Xprobe2++

Low Volume Remote Network Information Gathering Tool

Yarochkin Fyodor, Ofir Arkin, Meder Kydyraliev, Shih-Yao Dai, Yennun HUang, Sy-Yen Kuo

National Taiwan University

July 1, 2009

Contents

Contents

Contents

Contents

Contents

The Xprobe Tool

Xprobe project started as remote fingerprinting tool to probe remote systems using ICMP protocol queries.
Other protocols support was added later.
Fuzzy fingerprinting mechanism was introduced to improve fingerprinting of unknown systems, or systems with modified network stack settings.
With new version of the tool we optimize algorithm of module execution, introduce information gain metrics.
Introduce new modules that work at application-layer.

Objectives

With this tool we ..

- Collect Remote Host Information(OS type)
- Selectively Collect Services information
- Selectively Collect Network Information(en route)
- Minimize network overhead and tool detectability by minimizing number of queries and selective probes execution.

What is Remote FingerPrinting?

Fingerprinting

- identifying type of software on remote system.

Passive vs. Active

Passive and Active Fingerprinting methods exist.

With Xprobe we implement active fingerprinting methods, i.e. we DO send packets to elicit responses.

Why we do it?

Attack Surface

Collecting information on remote system is the first step of identifying potential "attack surface" of a remote host.

Data Usage

Collected data could be used for correlation of IDS data, targetted exploitation of remote systems, network inventory etc.

Challenges

Modern Fingerprinting tool faces following problems

- Altered network stack settings lead to deviations from pre-collected signatures.
- itermediate devices may alter or normalize traffic.
- analyze intermediate device responses as they also may be used as jump-points for attacks
- dealing with "honeypots"

Remote FingerPrinting

What

We fingerprint different types or families of operating systems trying to identify the type and version of remote OS as close as possible.

How

We manually analyze the protocol specifications and identify "vague" spots. Then we analyze how different types of software (OS) are responding to these types of "vague" packets/requests. Signatures are created using the difference in responses. Further signature collection for "untested" platforms is automated.

What are we trying to improve here

Application Layer Probing

We expand the types of probes that we could use to cover Layer7 (application).
This gives us more probes to collect data from.
This also allows us to trigger responses from intermediate devices.

