An Innovative UDP Port Scanning Technique

Sumit Kumar and Sithu D. Sudarsan, *Member IACSIT*

***Abstract*—In this paper, we address the challenge of speeding up UDP port scan. We propose a novel scanning technique to perform significantly faster UDP port scanning when the target machine is directly connected to port scanner by utilizing multiple IP addresses. Our experiments show that both Linux and Windows systems could be scanned faster. In particular the speed up in tested Linux systems using our scanner are about 19000% in comparison with traditional port scan method.**

***Index Terms*—Asynchronous scanning, port scan, reconnaissance, UDP scanning.**

I. Introduction

A. Port Scanning

Identifying open ports to determine services running on a device or system is a very important task, as any port that is not needed, if left open, adds to the vulnerabilities and hence becomes a potential cyber security threat of the device or system in question.

Port scanning is a method in which a machine is scanned for open Transmission Control Protocol (TCP)/Universal Datagram Protocol (UDP) ports. It is used during various security audits, vulnerability assessment and penetration testing against a server or device which is commonly referred to as Server under Testing (SUT). During testing, a port which is active for communication is referred to as an open port. A port which is closed is referred to as a closed port. All the ports for which no state could be determined, due to lack of valid response from the SUT are considered to be filtered ports.

A port can be marked as filtered port due to multiple reasons. A firewall actively blocking all the packets coming to a particular port or set of ports could be one reason. Another reason in case of UDP ports could be that the service running on UDP port is configured to respond for only a particular input or type of input.

B. User Datagram Protocol (UDP)

UDP is a communication protocol which works over the Internet Protocol (IP). It is a connection-less and stateless protocol. When a packet is sent to a UDP port, three responses are possible, which is different from the way TCP ports respond. If there is no service running on the UDP port, the system will reply back with "ICMP port unreachable" message [1]. ICMP stands for Internet Control Message Protocol. If a service is running, and UDP packet is not a valid query packet with respect to the application protocol, it may silently drop the packet without giving any response. If the UDP packet is a valid packet with respect to application protocol and to which a response is expected, the application running on UDP will send back a response packet. Accordingly, port scanning may report any UDP port to be in a closed, filtered or open state.

The rest of the paper is organized as below: Section II provides the background followed by related work in Section III. Our solution is outlined in Section IV followed by implementation details in Section V. Our evaluation forms Section VI followed by conclusions in section VII.

Manuscript submitted July 20, 2014; revised October 14, 2014.

The authors are with ABB Global Industries and Services Limited, Bangalore, India (e-mail: sumit.k@in.abb.com, Sudarsan.sd@in.abb.com).

II. Background

UDP port scanning is generally much more restrictive to perform as compare to TCP port scanning. This is because of the fact that TCP port scanning techniques can use the three-way handshake and combination of various TCP flags to determine open, closed and filtered ports. However, in case of UDP port scanning, port scanners like nmap [2] generally send empty UDP datagram to the ports of the target machine. If the port is closed, the SUT replies back with "ICMP Port Unreachable" message. If the port is open, it may or may not reply, depending upon the application layer service configuration [3], [4]. Further, a firewall may block "ICMP port unreachable" messages, causing the port scanner to report all closed ports as "filtered" and thereby decreasing the efficiency of port scanning.

If the firewall configured on the SUT does not block ICMP port unreachable messages, one of the ways to differentiate between open and closed ports is by using "ICMP port unreachable" messages. In this case, all the filtered ports can be considered as open ports.

Several operating systems, like Linux and Solaris implement a rate limiter, limit the number of "ICMP port unreachable" messages to one packet per second [5] for a remote IP address. Due to this, a remote scanner may require more than eighteen hours (65535 seconds) to perform a complete UDP port scan. Accordingly, if the port scanner utilizes two IP addresses to scan the port range, it causes a performance boost of 100% and takes nine hours, instead of eighteen, which is still a significant amount of time.

A. ARP Poisoning

ARP Poisoning is a network attack in which the attacking machine sends ARP Response to all or selected ARP Requests given by the victim machine [6], [7]. ARP stands for Address Resolution Protocol. Once the ARP cache of victim machine is poisoned, all the IP packets directed to destination IP address from the poisoned will be received by the attacking machine [8]. A two-way ARP poisoning is called ARP Man-in-the-Middle attack, i.e., in which two IP

DOI: 10.7763/IJFCC.2014.V3.332

addresses are poisoned against each others' ARP cache entry by the attacking machine, thereby forcing both of the machines to send IP packets to the attacking machine.

III. Related Work

UDP scanners typically send some empty UDP datagrams [9] to the UDP ports of SUT and decide the port status based on the UDP packet response. Nmap and Nessus [10] send application specific UDP probes to increase the efficiency in discovering the state of UDP packet.

A. Nmap Scanner

Nmap is the de-facto tool for port scanning [5]. It has various scanning options for TCP scan, including TCP SYN, TCP Connect, TCP Null, Xmas Scan and Mailmon scan. It also allows fine control over scanning rate by allowing user to specify the minimum and maximum packet rate. It has operating system detection capabilities as well. However it does not have enough scanning options for UDP scan.

For a typical UDP Scan, it sends an application specific UDP probe packet to the UDP port of SUT to determine its state. However, this scanning technique is limited to only those ports for which the probes are available which forms only a small fraction of 65535 ports. This technique will increase the efficiency in discovering port state. However, it will not decrease the scanning time. Where application specific probes are not available, nmap sends empty datagram as probe packets to the UDP ports.

