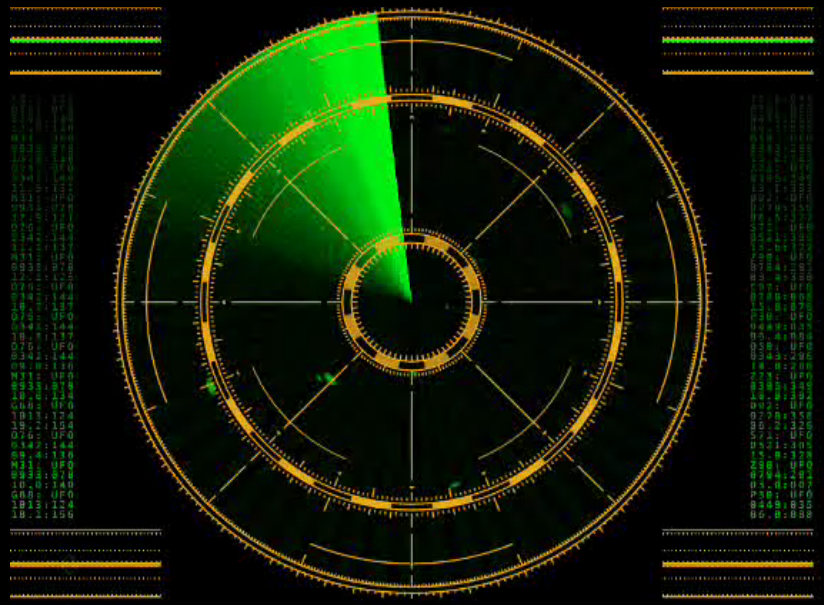
IOSMap: TCP and UDP Port Scanning on Cisco IOS Platforms



Rob VandenBrink

June 2009

GIAC GSEC gold, GSNA, GAWN, GCIA gold, GCIH, GPEN

Objective

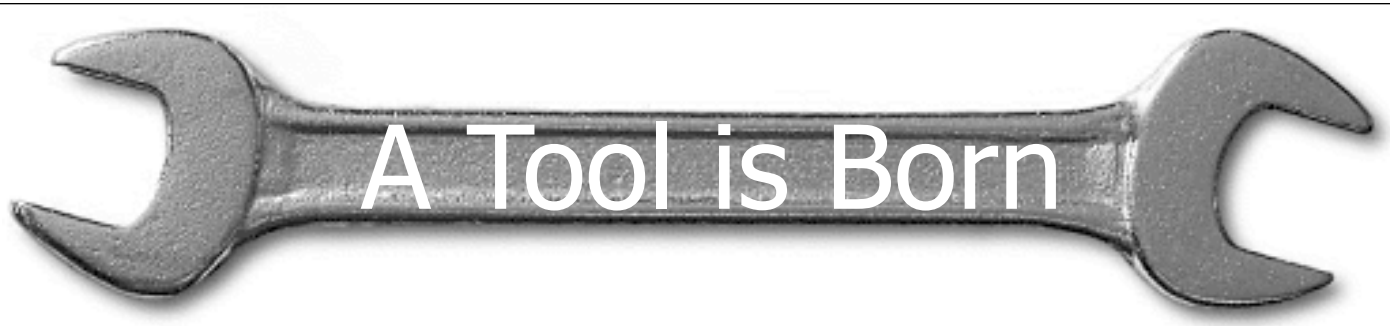
- In a real business engagement (a case of illegal activity by an IT department member), I was tasked with the following list of requirements:
  - **Need** to scan for open ports (TCP and UDP).
  - **Must** remain covert from the Server / Workstation team within the IT group.
  - **Preference** was to use equipment local to "the problem"
  - **Must** use only corporate gear (no consultant laptops, no external software)

The Solution

- We chose to implement port scanning on a local Cisco IOS router
- The scripting language on IOS is TCL (Tool Command Language)
- Initial scans were simply run from the TCLSH/ CLI command line
- "Anything worth doing twice should be scripted"
- We were able to extend the process of port scanning to include generic packet capture

The Outcome

- The person suspected was in fact found to be sharing material illegally
- Their access to sensitive company information also put corporate data at risk
- Subsequent packet captures (done from the same IOS platform) allowed us to capture some actual datastreams, so we were not reliant only on port scans for evidence
- All this was handed over to the HR Group for further action.



