

Bringing the Pieces Together: an Architecture for Network Scan Mitigation

Erwan Le Malécot, Yoshiaki Hori, Kouichi Sakurai

Kyushu University

2007

Introduction (Cont'd)

Issues

- ▶ When detecting a scan, usually too late to prevent the scanner from collecting the associated information.
- ▶ Use of stealthy scanning techniques. . .

Solution?

- ▶ Act apriori to reduce the impact of network scans → proposal of an architecture for network scan mitigation.
- ▶ Purpose: limit the accuracy of the information an attacker can learn by scanning a protected network + slow down the scanning process (e.g. worm propagation).

TCP/IP Network Scanning

Modus Operandi

- ▶ Attackers, mainly interested in knowing: what hosts are up, what operating systems they are running and the list of their TCP/UDP open ports (potential targets).
- ▶ Easiest way: send probes (normal/crafted packets) to the targeted hosts and analyze the replies, if any.
- ▶ For instance, ping scanning: send ICMP echo requests.

TCP/UDP Port Scanning Techniques [de Vivo et al., 1999]

- ▶ Try to establish a regular TCP connection to the port.
- ▶ SYN, FIN, Xmas Tree, Null scanning or UDP segments.



Filtering

Use of Firewalls

- ▶ System administrators mainly rely on firewalls to control the traffic exchanged by their networks.
- ▶ Initial firewalls matched packets against simple criteria (e.g. protocol, src/dst IP address) → packets accepted or rejected according to predefined static rules.

Extended Capabilities

[Lowth, 2004]

- ▶ Nowadays, most firewalls are also stateful: can keep track of connections and match packets accordingly.
- ▶ Provide additional matching criteria (e.g. time of day, random choice) + modularity and extensibility.



Additional Techniques

Tarpitting

[Liston,]

- ▶ Tarpit: answers to any connection attempts to unused IP addresses in a way that slows down the initiator.
- ▶ Sets the TCP receive window size to 0 → forces the initiator to stop sending data and periodically ask if it can resume the transfer, until it timeouts. . .

Packet Scrubbing (Packet Normalization)

[Smart et al., 2000], [Handley et al., 2001], [Watson et al., 2004]

- ▶ Packet scrubber: removes ambiguities from TCP/IP traffic.
- ▶ Can be used to prevent TCP/IP stack fingerprinting attacks and some IDS evasion techniques.

Description

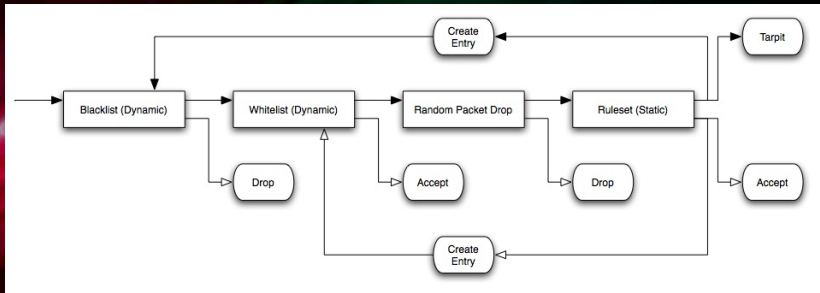
Overview

- ▶ Architecture: located at the border of the network to protect, based on several modules that successively process all incoming/outgoing traffic.
- ▶ Outgoing traffic: only to be processed by a packet scrubber + auxiliary filtering by system administrators if needed.

Incoming Traffic

- ▶ At first, also processed by the packet scrubber.
- ▶ Then: chain of modules that drop, pass or tarpit the traffic according to selected rules: Dynamic Blacklist, Dynamic Whitelist, Random Packet Drop, Static Ruleset.

Processing of Incoming Traffic



- ▶ Static Ruleset: states allowed destinations (forwarding + new entry in the Dynamic Whitelist) and forbidden ones (tarpitting + new entry in the Dynamic Blacklist).

Discussion

Mitigation Capabilities?

- ▶ Stealthy port scanning techniques + TCP/IP stack fingerprinting prevented by the packet scrubber.
- ▶ ICMP echo requests either dropped or replied, by either authentic hosts or the tarpit → determining authentic hosts is harder; idem for scanning based on connection attempts.
- ▶ Automated scans are slowed down and further incoming traffic from suspicious sources is dropped.
- ▶ Random Packet Drop used to produce an effect similar to greylisting in e-mail spam defense (genuine users usually retry on failure, whereas malicious users do not).
- ▶ Mitigation of more advanced attacks!

Prototype

Strategy

- ▶ At first, implementation of the architecture using existing off-the-shelf components.
- ▶ Packet scrubber → provided by PF (OpenBSD Packet Filter); other modules → use of Netfilter framework (Linux).
- ▶ Goal: behavior similar to a transparent filtering bridge, so that the architecture could be placed anywhere.

Status...

- ▶ Functional implementation, testing on a small testbed.
- ▶ But, components not specifically designed for the task we use them for, still need for some optimization.

Conclusion

Summary

- ▶ Combining efficiently several previously proposed techniques to achieve network scan mitigation.
- ▶ Prevention of several network scanning techniques + reduction of the efficiency of remaining ones.

Future work?

- ▶ Finish the implementation of the proposed architecture.
- ▶ Deploy it in a real environment to test its practical efficiency and performances.
- ▶ Determine optimal values for several parameters of the system (e.g. % of packets to randomly match).

Questions-answers

Any questions?

Bibliography I



de Vivo, M., Carrasco, E., Isern, G., and de Vivo, G. O. (1999).

A Review of Port Scanning Techniques.

SIGCOMM Computer Communication Review, 29(2):41–48.



Handley, M., Paxson, V., and Kreibich, C. (2001).

Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics.

In *Proceedings of the 10th conference on USENIX Security Symposium (SSYM'01)*, pages 9–9, Berkeley, CA, USA. USENIX Association.



Liston, T.

Tom Liston Talks About LaBrea.

<http://labrea.sourceforge.net/Intro-History.html>.



Lowth, C. (2004).

Kernel Korner: The Hidden Treasures of iptables.

Linux Journal, 2004(120):8.



Smart, M., Malan, G. R., and Jahanian, F. (2000).

Defeating TCP/IP Stack Fingerprinting.

In *Proceedings of the 9th conference on USENIX Security Symposium (SSYM'00)*, pages 17–17, Berkeley, CA, USA. USENIX Association.



Watson, D., Smart, M., Malan, G. R., and Jahanian, F. (2004).

Protocol Scrubbing: Network Security Through Transparent Flow Modification.

IEEE/ACM Transactions on Networking, 12(2):261–273.

