Spoofing Nmap Service Detection

@alex.kropivny

March 11, 2014

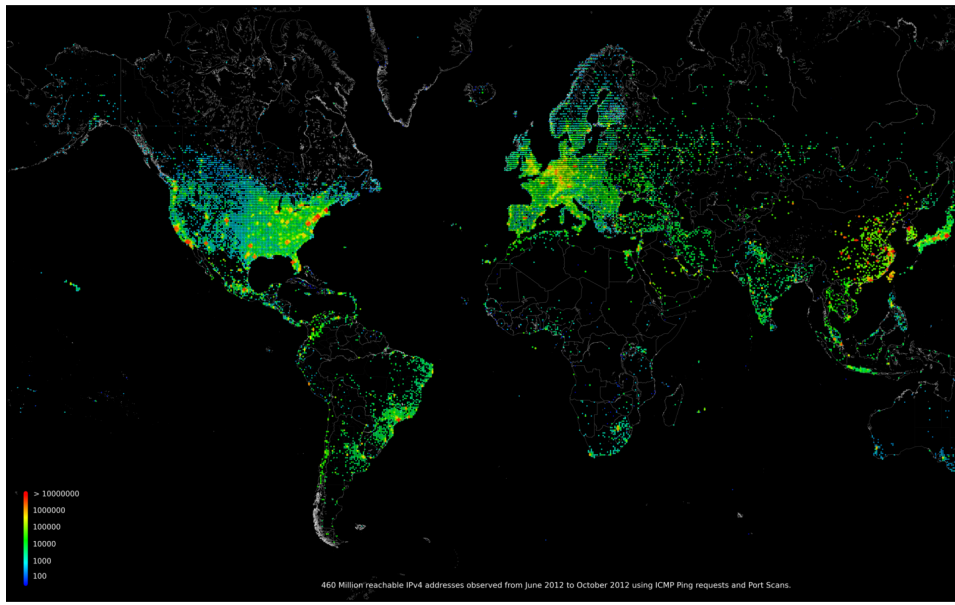
Purpose

Generating fake Nmap results for:

1. Fun
2. A possible countermeasure to data collection
3. Gives better understand of security tools

Privacy is Dead

- What data is collected?
- How is it used?
- Who handles it?
- ???????????



Possible Stopgap

- If it's not important enough for manual analysis...
- And if automated analysis sucks (it usually does)...
- Then a "Big Data generator" may mask our signals

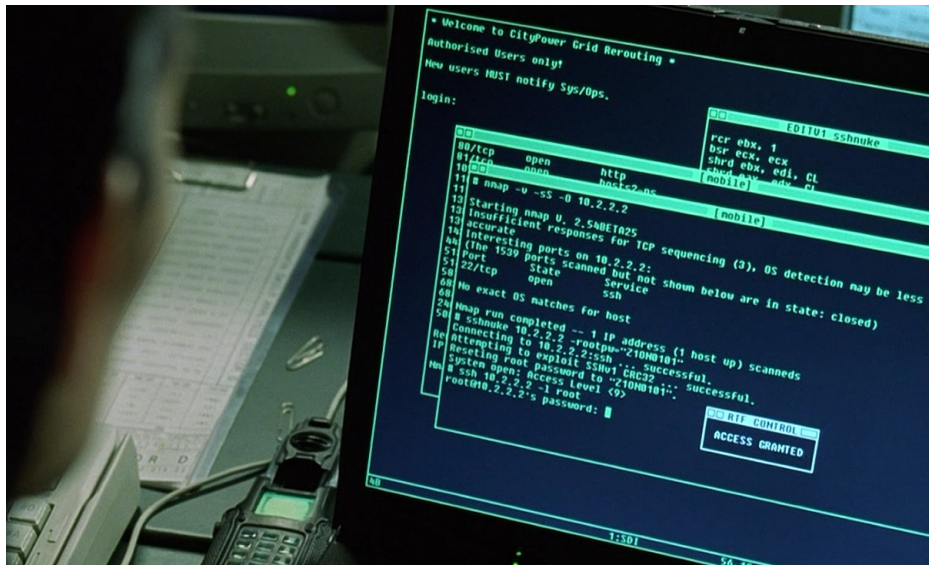
Know the Limits

- What assumptions were tools built under?
- Where do they start to fail?
- How does an adversarial scenario play out?

Target Selection

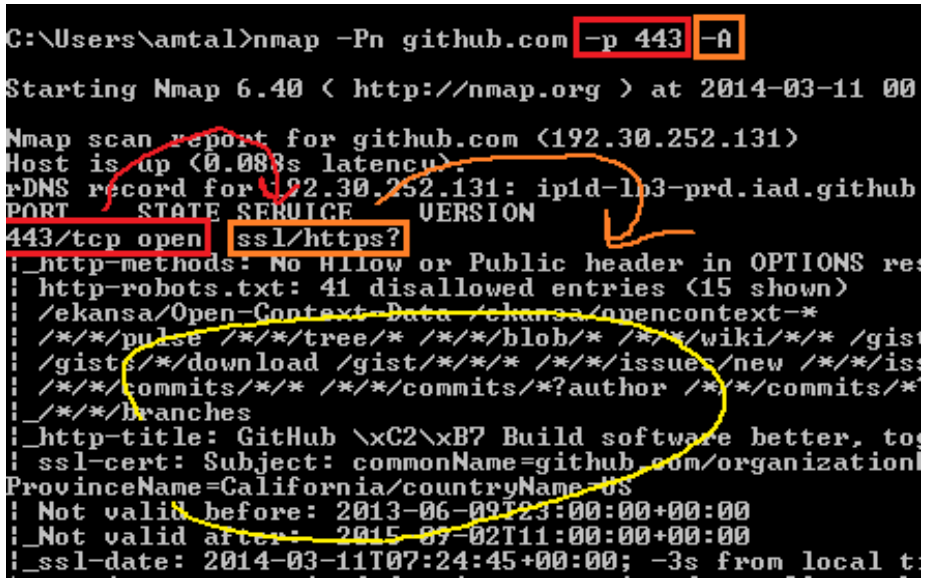
Nmap was chosen since:

- It's well known enough to be the first tool people reach for
- It's complex enough to have interesting bits of functionality



3 Stage Scan Process

- Port discovery (open/closed/filtered)
- Service detection (http/ftp/snmp/... + versions)
- NSE scripts (deep protocol details)



Spoof NSE Script Results?

