

Fingerprint all the things with SCANNERL

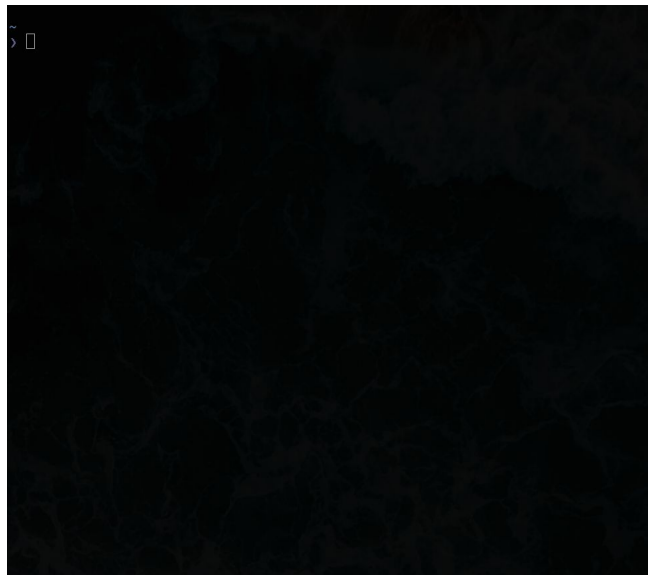
Adrien Giner
Kudelski Security
BlackAlps November 2017



Who am I

- Security Researcher in the R&D team at Kudelski Security
- Focus on network-related security research
- Threat intel port scanning and fingerprinting
- Blog sometimes, code a lot
- CLI and open-source software

This is a terminal



Ever wanted to ...

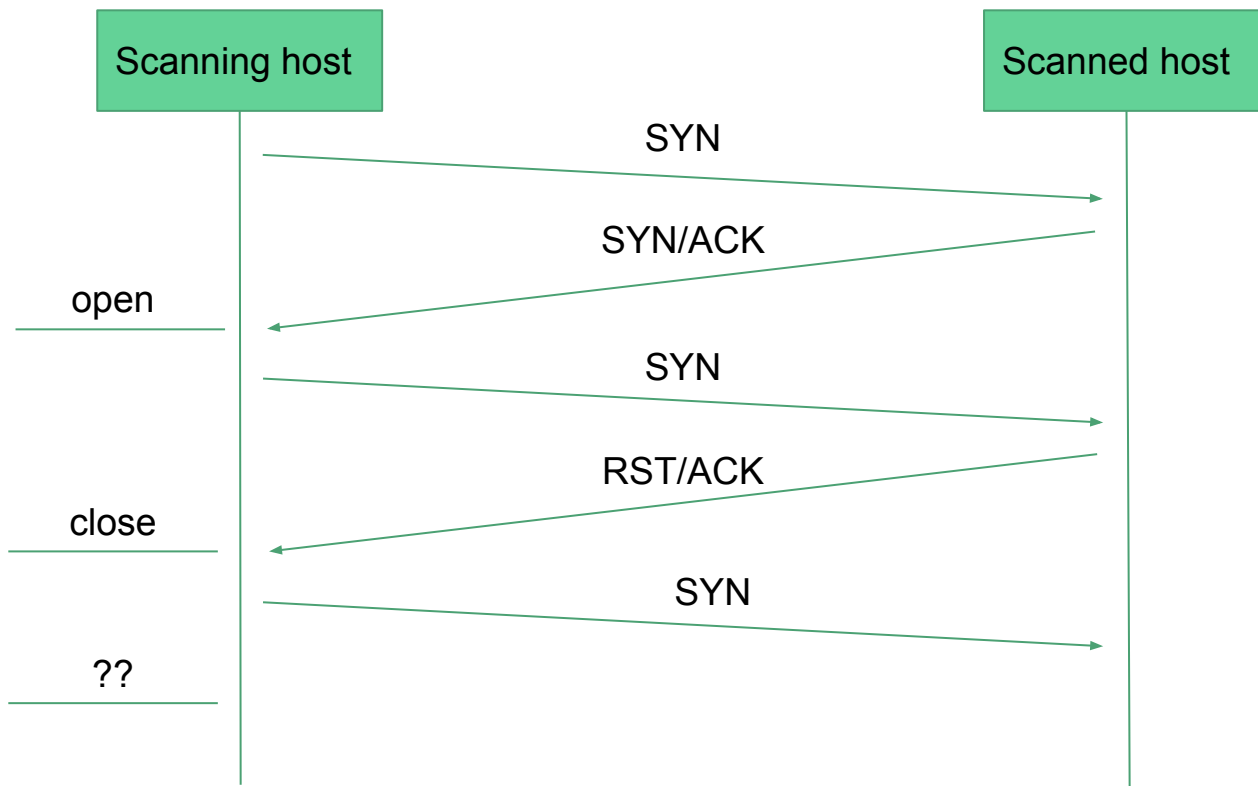
- Know the versions of all DNS servers on IPv4
- Match a specific pattern on all pages of the top 1M websites
- Retrieve Certificates from a list of websites
- Identify SSH versions deployed on your perimeter
- ...