A Tool is Born

- The basic tool used for the engagement was a 25(ish) line script in TCL
- After the engagement, I decided to make IOSmap a formal tool
- Extended the initial script to accept command line input, in roughly the same format as popular Linux and Windows port scanners
- Extended the script output to similarly match that of popular tools

TCP Port Scanning

- Used the native TCL "socket" command to implement a TCP connect scan
- No device configuration changes are required
- Minimal impact on router CPU and memory
- A more complex methodology is possible (similar to the UDP scan on next slide), but is not implemented on this release of IOSmap

UDP Port Scanning

A more complex process than TCP Port scanning:

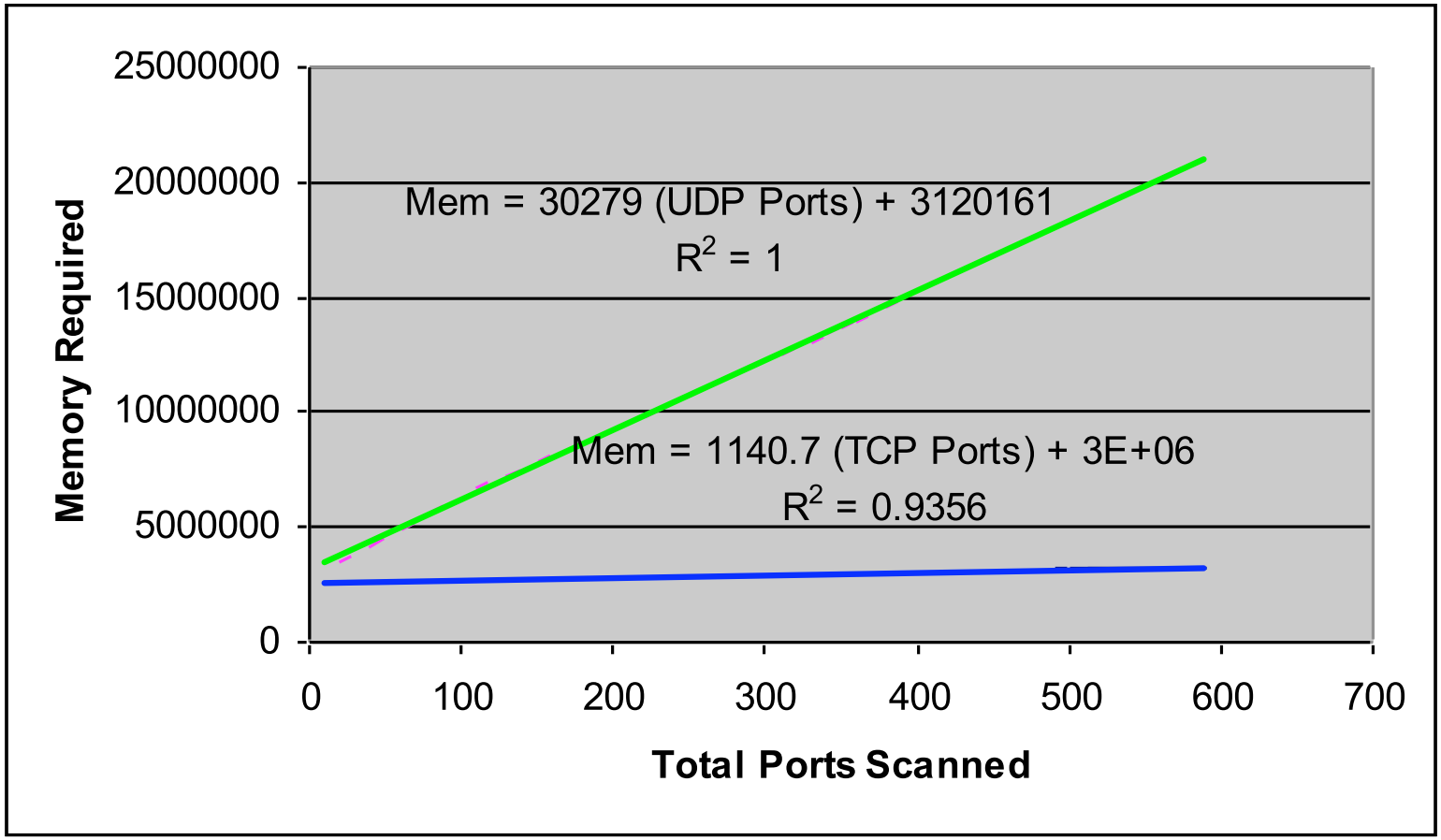
- Send a UDP "probe" packet to the target ip/port
- Capture the return traffic - 4 cases to consider:

