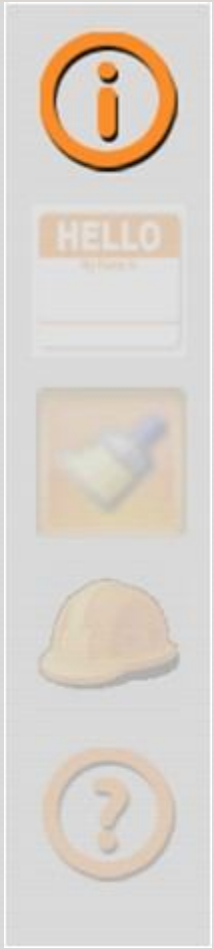
Let's Screw With nMap

DefCon 21, Las Vegas 2013



Hellfire Security

Gregory Pickett, CISSP, GCIA, GPEN

Chicago, Illinois

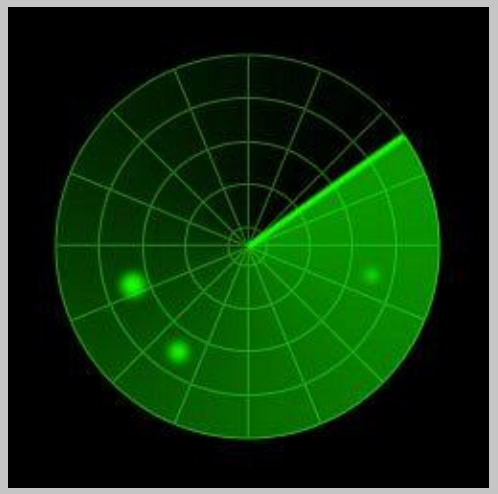
gregory.pickett@hellfiresecurity.com

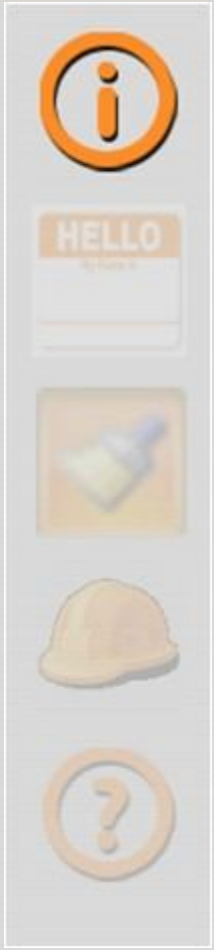
Overview

- Nosey Bastards!
- All About Packet Normalization
- Working It All Out
- Putting It Into Practice
- Finishing Up

Network Defenders

- We see scans and probes of our network every day
- From the inside and from the outside
- Everybody is targeting us
- Identifying our assets

