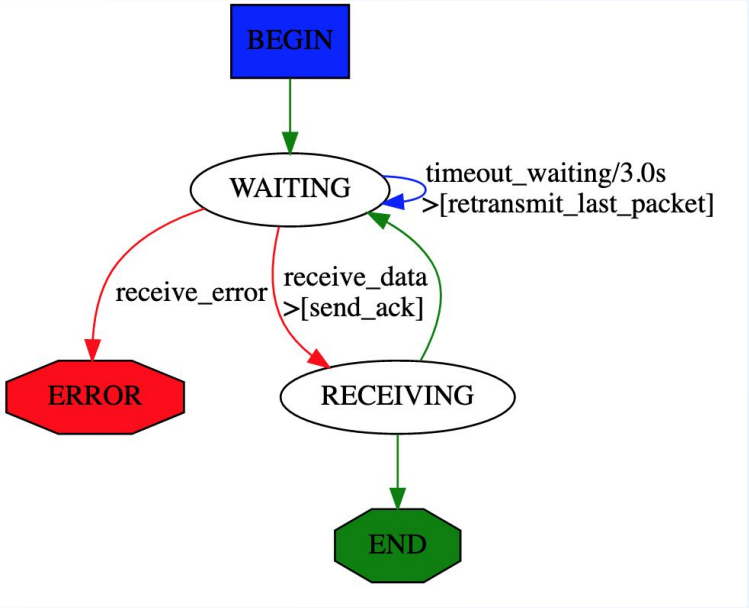
Automaton

define finite-state machines

states, conditions & actions

extend the question/answer model

examples: TCP, TFTP



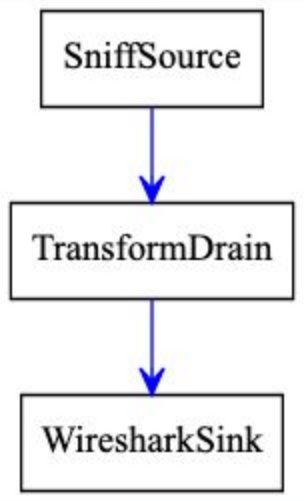
Pipes

complex data management

- sequence of inputs and outputs

many building blocks

- sniff packets
- transform packets
- TCP listen & connect



#sf21veu

```
from scapy.all import *

source = SniffSource(iface=conf.iface)
wire = WiresharkSink()

def transf(pkt):
    if not pkt or IP not in pkt:
        return pkt
    pkt[IP].src = "1.1.1.1"
    pkt[IP].dst = "2.2.2.2"
    return pkt

source > TransformDrain(transf) > wire

p = PipeEngine(source)
p.start()
p.wait_and_stop()
```