Packet Returned	**Port State Deduced**
ICMP Port Unreachable is returned (ICMP Type 3, Code 3)	**Port is closed -** This packet comes from the target host
Any other ICMP Unreachable (ICMP Type 3)	**Port is filtered** – this packet comes from firewall type devices
UDP Packet returns **from** target port	**Port is open**
Nothing returns	**Port is Open/Filtered -** This is the most common scenario

Platform Impact – Memory Usage

- Memory usage was found to be a straight line function, dependent only on total ports scanned
- True for both TCP and UDP scans, though UDP scans take **significantly** more memory
- For instance, on a /24 network (254 addresses), a UDP scan of 5 ports will exhaust the memory on a typical 256MB router platform

Platform Impact – Memory Utilization



Platform Impact – Change Control

- UDP port scanning requires changes to the running configuration and running status:
  - Access list is created to define interesting traffic.
  - IP SLA function is used to generate UDP "probe" packets
  - Debug IP Packet (exec command) is required to capture the packets of interest – this will affect CPU utilization
- All of this means that UDP port scanning on an IOS platform will generally involve **getting permission first.**

IOSmap - Syntax

HOST DISCOVERY:

-P0 Treat all hosts as online - skip Ping test

-SL List hosts and ports to scan

SCAN TYPE:

-sP Ping scan only <ICMP ECHO>

-sT TCP Connect Scan

-sU UDP Scan

--reason: display the reason a port state is reported as such

PORT SPECIFICATION:

-p <port ranges> Specify ports to scan.