- Uses a LUA interpreter that could be instrumented
- Uses some standard set of library calls for external access
- Could try and find "external" inputs that exercise maximum code paths through instrumented interpreter...

Conclusion: hard but potentially possible

Spoof Service Detection?

- Uses regular expressions and well defined probes: `https://svn.nmap.org/nmap/nmap-service-probes`
- Patterns defined in simple text format: `http://nmap.org/book/vscan-fileformat.html`
- Regular expressions can generate as well as consume text

Conclusion: low effort, high yield!

Architecture

Used Erlang to rapid prototype:

- Partially for trivial cross-platform support
- Partially for memory safety + error recovery on
- Weekend hack that can invert over 90% of the regexes in current nmap-service-probes

```
[amtal@foo src]$ wc -l *.erl
  16 nnop_app.erl
 123 nnop_parse.erl
  34 nnop_port.erl
 157 nnop_regen.erl
  28 nnop_sup.erl
  75 nnop_tree.erl
 433 total
```

Implementation

Simple:

- Parse nmap-service-probes file
- Generate valid matches for regexes found
- Pick a probe for a TCP port, and respond to it

Demo

```
nnop_port:run(2000,2099, [], ["ftp","telnet","http"]),
nnop_port:run(2100,2149, ["backdoor"], []),
nnop_port:run(2150,2200, ["telnet","ftp","http"], []).
```

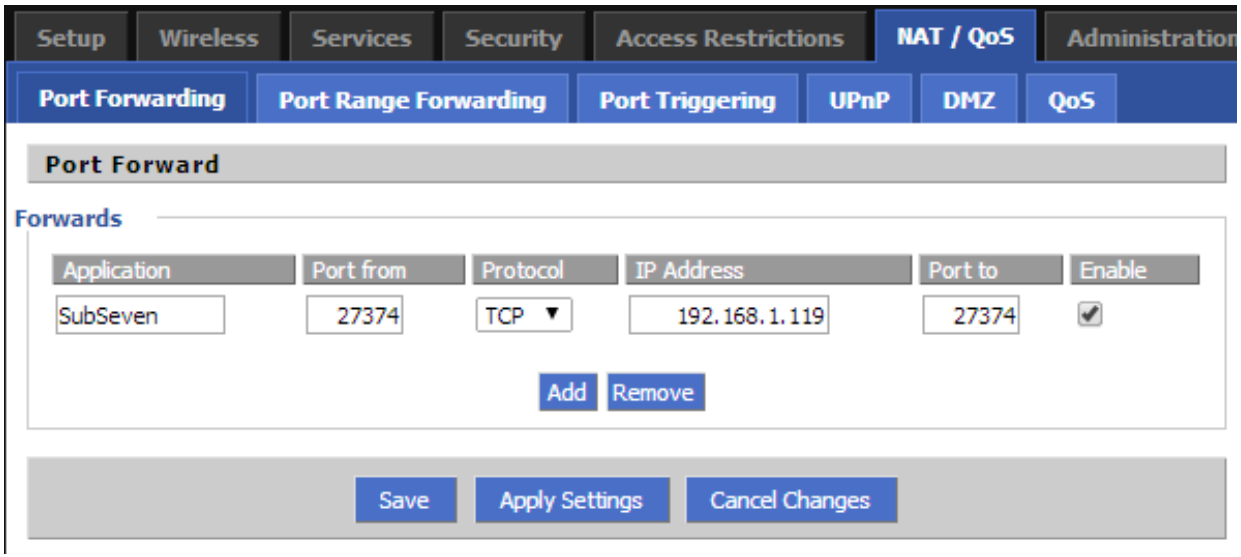
Heuristics

How do we actually "lie in a way which is not easily filtered"?

- Nmap groups services by http, ftp, telnet, etc...
- Multiple telnet or http servers are a red flag, avoid
- Results can pass visual inspection if ports are non-consecutive
- Could start using Nmap's service frequency data to allocate fake ports

Resourse Constrained Hosts

- Rewrite in C, while retaining memory safety and fault recovery? (No)
- How about tunneling to higher-resourse device?



Not Being Easy to Filter

- Regex output is currently deterministic (see version numbers)
- Could match services to ports they're expected to show up on

Messing with Humans

- Could randomize success
- Humans hate random failure in tools
- Alternatively, proxy a legitimate service but catch Nmap probes

Plot Twist: Someone Already Did It!

`http://www.saltwaterc.eu/portspoof-trolling.html`
(Found out morning before presentation.)

Portspoof Comparison (http://portspoof.org)

- Targets a single platform
- Lightweight C implementation
- Aims to remove Nmap as a tool altogether
- Significantly farther along

Different goals, far more functional and featureful

Big Data Generation

- Can be very easy (see `https://github.com/amtal/nnop`)
- Look for well structured collections of pattern matches

Tool Limitations

- Things break when you run into an adversarial situation
- Attempting to fix resulting problem would result in arms race