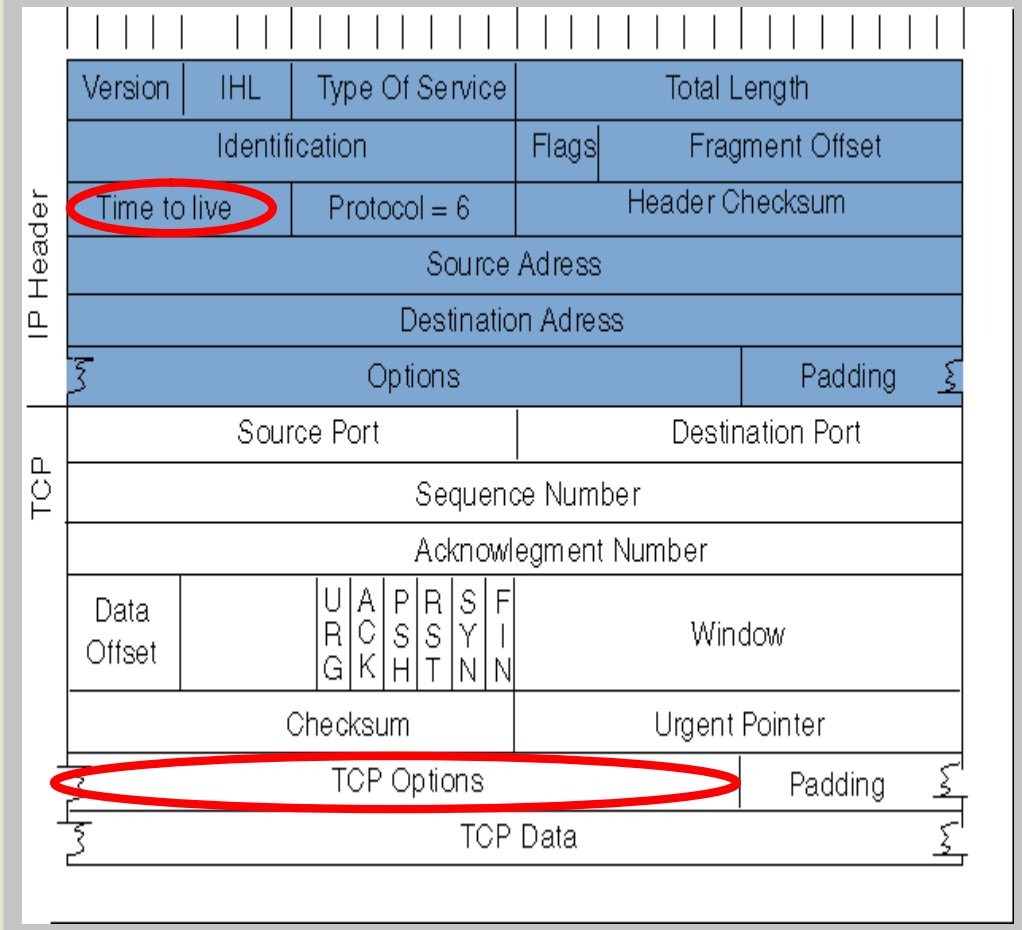




How They Do It

- Network stack implementation is highly discretionary
- Differences identify the operating system type and version
- Allowing Attackers to identify their targets
- By matching the headers of their target to known operating system implementations

If your target …



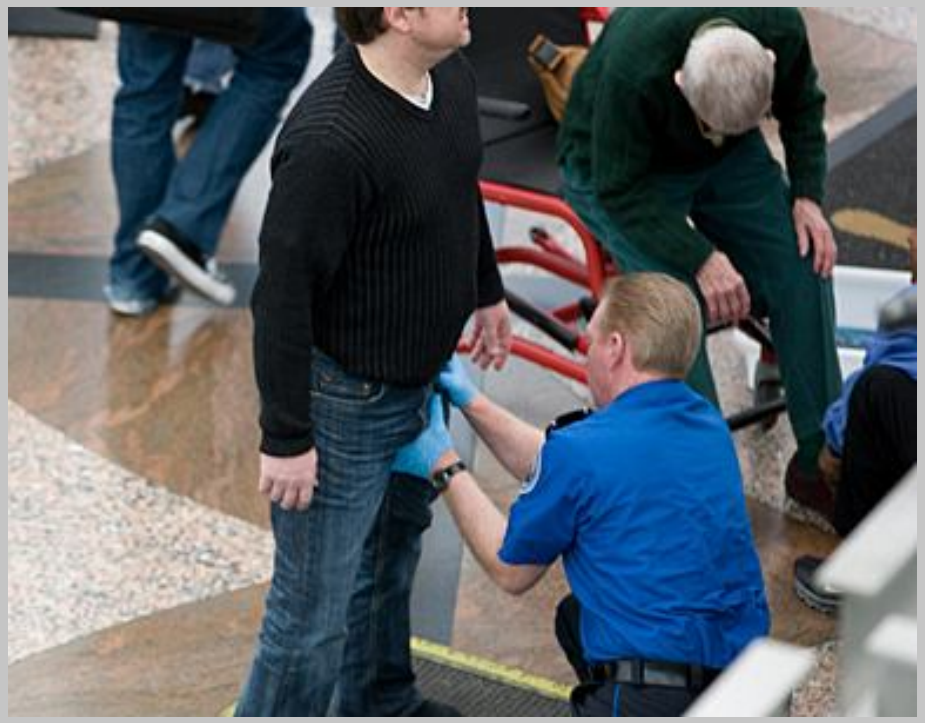
- Has a TTL of 128
- Uses the following options
  - MSS of 1460
  - Single NOP
  - Window Size 0
  - Single NOP
  - Single NOP
  - Ending SACK

… then it's likely a Windows 2003 Sever!