Agenda

- Service fingerprinting challenges
- Existing tools and their limitations
- Scaling vertically doesn't work
- Scaling horizontally is better
- Scannerl
- Summary

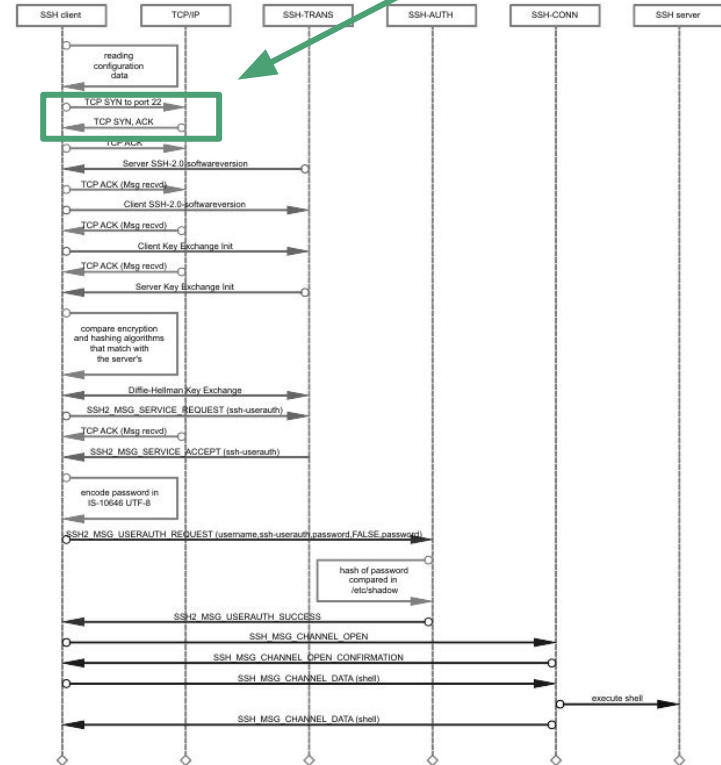
Service Fingerprinting

Port scanning recap (TCP)



Service fingerprinting goes beyond

- Targets the application layer
- Involves multiple exchanges
- Depends on the information needed
- Different behaviors for all protocols



Service fingerprinting challenges

- More complex than port scanning
- Understand the protocol
- Longer process than port scanning
- Much larger data set
- Potentially unexpected data

Now consider running it on millions of targets

- Long (very long) process
- Network impacts
- Handle the dataset and its storage
- Be fast and efficient



Existing tools and their limitations

Solutions to fingerprint services - scripting

Script it

- Scapy
- Hping
- Netcat
- ...



a lot of power but tedious and very specific

\$ # the hardcore way

\$ **export** host="127.0.0.1"; **export** port="80"

\$ **exec** 3<>/dev/tcp/\${host}/\${port}

\$ **echo** -e "GET / HTTP/1.1\r\nhost: \${host}\r\n\r\n" >
&3

\$ **cat** <&3

Solutions to fingerprint services - tools

Use **existing tools**

- Nmap/NSE scripting engine
- Zgrab (from zmap team)
- ...



flexible and complete but do they scale ? How fast are they ?