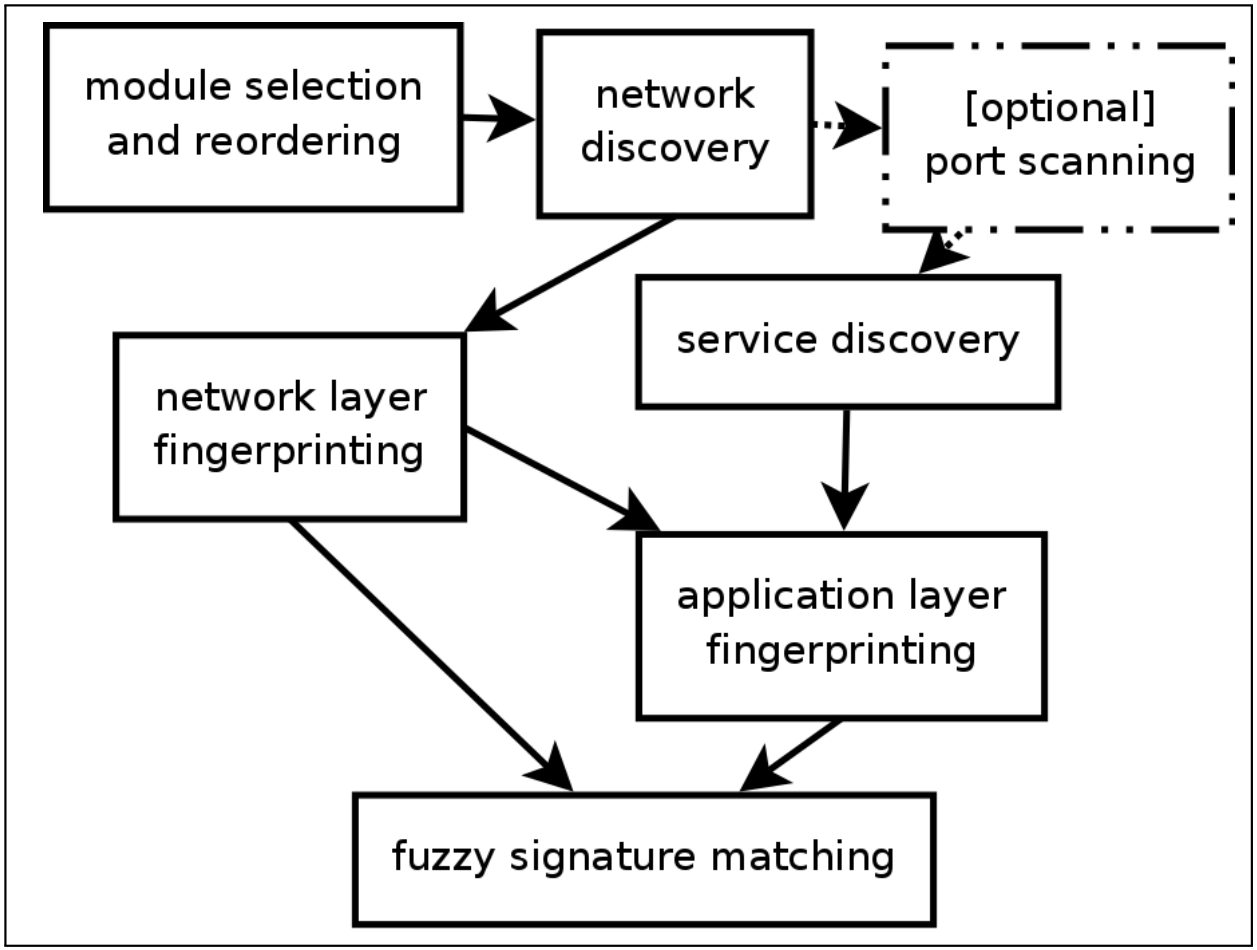
Minimizing probes

We minimize the number of 'probes' that we sent to the target by introducing new adaptive algorithm which does not execute 'probes' that could bring us no new knowledge about target system.

Benchmarking difficulties

It is difficult to perform a fair benchmarking of operating system fingerprinting tools because the success of fingerprinting process is heavily dependent on the network environemt, available protocols, running services and so on.
Different network configurations may lead to different number of "queries" to be required, type of target operating system (some queries could be more successeful in fingerprinting particular platforms.)

Overall Tool architecture



Modules and Information Gain

Prior fingerprinting

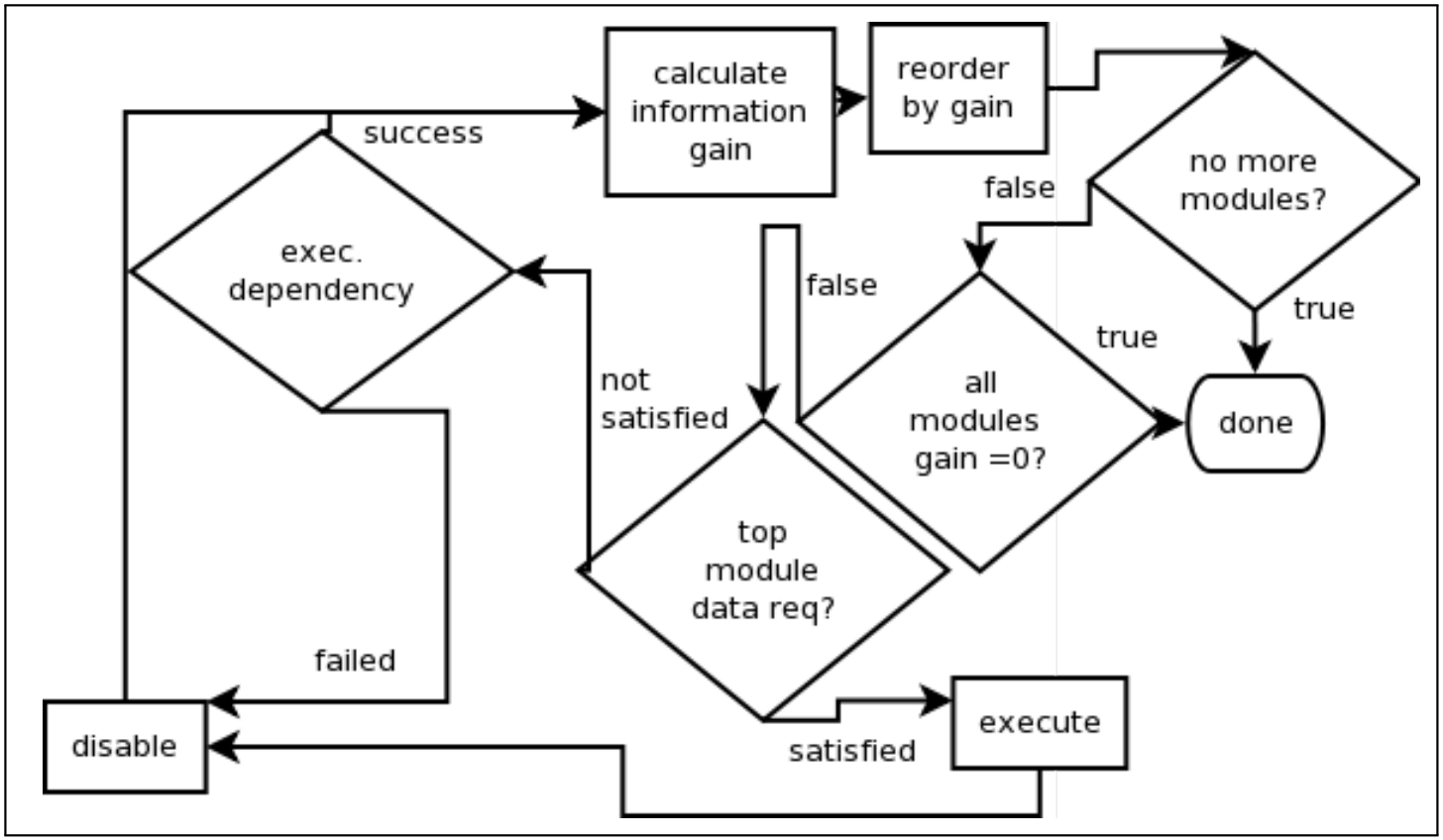
Information gain is calculated based on prior distribution of OS classifications, p(x_i) over all possible classifications.