Implications

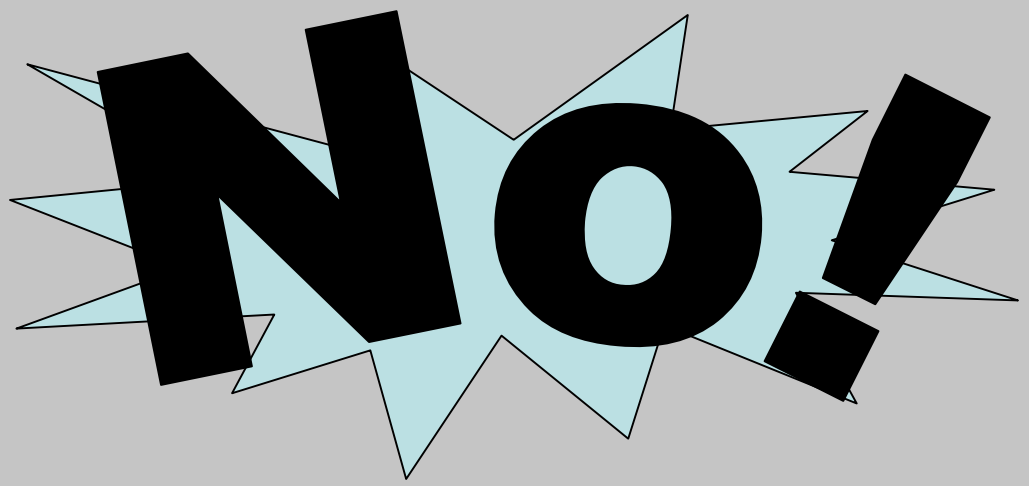
- If they identify your assets …
- They know their weaknesses
- How to attack them successfully
- Without triggering your sensors

TSA-Style patdowns …



It's fact of life

But does it have to be?



Why can't we …

- Remove the differences
- To remove their advantage
- Strip them of their ability to fingerprint
- To significantly reduce their chance of success

My Answer

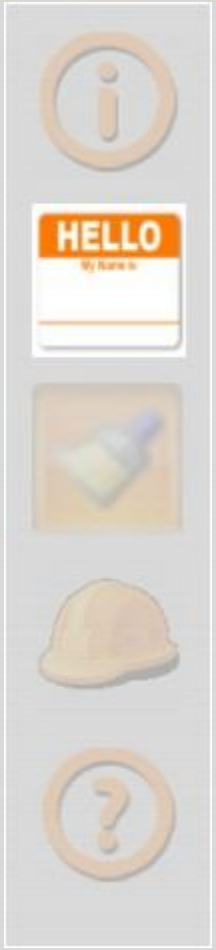
Packet

OK. What is packet normalization?

- Had anyone thought of this before?
- Not an entirely developed concept
- Many expressions but most incomplete …

Normalization vs. Scrubbing

- Scrubbing is to do away with; cancel
- Normalization is to make normal, especially to cause to conform to a standard or norm
- Both are seen in varying degrees