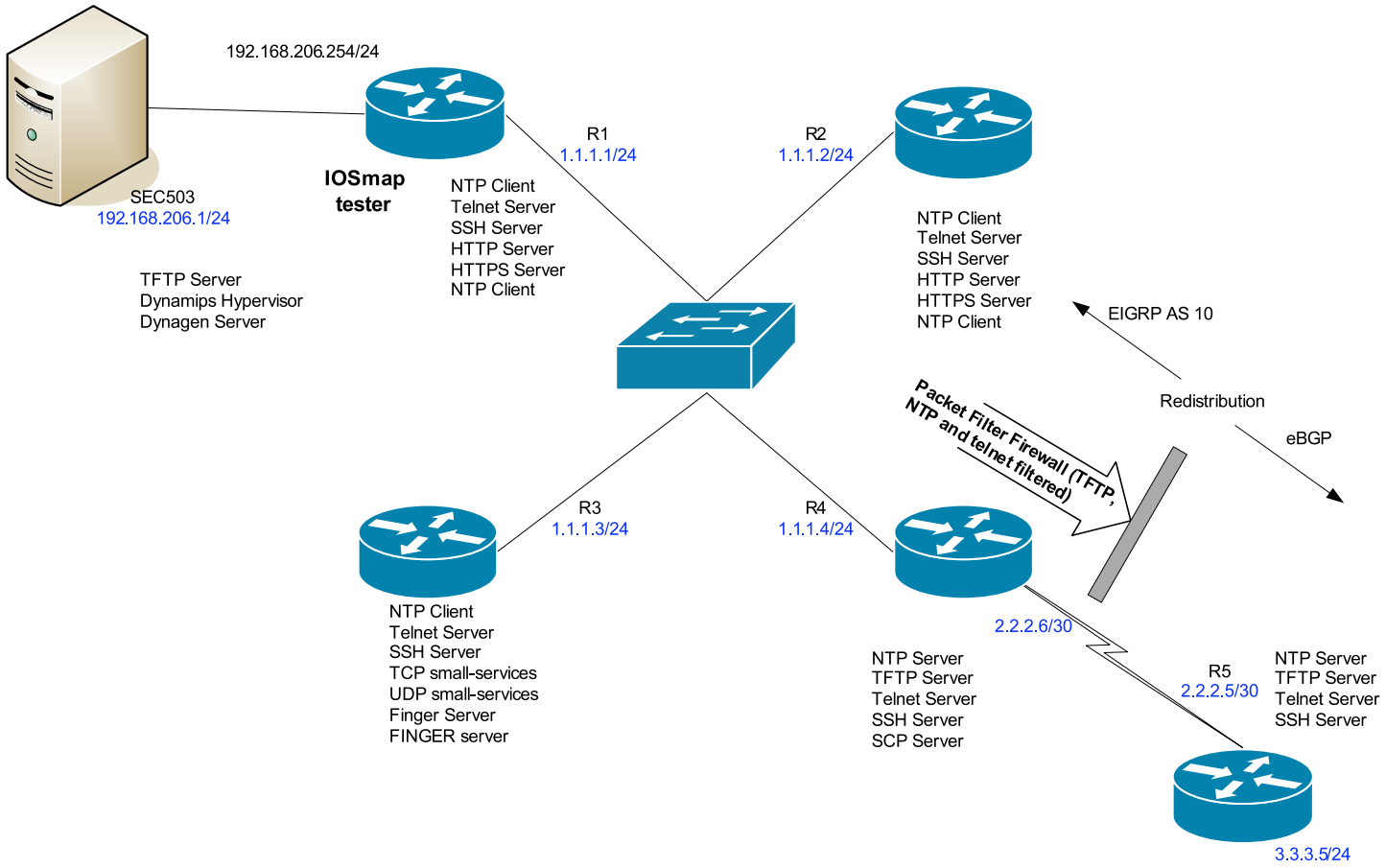
-p22 Scan port 22

-p22,23,135-139,445 Scan ports 22, 23, 135, 136, 137, 138, 139, 445

TARGET SPECIFICATION:

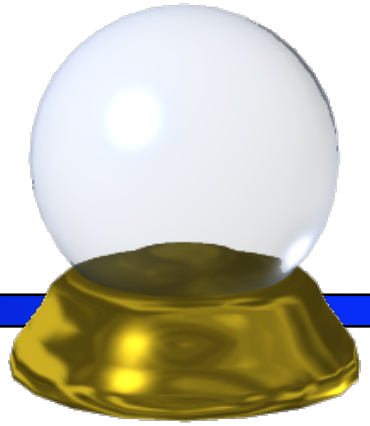
CIDR, IP range and single IPs are all a supported - comma delimited

Demonstration Network



Demonstration

Future Development



- "Top N Ports" scans
- --packet-trace option (display packets)
- Support for IPv6
- Support for more TCP state detection (will require packet capture similar to current UDP approach)
- IP protocol detection (may not be possible)
- No RAW packet injection is available (yet), so IP Protocol Scans, SYN scans, script options and other more advanced functions aren't practical
- **Suggestions for IOSmap development are welcome**

Summary

- I've found IOSmap to be a useful tool, and have used it in subsequent engagements.
- I've learned that code is more than 80% input validation, parsing and formatting, and less than 20% actual "function"
- This tool is posted on:
  - Cisco Community Site: http://forums.cisco.com/eforum/servlet/EEM?page=eem&fn=script&scriptId=1621
  - http://www.sourceforge.net/projects/iostools
- Thanks for your time and interest!