After fingerprinting

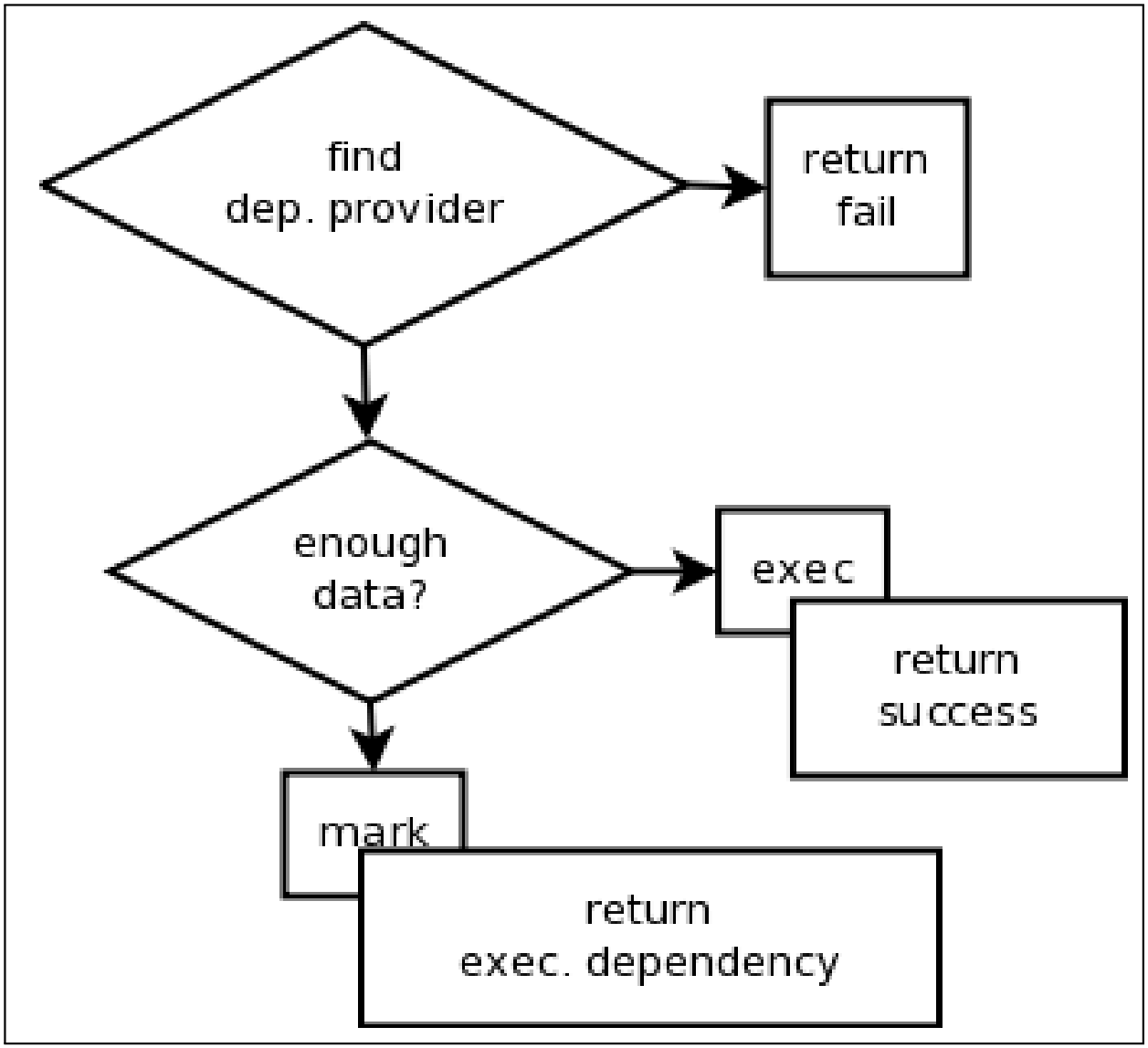
Information gain is calculated based on posteriori distribution of p(x_i—$test_i k$).