How slow is it

- Alexa top 1M
- 24 CPUs / 190G RAM
- 1G pipe
- Tweaked for scanning

Zgrab: 48 minutes

Nmap: 8% done after 14h ...
cancelled



Scaling vertically does not work

Existing tools do not scale well

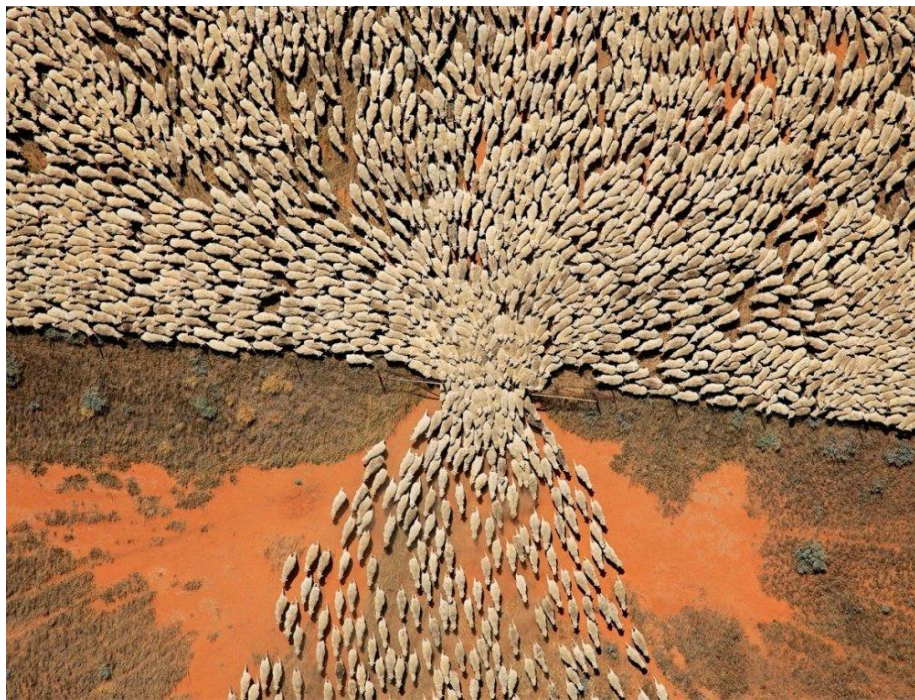
Only option is to scale vertically

- Add more resources (CPU / RAM / IOPS)
- Add a better network card
- Get a bigger pipe
- Tweaking (network stack, ...)

At some point you will hit a limit

Also scaling vertically does not solve these issues

- Limited by low bandwidth routing devices
- Blacklisting
- Single point of failure (blocked, crash, ...)
- Expensive



Scaling horizontally is better

What about scaling horizontally?

Get your own army of VMs and distribute your tasks

- Use the cloud
- throw-away and cheap VMs
- Virtually not limited to the number of nodes
- Affordable



Horizontal scaling advantages for fingerprinting

- **Stealthier**

more source IPs, less likely to be blacklisted

- **Fairness**

Reduce impact on routing devices by spreading the load

- **Resiliency**

if it crashes/gets blocked, only part of it is *lost*

- **Speed**

More minions = faster

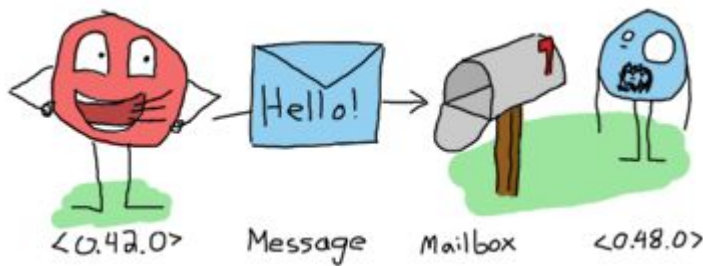
Distributing an application brings its own issues

- Synchronization is a nightmare
- More management work (nodes)
- Communication is a pain
- Task distribution
- Failures handling
- Results aggregation