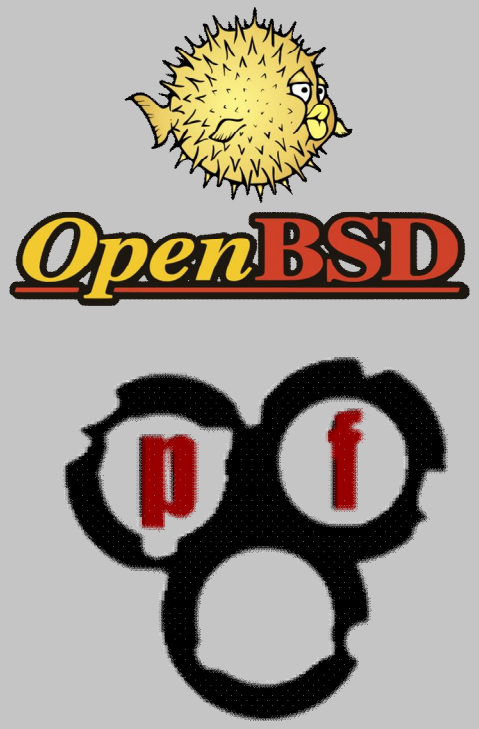


Scrubbing

- Used by a number of firewalls
  - Randomize IP ID
  - Clear IP DF
- Also …
  - Set IP tos/dscp, and ttl
  - IP Fragment Reassembly
- Primarily Concern
  - Policy Violations
  - Abnormal Packets
  - Abnormal Flows

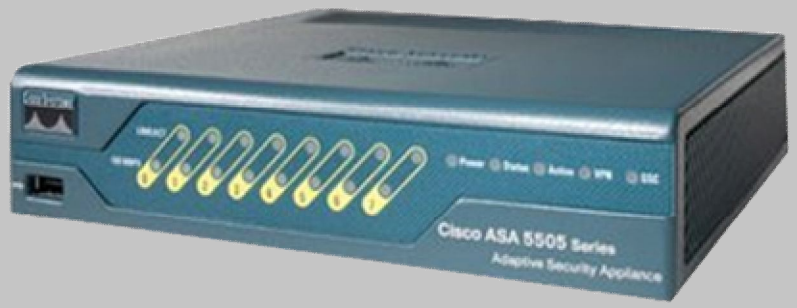
Scrubbing

- Custom patch for netfilter
  - Random IP ID
  - Randomize TCP Timestamp
  - Randomize TCP SEQ
  - Clear IP tos/dscp
  - IP TTL Tinkering
- Developed by Nicolas Bareil
- Mentions fingerprint prevention
- Host Only



Scrubbing

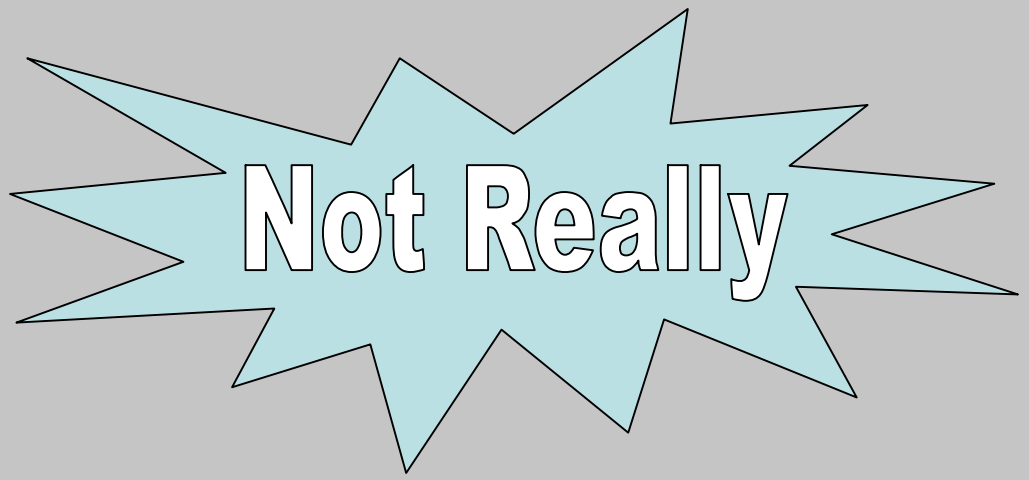
- Used by some network devices such as Cisco ACE and ASA
  - Random TCP SEQ
  - Clear TCP Reserved, and URG
  - Clears TCP Options
  - Minimum IP TTL
- Fragment Reassembly too …
- Primarily Concern
  - Policy Violations
  - Abnormal Packets
  - Abnormal Flows



Incoming Normalization

- Used by IPS and IDS devices
  - IP Fragment Reassembly
  - IP TTL Evasion
- Primarily Concern
  - Detect Attacks
  - Detection Evasion

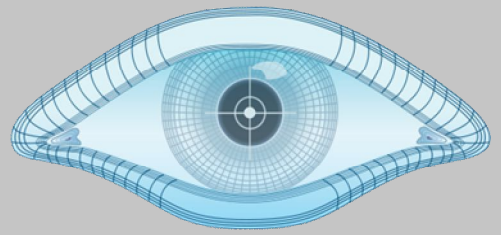
Outgoing Normalization?