Information gain is re-calculated on every iteration.
Derrived from *"Towards Undetected Operating System Fingerprinting" by Greenwald et. al.*

Adaptive Module Execution Algorithm



Recursion on "execute dependency"



Why preluminary port scanning is not neccessary

Using this algorithm we can ensure that the only time we perform a probe of host reachability, or port status, when the fingerprinting process actually requires such data in order to execute module with non-zero information gain value.

Extending the tool with application-layer probes

Application-layer probes can be used as complementary tests to the network layer modules. (i.e. http probing)

Extending the tool with application-layer probes

Test type	Usable Protocol	Test
Directory Separator	HTTP	Win/Unx
New line characters	HTTP	Win/Unx
Special/reserved filenames	HTTP	Win/Unx
Root directory	FTP	Win/Unx..
Special characters (EOF,EOL		
Filesystem limitations	HTTP, FTP	..
Filesystem illegal characters	HTTP, FTP	..
Case sensitivity	HTTP, FTP	Win/Unx
Special filenames handling	HTTP, FTP	Win/Unx
Special files in directory	HTTP, FTP	Win/Unx
Binary file fingerprinting	FTP	Win/Unx

Other things that can be found at application-layer

```
bowzh lan # (echo -e "CONNECT 192.168.8.254:23 HTTP/1.0\r\n\r\n";cat) | nc [illegible].145 80
HTTP/1.0 200 Connection established
Proxy-agent: CacheFlow-Proxy/1.0

ÿûÿûÿÿÿÿCC

Authorised access only

This system is the property of HiNet

Disconnected IMMEDIATELY if you are no an authorised user!

Contact backbone@[illegible] (02)[illegible] for help


User Access Verification

Password:
```

Other things that can be found at application-layer

```
Tracing the path to www.orzteam.com (58.222.16.55) on TCP port 80 (http), 11 hop
s max, 791 byte packets
 2  114.45.208.254  157.892 ms  150.266 ms  151.822 ms
 3  168.95.71.62  151.827 ms  152.767 ms  166.531 ms
 4  220.128.4.118  155.682 ms  152.328 ms  151.788 ms
 5  * * *
 6  210.65.255.241  154.322 ms  160.305 ms  151.788 ms
 7  211.22.33.225  211.852 ms
    58.222.16.55 [unknown, ACK FIN]  109.508 ms
    211.22.33.225  315.486 ms
```