Erlang/OTP to the rescue

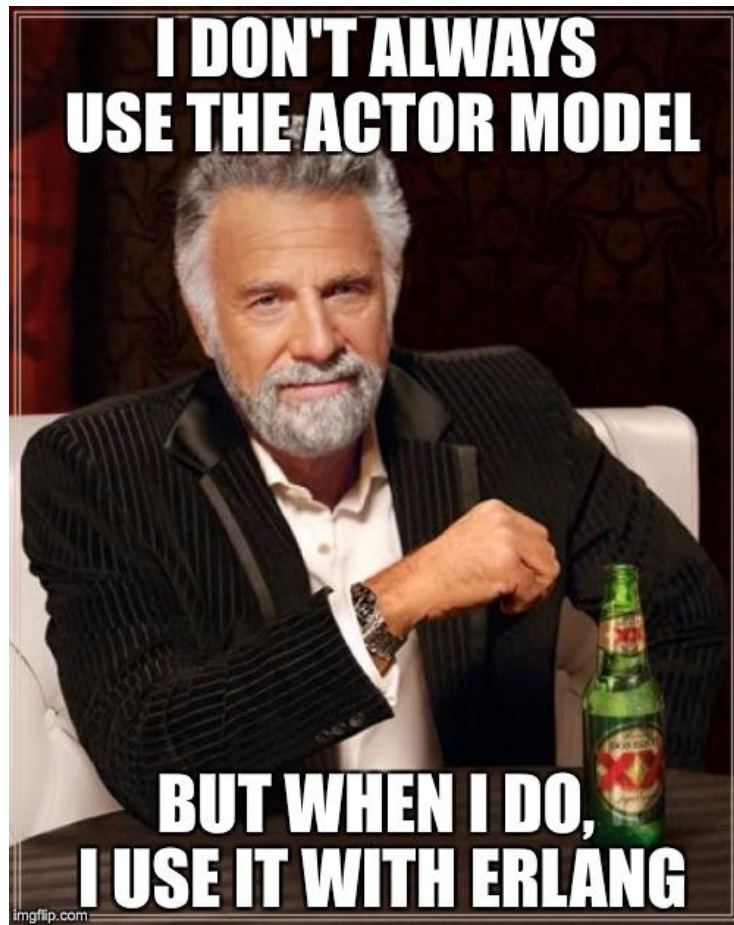
- Concurrency available out of the box
- Distributed Applications
- Message passing for the communication
- *Actor model*
- Resiliency
- *let it crash philosophy*
- Very small *process* size



source: <http://learnyoussomeerlang.com/>

The actor model FTW

- Each object is an actor in a (erlang) process
- Isolated actor receive messages and act on it
- Communication through message passing
- Sequentially process async tasks