Fingerprinting Process

- TCP, UDP, and ICMP probes are sent
- Compile results into fingerprint

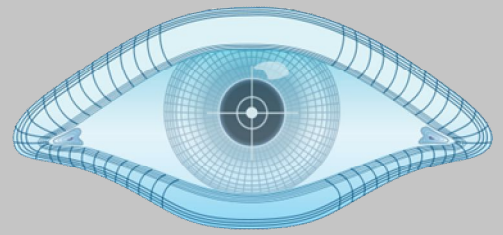
```
Fingerprint Linux 2.6.17 - 2.6.24
Class Linux | Linux | 2.6.X | general purpose
SEQ(SP=A5-D5%GCD=1-6%ISR=A7-D7%TI=Z%II=I%TS=U)
OPS(O1=M400C%O2=M400C%O3=M400C%O4=M400C%O5=M400C%O6=M400C)
WIN(W1=8018%W2=8018%W3=8018%W4=8018%W5=8018%W6=8018)
ECN(R=Y%DF=Y%T=3B-45%TG=40%W=8018%O=M400C%CC=N%Q=)
T1(R=Y%DF=Y%T=3B-45%TG=40%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=N)
T3(R=Y%DF=Y%T=3B-45%TG=40%W=8018%S=O%A=S+%F=AS%O=M400C%RD=0%Q=)
T4(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T5(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T7(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
U1(DF=N%T=3B-45%TG=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)
IE(DFI=N%T=3B-45%TG=40%CD=S)
```



- Compare against database
- Identify operating system

Where to Start?

- Nmap fingerprint database
- What about other fingerprinting tools?
  - xprobe2
  - amap
  - Vulnerability scanners … Nessus, Et. Al
- Best to disrupt any existing patterns



Scrubbing

- Clear out any unnecessary values
  - IP ToS/DCSP/Traffic Class Cleared
  - IP ECN Cleared
  - TCP URG Flag and URG Pointer Cleared
- Randomize anything that you can
  - IP ID
- IP TTL/HOP Limit? TCP Options?

Our Regular Normalization

Normalizing
(IP Time-To-Live / Hop Limit)

- Make some assumptions
  - Originally Well-Known TTL
  - Decrements Only
  - Traveled < 32 hops
- Back into Original Starting TTL
- Estimate number of hops traveled
- Recalibrate current TTL
- Using Starting TTL of 255

Normalizing
(IP Time-To-Live / Hop Limit)

```
If <= 32 traveled = 32-current Then ttl = 255 - traveled
If <= 64 traveled = 64-current Then ttl = 255 - traveled
If <= 128 traveled = 128-current Then ttl = 255 - traveled
Else ttl = current
```

- Start with the lowest well known TTL first!
- Several exceptions to this normalization …
- Will be discussed later

Normalizing (TCP Options)

- Assumptions
  - Only Few Well Known Options Needed
  - Order is unimportant
- Requirement …Values can't be changed
- Read necessary options
- Discard the rest
- Rewrite options in proper order
- NOP … till the end of the options

Normalizing (TCP Options)

- Options selected … And their order
  - MSS
  - Window
  - SACK
  - MD5 … if present
- After processing …

MSS = 1460
Window = 0
SACK
NOP
NOP
NOP

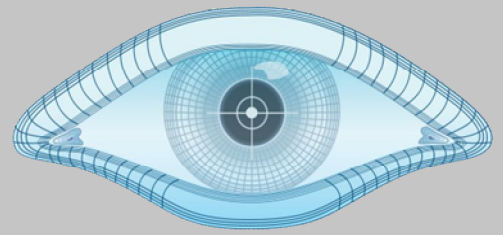
Making Everyone Look the same

Putting It All Together

With IDGuard

Selecting The Platform

- Identified Suitable Hardware
  - Already Modified By Others
  - Documentation Available … Mikrotik Routerboards
- Identified Suitable Operating System
  - Available Base
  - Writeable File System …OpenWrt
- Best to develop in a VM first!