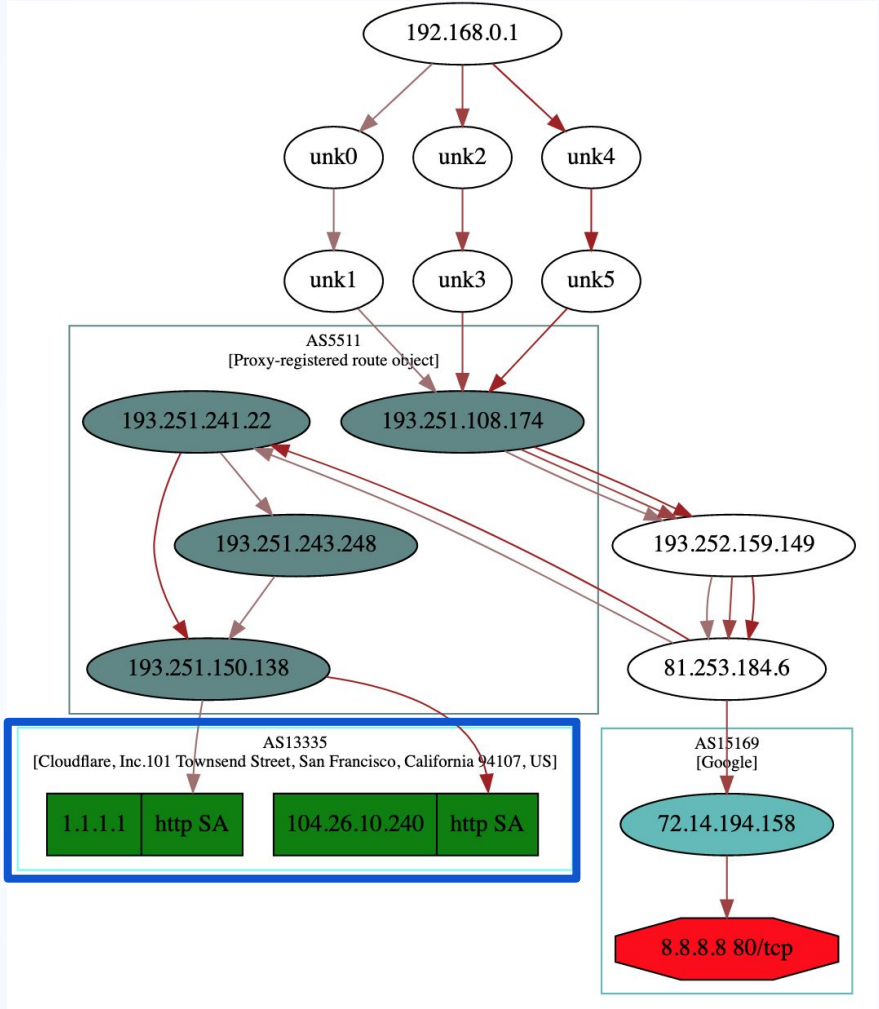
traceroute()

typical TTL-based measurement with a twist

- specify the IP payload
- several target at once

many visualizations

- world map
- 3D representation


192.168.0.1
unk0
unk2
unk4
unk1
unk3
unk5
AS5511
[Proxy-registered route object]
193.251.241.22
193.251.108.174
193.251.243.248
193.252.159.149
193.251.150.138
81.253.184.6
AS13335
[Cloudflare, Inc.101 Townsend Street, San Francisco, California 94107, US]
1.1.1.1
http SA
104.26.10.240
http SA
AS15169
[Google]
72.14.194.158
8.8.8.8 80/tcp

#sf21veu

Scapy 2.0 - May 2008

split the 14000 lines file

- allow specific imports
- simpler modifications & merges

directories hierarchy

- arch - platform related code
- layers - protocols on a 'typical' LAN
- contrib - exotic protocols

```
$ tree -L 1 -d scapy/
scapy/
├── arch
├── asn1
├── contrib
├── crypto
├── layers
├── modules
└── tools

7 directories
```

Loss of Speed

less commits after 2010

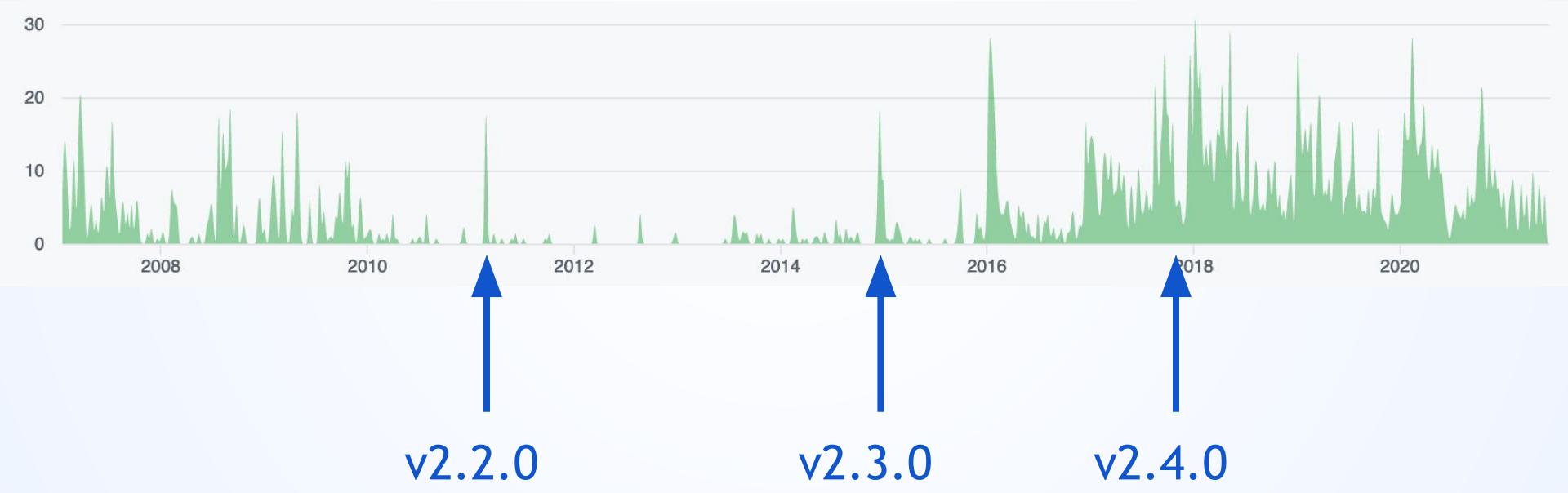
- Philippe was the only developer
- contributions were difficult
- no release during 3 years