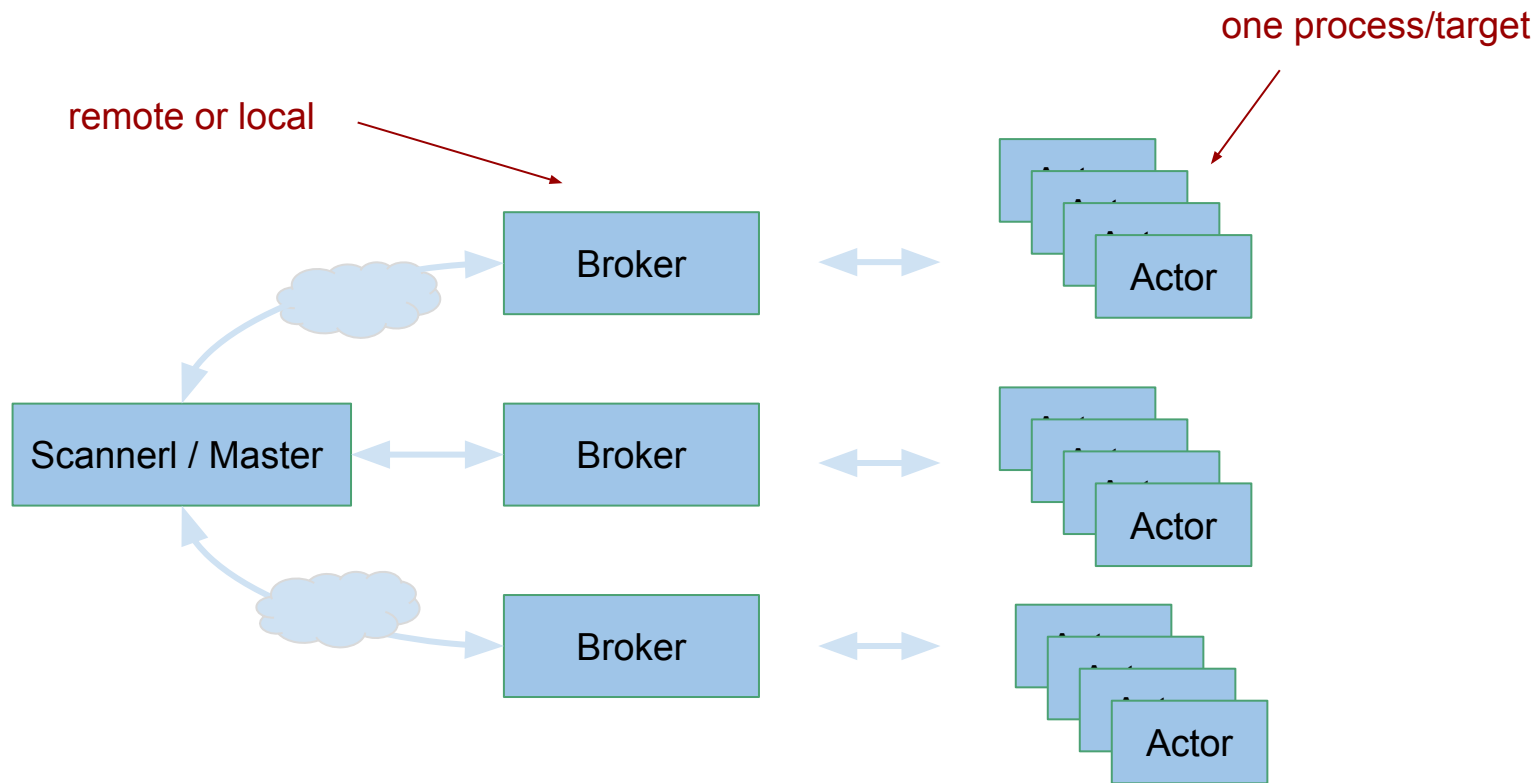
Let's consider each target as an actor

- Spin a new (erlang) process per target
- Each actor executes the requested task (connect, send, receive, ...)
- Once done, return results and die



Scannerl

Scannerl - the distributed fingerprinting tool



Allows for *infinite* scaling

- Decide the number of broker (VM)
- Decide the number of actor (process) per broker

Me being optimistic



Add modularity in the mix for more power

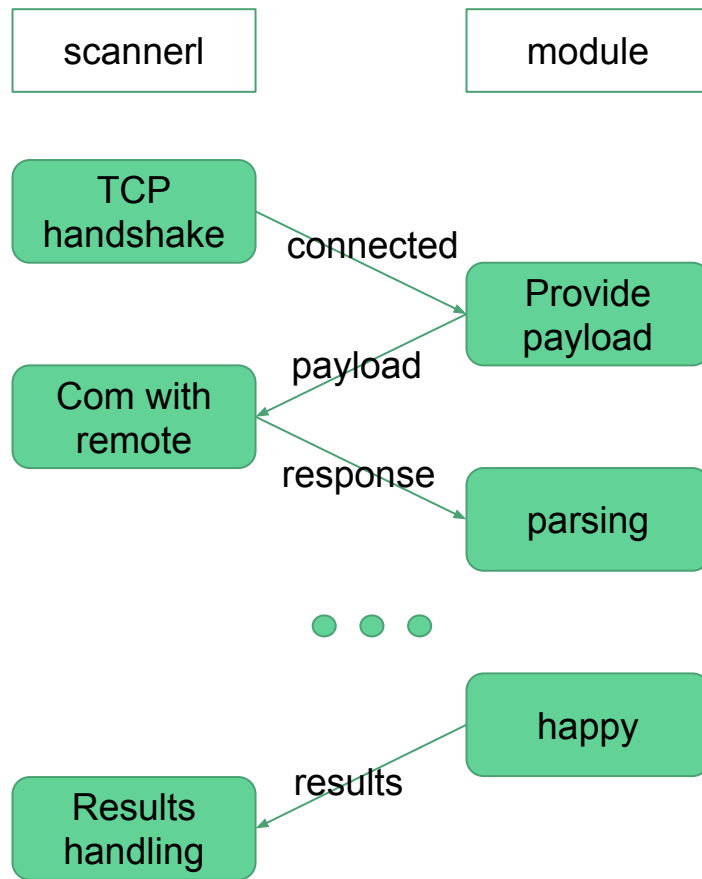
Core provides the basic blocks (parsing, setup remote nodes, distributing tasks, ...)