Building the Development Environment

- Download Debian v6.0 Net-install CD-ROM
- Build a VMWare VM
- Install rcp100 from Sourceforge
- Configure rcp100 routing functions

Building the Development Environment

```
Before RCP100 Installation
--------------
apt-get update
apt-get install linux-headers-$(uname -r)
apt-get install ncurses-dev, bridge-utils, make, g++, rsync



RCP100 Installation
--------------
wget http://downloads.sourceforge.net/project/rcp100/rcp100/rcp100-0.99.4.tar.bz2
mkdir /opt/rcp
tar -xjvf rcp100-X.Y.Z.tar.bz2 /opt/rcp
cd /opt/rcp

./configure
make
su
make install
exit

--> RCP "README" for more information



After RCP Installation
--------------
Copy contents of rcp folder "operating System" subfolders into matching VM locations
```

Configuring the Development Environment

```
Configure VM Adapters
--------------
First: Local LAN (Bridged)
Second: Virtual LAN (Host-Only) --> Same As "BackTrack"


Configure RCP100 (via http://localhost)
--------------
Interfaces
-       Eth0: Local LAN (192.168.2.60)
-       Eth1: Virtual LAN (9.0.0.10)
Routing
-       RIP on both interfaces
-       Redistribute RIP
DNS
- 8.8.8.8, 8.8.4.4
```

Deploying the Kernel Module

- Download IDguard v0.50
- Install IDGuard

Deploying the Kernel Module

```
IDGuard Installation
--------------
wget http://downloads.sourceforge.net/project/idguard/idguard_v0.50_linux_networking.tar.bz2
mkdir /opt/idguard
tar -xjvf idguard_v0.50_linux_networking.tar.bz2 /opt/idguard
cd /opt/idguard

./configure
su
make
sudo insmod idguard.ko
```

OK … What worked?

I am really tired of those nosey bastards!

What Didn't Work

- ToS/DCSP/Traffic Class Clearing
- ECN Clearing
- URG Flag and URG Pointer Clearing
- IP ID Randomization
- DF Clearing

… the Scrubbing

What Worked

- TTL Standardizing
- TCP Option Standardizing

… the Normalization

End Results

Operating System	Unprotected	Protected
Windows 7	Microsoft Windows 7|2008	Allied Telesyn AlliedWare
Windows Server 2003	Microsoft Windows 2003	Allied Telesyn AlliedWare
Ubuntu Desktop 11.10	Linux 2.6.X|3.X	Cisco IOS 12.X
Red Hat Enterprise Linux 6	Linux 2.6.X|3.X	D-Link embedded

Other Effects

- Nmap
  - Network Distance
- Other Fingerprinting
  - xprobe2
  - Nessus …
- Other Tools
  - ping
  - traceroute

Deploying to Hardware

- Purchase the hardware from a local vendor
- Download OpenWrt kernel image with an embedded initramfs
- Setup dhcp & tftp netboot environment
- Connect to the routerboard
- Configure routerboard for DHCP
- Back up RouterOS
- Prepare the OpenWrt images
- Flash it

Deploying to Hardware

Flashing with OpenWrt/DebWrt

To flash the system you need to replace both the kernel and the root filesystem, which are two different images know if there is a different way...). To get to a command line you will need to boot a working OpenWrt kernel+in