Dealing with honeypot systems

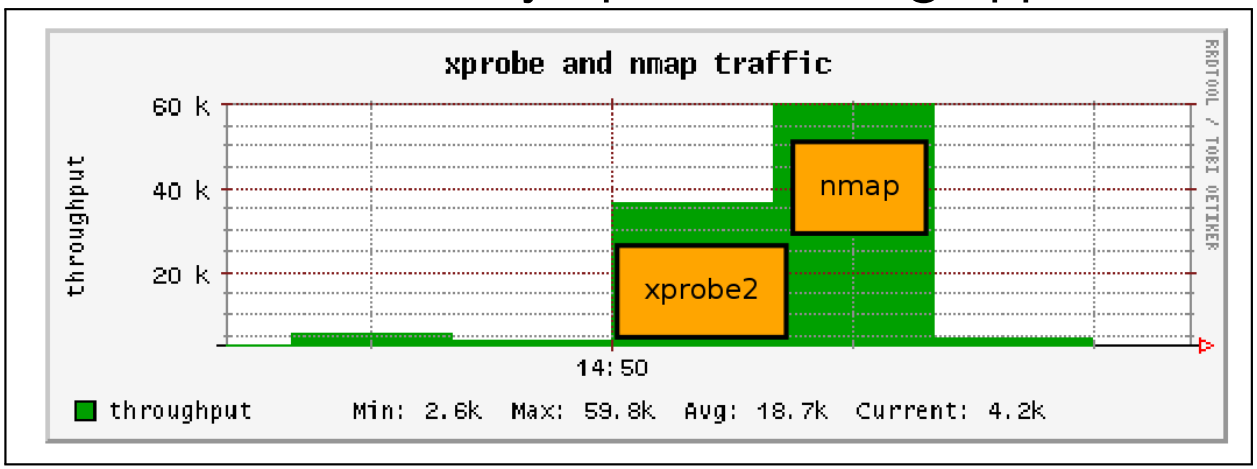
of course there's no way to identify all types of honeypots.
However we can make "educated guess" if the target systems are part of honeypot based on similarity of discovered services and systems, responses to malformed requests, similarity in application layer responses and so on.
This is still an open-research direction.

Tool Evaluation

We already mentioned that doing a fair-benchmarking of fingerprinting tools is hard to implement due to difference in methods using by different tools, which relay on availability of different network protocols.

Tool Evaluation

This illustration simply demonstrates that less network bandwith is consumed when "lazy" port-scanning approach is used.



Current developments

We are still working on ..

- Integrate python as module-development language
- Implementing prototyped tests in the tool
- More application-layer modules
- en-route discovery
- re-engineering of module-API for effective mass-scanning

Future Developments

Prototyping the decentralized network for data sharing between instances of the scanning tool.

- queries for existing results
- signatures and modules sharing
- statistics mining (improves signature matching)
- collaborative architecture

Tool Availability

Released under GPL license

project website: http://xprobe.sourceforge.net

Availability of presented version

- will be released via @sourceforge after 7/7
- git for "bleeding edge" code.

Quick Demo

(if I get my computer to show stuff here) it's a command line tool. so no impressive flashing-lights ;-)

```
[+] 13 modules registered
[+] Initializing scan engine
[+] Running scan engine
[-] ping:udp_ping module: no closed/open UDP ports known on
 172.16.131.209. Module test failed
[+] Host: 172.16.131.209 is up (Guess probability: 66%)
[+] Target: 172.16.131.209 is alive. Round-Trip Time: 0.010
27 sec
[+] Selected safe Round-Trip Time value is: 0.02054 sec
[+] SMB [Native OS: Windows 5.1] [Native Lanman: Windows 20
00 LAN Manager] [Domain: MSHOME]
[+] SMB [Called name: TATSU          ] [MAC: 00:0a:e4:29:3f
:4a]
[-] fingerprint:snmp: need UDP port 161 open
[+] Primary guess:
[+] Host 172.16.131.209 Running OS: "Microsoft Windows XP"
(Guess probability: 91%)
[+] Other guesses:
[+] Host 172.16.131.209 Running OS: "Microsoft Windows XP S
P1" (Guess probability: 91%)
[+] Host 172.16.131.209 Running OS: "Microsoft Windows XP S
P2" (Guess probability: 91%)
[+] Host 172.16.131.209 Running OS: "Microsoft Windows 2000
 Workstation SP3" (Guess probability: 87%)
[+] Host 172.16.131.209 Running OS: "Microsoft Windows 2000
 Workstation SP2" (Guess probability: 87%)
[+] Host 172.16.131.209 Running OS: "Microsoft Windows 2000
```

Conclusion

We consider these as our primary contributions:

- Fingerprinting process without pre-scanning ("lazy" portscanning)
- Adaptive algorithm, that utilizes information gain measure and data dependency for scheduling of fingerprinting tests.
- OS fingerprinting using application layer tests

Questions/Answers/Comments

We love questions :)