Fingerprint modules - this is the actual communication with the remote service (the actor part)

Output modules - this is the result aggregation / filtering / outputting


Fingerprinting module API

- Different types: TCP, UDP and SSL
- *Callback* function called at each stage
- Scannerl handles the transport layer (abstraction)
- Multiple exchanges possible




```
-module(fp_http).  
-define(TYPE, tcp). % defines the type
```

```
callback_next_step(Args) when Args#args.moddata == undefined ->  
    % connected - request for payload  
    {continue, 1, SOME_PAYLOAD, true};
```



State data

```
callback_next_step(Args) when Args#args.packetrcv < 1 ->  
    % no packet received  
    {result, [{error, up}, timeout]};
```



Nb packet received

```
callback_next_step(Args) ->  
    % next packet: DO SOMETHING HERE WITH Args  
    {result, [{ok, result}, [SOME_RESULT]]}.
```



Do something clever here

Output module API

- Callback function called for each result
- Provides *Init/clean* functions for setup/management
- Format results for specific output
 - Stdout
 - File
 - Database
 - Streaming
 - ...

```
-module(out_stdout).
```

```
% init output at the beginning
```

```
init(_Scaninfo, _Options) ->  
    {ok, []}.
```

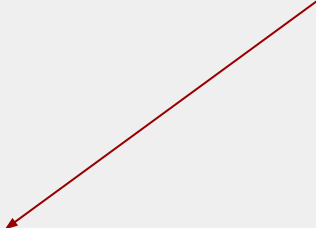
```
% clean output at the end
```

```
clean(_Obj) ->  
    Ok.
```

```
% new result to handle
```

```
output(_Obj, Msg) ->  
    io:fwrite("Result: ~p~n", [Msg]).
```

Format the data how you want it
No very advanced here

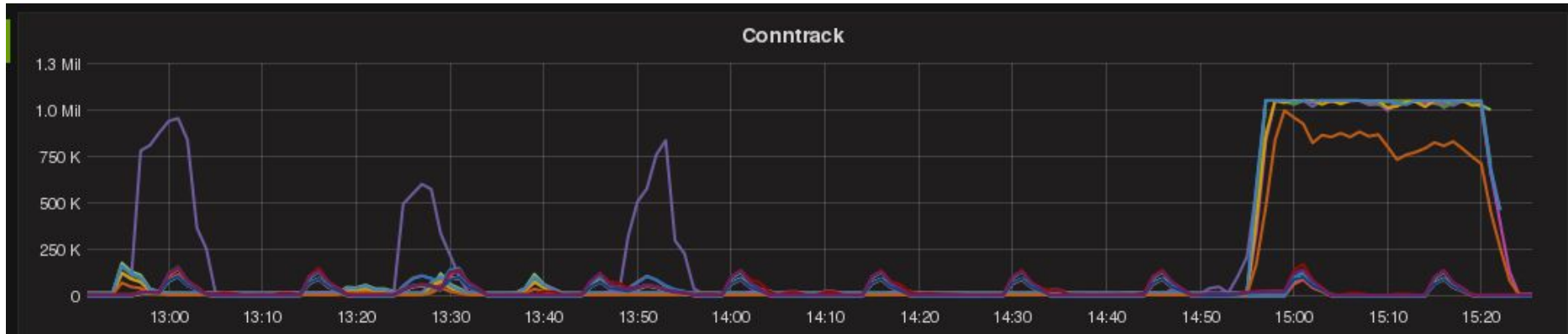


Scannerl in practice

Lessons learned using scannerl

- Fingerprinting can generate a lot of data -> *heap overflow*
- Bottleneck is usually not on the actor side but on the saving process
- Difficult to debug distributed application
- Internal communications impact the network too (management, results, ...)
- Cluster needs to be set up beforehand (ansible, ...)