self-hosting limits

- switch to Bitbucket in 2013
- easier contributions meant more contributors
- significant maintenance effort

Commits History

#sf21veu



#sf21veu

Rebirth

github

#sf21veu

move from Bitbucket to github in January 2016

- and from Mercurial to git

many benefits

- better project visibility
- more contributors
- github ecosystem: Travis, AppVeyor, codecov, gitter...

Continuous Integration

first improvement after github migration

- Linux, macOS & Windows with Travis and AppVeyor

run UTScapy unit tests automatically

- catch bugs across Python versions & platforms
- identify regressions

Python3

several constraints

- Scapy 3.0 PoC, a Python3 rewrite
- keep Python2 compatibility
- tests only cover 50% of the code

divide & conquer

1. coverage - enhance code coverage
2. convergence - small Python3 related changes

PEP08

no coding convention was enforced

- confusing for new contributors
- difficult to introduce new clean code

constraints

- credit original authors
- preserve git history
- simplify reviews & avoid conflicts

TLS & X.509

up to TLS v1.3

- sniff
- encrypt / decrypt TLS messages
- extract certificates

certificates manipulation

- parse & display content
- verify signatures
- change values & resign

#sf21veu

```
cert_sharkfesteu = Cert(pem2der(open("sharkfesteu.pem", "rb").read()))

cert_cloudflare = Cert(pem2der(open("cloudflare.pem", "rb").read()))

cert_sharkfesteu.isIssuerCert(cert_cloudflare)
```

Automotive

use Scapy for automotive pentest

- biggest contrib to date

swiss-army knife

- from data-link to application layers: CAN, ISO-TP, OBD…
- forge, sniff, MiTM...

Marketing

reaching out

- logo by @BenRenaut
- tutorials during conferences
- gitter chat

documentations pyramid

- concise README
- IPython notebooks
- scapy.readthedocs.io

Some Use Cases

Exploits

EXTRABACON

- part of the NSA Vault7 leak
- SNMP RCE on Cisco ASA

IPv6

- CVE-2021-24086 & CVE-2019-5597 - fragment header
- CVE-2020-25577 & CVE-2020-16898 - router advertisement

Recent Wireless Vulnerabilities

802.11

- KrackAttacks
- FragAttacks

BLE

- SweynTooth
- BLURtooth

Unit Tests

OS networking stacks

- Linux
- OpenBSD
- FreeBSD
- RIOT-OS

eBPF ecosystem

- Facebook Katran
- xpress-dns

#sf21veu

Looking Ahead

#sf21veu

improvements

- Python type annotations
- eBPF-based per process sniffing
- question the Python2 support

experiments

- Packet JIT
- lazy parsing
- Rust core

#sf21veu

Questions?

Issues?

Pull Requests?