Booting OpenWrt from the network

- **Get a working OpenWrt kernel image with an embedded initramfs**. To build this image, in the OpenW Make sure to use OpenWrt trunk (kernel >= 2.6.39) and **not** the stable version, backfire. Backfire's kernel h not suitable for this job. OpenWrt will build a lot of images; the one you will use is "openwrt-ar71xx-nand-vm RouterBoot can boot (it expects to read elf headers). Alternatively, you can download a ready working image
- **Setup dhcp & tftp netboot enviroment**: Once you have the image, you will need to configure a tftp & dhc scope of this article.
- **Connect the routerboard with rs232 and ethernet**: Connect the routerboard's RS232 with your computer **Eth1/POE** port (do **not** use Eth2-Eth5 as they are connected to internal switch and it is offline on bootloadin On your computer, setup a terminal emulator to get access to the serial port. A very popular one is minicom execute something like that: "sudo minicom -D /dev/ttyUSB0", where ttyUSB0 here is the serial device (ass that it will need to be run as root, unless you have access to the serial device (which is usually achieved by
- **Configure routerboard to use DHCP instead of BOOTP**: When the setup is ready, power up the routerbo routerboard will try to boot only with BOOTP protocol which is not what we want. So to change it enter the s bootup process.

Demonstration

Challenges

- Authorized Activity
- Other Methods
  - Banners and Direct Query
  - Identification Through Layer-7

Challenges

- Authorized Activity
  - Scanners
  - Management Platforms
- Resolution
  - Exclude them …

Challenges

- Banners and Direct Query
  - Windows Networking Available
  - Application-Layer Query
  - OS Details in Reply
- Resolution
  - Perimeter Network
  - Internal Network

Concerns

- Connectivity
  - Fragmentation
    - Upstream
    - Downstream
  - TTL Attenuation
  - TTL Special Uses
- TCP Options Sensitivity?
- Link-Local Routing Protocols

Concern

- Upstream Fragmentation
  - IP ID Randomized
  - “Fragmentation Needed” ICMP Message Received
  - Host is confused
  - Keeps sending original packet
- Resolution
  - Clear DF

Concern

- Downstream Fragmentation
  - Each fragment given a different IP ID
  - Destination can't be reassembled
- Resolution
  - End-Point Switch Placement
  - Exclude Fragments

Concern

- TTL Attenuation
  - Packet travels more than 32 hops
  - Packet TTL is continually extended
  - Routing Loop occurs
- Resolution
  - End-Point Switch Placement

Concern

- TTL Special Uses
  - TTL recalibrated
  - TTL never runs out
  - Traceroute fails
- Resolution
  - Exclude ICMP Echo Requests

Concern

- Link-Local Routing Protocols
  - TTL of 1 for RIP packet
  - TTL of 255 is abnormal
  - Packet is malformed
- Resolution
  - Exclude routing protocols

Concerns

- Performance
- Break Something
  - Poorly Coded Applications
  - What else?

Benefits

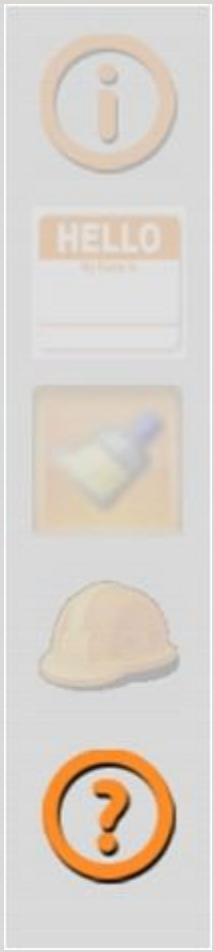
- Shields from …
  - Casual Attackers
  - Automated Assaults
  - Oblique Threats
- Protects …
  - Unmanaged
  - Unpatched
  - Unhardened
- Defeats … canned exploits

What's Next

- More Platforms
  - Open-Source Router Firmware
  - Linux-Based Switches
- Production Trials
- Talk to vendors

Final Thoughts

- Accurate target identification is key to a successful attack
- Identification that is way too easy for an attacker to perform
- Let's change that with fingerprint prevention
- I've proven that it can be done
- Now, we just have to make it happen