Give love to your sysadmins



Scannerl performances

- Alexa top 1M
- 20 VMs
- 1 vCPU / 2GB RAM
- (vs 24 CPUs / 190G)

Zgrab: 48 minutes

Nmap: 8% after 14h

Scannerl: 4 minutes

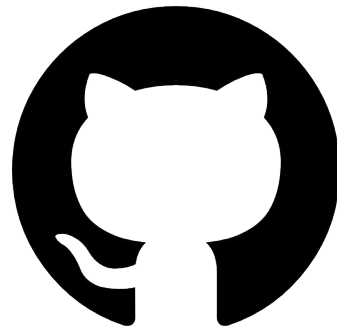


Some more results with scannerl

- HTTP Alexa top **1M - 1m30** with **60** VMs
- HTTPs Alexa top **1M - 11 min** with 10 VMs
- NTP (UDP/123) - **11M - 30 min** with 20 VMs
- SSH (TCP/22) - **20M targets - 30 min** with 10 VMs
- HTTPs (SSL/443) - **46M targets - 2h** with 30 VMs
- HTTP banner grabbing - **80M targets - 2h** with 30 VMs

Give it a try

- It is open-source on github (GPLv3)
- We are releasing more modules
- Mostly related to ICS/Scada
 - BACNet
 - ModBus
 - Fox
 - MQTT
 - MQTT over SSL



<https://github.com/kudelskisecurity/scannerl>

The future of scannerl

- Packaging for big linux distributions
- Increase the pool of available modules
- Get the community involved
- Improve documentation
- IPv6 port maybe
- Get a fancy logo

Summary

Scannerl

- Distributed (Fast)
- Flexible (fingerprinting and output modules)
- Advanced fingerprinting capabilities
- Open source
- Handles TCP/UDP/SSL
- Able to target IPs/domains
- Few dependencies (Erlang)

Thanks

... and give it a try, you might like it :-)

adrien.giner@kudelskisecurity.com

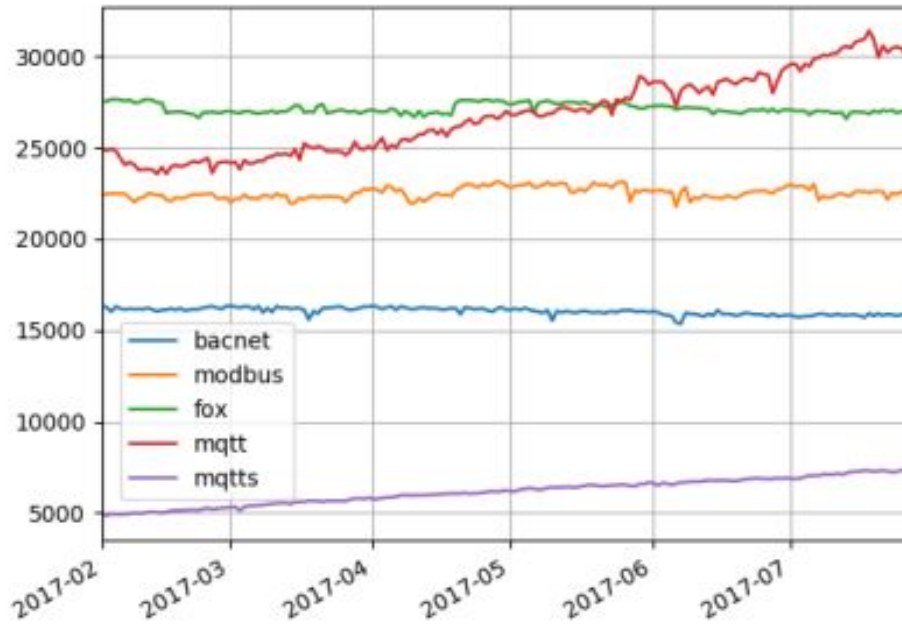
<https://github.com/kudelskisecurity/scannerl>