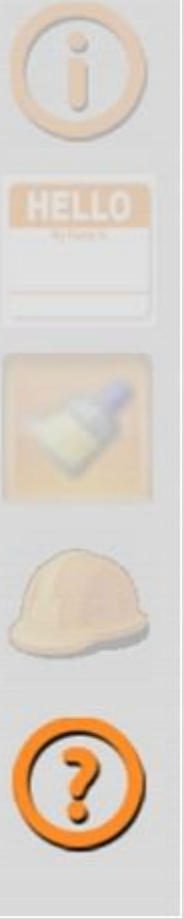


Proof of Concept

IDGuard v0.50 for Linux-Based Networking

- Fingerprint obfuscation support
- IPv4, IPv6, and TCP normalization
- Authorized Activity Exclusions
- Linux Kernel Module Implementation

SHA256 hash is **e97b2c8325a0ba3459c9a3a1d67a6306**
Updates can be found at http://idguard.sourceforge.net/

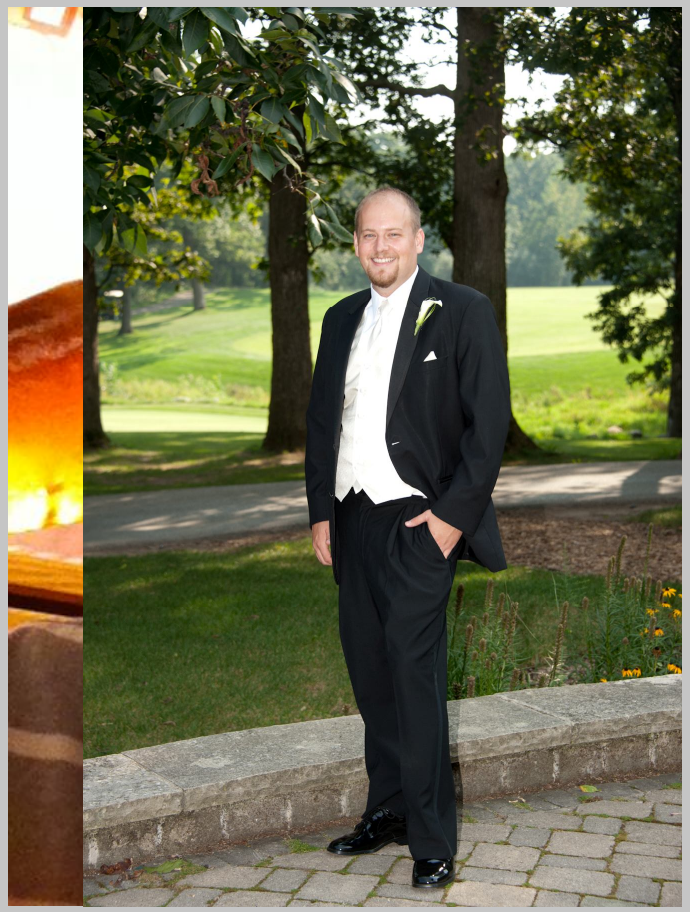


Links

- http://www.wisegeek.com/what-is-packet-mangling.htm
- http://www.openbsd.gr/faq/pf/scrub.html
- http://www.linuxsecurity.com.br/info/fw/PacketManglingwithiptables.doc
- http://chdir.org/~nico/scrub/
- http://www.cisco.com/en/US/docs/security/asa/asa82/configuration/guide/conns_tcpnorm.pdf
- http://www.cisco.com/en/US/docs/interfaces_modules/services_modules/ace/v3.00_A2/configuration/security/guide/tcpipnrm.pdf
- http://www.sans.org/reading_room/whitepapers/intrusion/packet-level-normalisation_1128
- http://nmap.org/book/osdetect-methods.html
- http://rcp100.sourceforge.net
- http://wiki.hwmn.org/w/Mikrotik_RouterBoard_450G
- http://downloads.openwrt.org/snapshots/trunk/ar71xx/openwrt-ar71xx-generic-vmlinux.elf
- http://downloads.openwrt.org/snapshots/trunk/ar71xx/openwrt-ar71xx-generic-rootfs.tar.gz
- https://sites.google.com/site/guenterbartsch/blog/myfirstlinuxkernelmodule
- http://www.farlock.org/nslu2/openwrt-non-standard-module-compiling/

Special Thanks

- Aditiya Sood
- Kenny Nguyen and E-CQURITY
- Kathy Gillette
- Nick Pruitt




Q&A