<https://research.kudelskisecurity.com/>



<https://research.kudelskisecurity.com/>

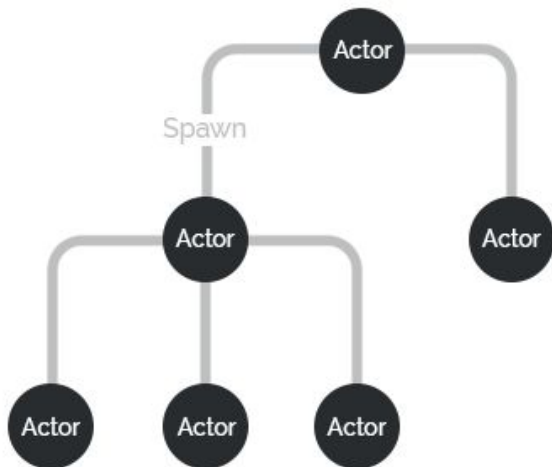
Some results from ICS



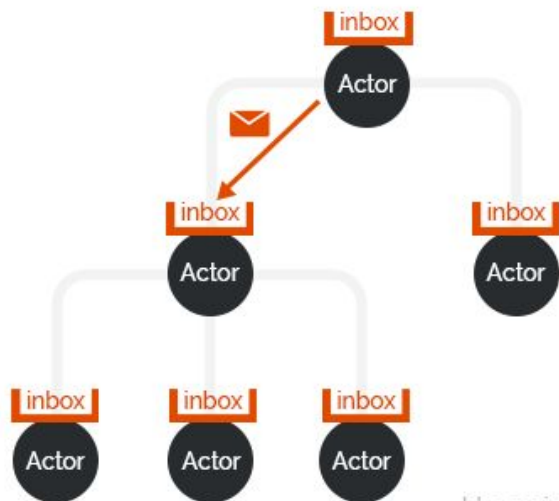
source: <https://research.kudelskisecurity.com/2017/10/24/6-months-of-ics-scanning/>

Actor model

Actors can spawn new actors building a hierarchy

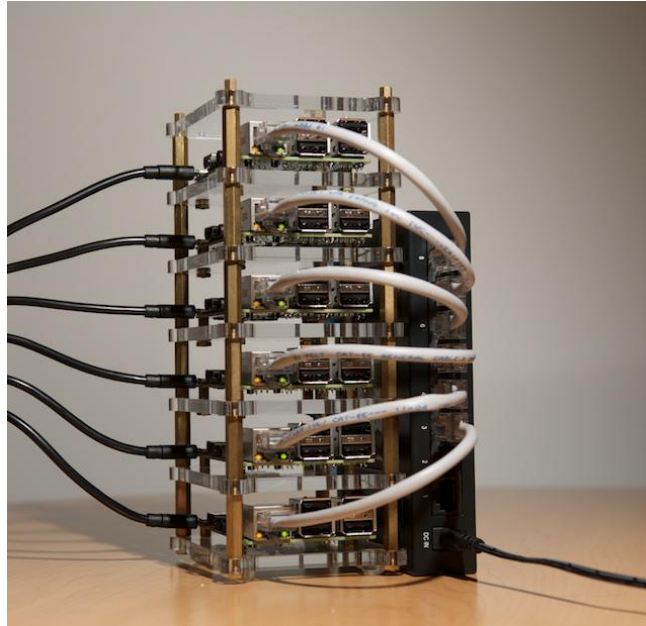


Actors only communicate through message passing. Messages are processed asynchronously one at a time.



blog.geist.no

RPi cluster



source: <https://www.jeffgeerling.com/project/raspberry-pi-dramble>

Nb VMs vs Duration

- Not linear
- Depends on the protocol (overhead, ...)
- Depends on data size (payload, response, ...)
- Bottleneck is often the saving process
- VMs performance
- More targets, more results
- More VMs, more management
- ...