B. ScanRand Scanner

ScanRand is a tool created by Dan Kaminsky to quickly scan a large number of ports [11]. It basically separates the sender and receiver mechanism which allows it to perform very fast scanning. The sender process sends TCP SYN packets without waiting for responses or timeout. The receiver process sniffs for the SYN-ACK or RST-ACK responses to categorize `open' and `closed' ports. This greatly decreases the scan duration as sender do not have to wait for timeouts to send the next packet.

Due to stateless nature of the tool, it can perform very fast TCP port scanning. However, it does not address the UDP port scanning.

IV. Proposed Solution

The port scanning techniques discussed so far take a long time because of the rate limiter present on the SUT, which could be as low as one "ICMP port unreachable" message per second. If the scanner utilizes two IP addresses instead of one, it causes performance boost of 100%, allowing the complete scan to complete in approx. nine hours. This situation is feasible when device or entity level UDP scanning is needed and direct connection between the scanner and SUT is possible.

Accordingly, the time taken during a complete scan can be represented as,

$$T \propto \frac{(p-o)}{n} \quad (1)$$

where,

T is the time taken to complete a scan

p is the number of ports to be scanned

o is the number of open ports which reply back

n is the number of IP addresses used by the port scanner

The time complexity is $O((p-o)/n)$.

During vulnerability assessments, robustness testing, black box testing, and penetration testing, generally all the ports are scanned. So p can be considered to be constant. For a typical system very few ports are left open, about ten ports or 0.015% of total ports, so o can be ignored.

Now, for relatively small number of n, and excluding storm issues, packet loss, processor load, packet re-transmission, and other performance factors, the time complexity can be represented as $O(1/n)$.

In this paper, we introduce an innovative UDP port scanning technique. This technique requires SUT to be directly connected to the scanner without any Network Address Translation (NAT) or Port Address Translation (PAT) enabled networking device in the connection path. We have carried out evaluation of this technique by directly connecting the SUT to the scanner.

The scanner has the following modules:

- ARP Poisoner
- Packet Sender
- Sniffer

A. ARP Poisoner

The ARP Poisoner module responds back to every ARP request made by the SUT with scanner's own MAC address. This allows the scanner to acquire all the IP addresses on the network as shown in Fig. 1. Poisoning the ARP ensures that the scanner receives all the packets sent by the SUT to scanner.

The ARP Poisoner does not however reply back for gratuitous ARP requests to avoid IP collision situation.

B. Packet Sender

Packet Sender sends UDP packets, also called probe packets, to the SUT. In these packets, the Ethernet source address is the address of the scanner and Ethernet destination address is the address of SUT. In IP header, the destination IP address is the address of the SUT and the source IP address is any valid IP address of the same subnet of SUT. If a gateway is configured on SUT, the source IP address can be any valid IP address.

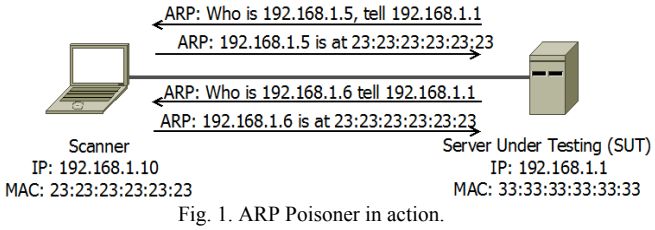

Fig. 1. ARP Poisoner in action.

In our context a valid IP address is one which can be assigned to a system and communication can be made on such an IP address. This excludes, IP addresses like the broadcast, network node, loopback, and Class E IP addresses.

The source port on the UDP header is the port under

scanning. The destination port can be any random port. Algorithm 1 describes the procedure for Packet Sender.

A record is kept of the source port, destination port and source IP address for comparison with the received packets. This allows the Sniffer to differentiate between response packets and packets coming from UDP clients which may be present on SUT.

Algorithm 1 Procedure for Packet Sender

```
begin
proc PacketSender(subnet, tgtip) ≡
srcip := firstip
for dstPort := 0 to 65535 step 1 do
if srcip = tgtip then srcip := srcip + 1 fi;
if srcip ∉ subnet then srcip := firstip fi;
srcPort := random(1024, 65535);
sendPacket(srcip, tgtip, srcPort, dstPort);
srcip := srcip + 1 od
end
```

where
subnet is the sequence of all valid IP addresses in subnet. *tgtip* is the IP address of SUT.

C. Sniffer

The Sniffer captures all the UDP and ICMP packets coming to the scanner. It has a list, referred here as port-list, of all the ports under scanning. When it receives a ICMP port unreachable packet, it marks the corresponding port as `closed' in the port-list. If it receives a UDP response packet, it sets the corresponding port as `open' in the port-list.

To maintain accuracy and to avoid producing False-positives by processing packets sent by UDP client, it validates received packet based on destination IP address, source port and destination port. If all the fields of received packet are as response expected based on the probe packet, only then the packet is processed, otherwise it is discarded.

At the end of scanning, all the remaining ports which are not marked as open or closed are set to `filtered'.

A schematic view of our solution is shown in Fig. 2.

V. Implementation

We created a working model of our technique. The IP assigned to the scanner is 192.168.1.10. The IP address of SUT is 192.168.1.1. They are connected via an Ethernet cable. There is no other IP in the network. The rate limiter for UDP packets assigned on the scanner is 200 packets per second. The size of subnet is /24.

In this subnet, there are 253 IP addresses which can be used for scanning. When a UDP probe is sent to the SUT, if the ARP cache for the source IP address is not in SUT, it will broadcast an ARP Request message. The scanner will respond to it and reply back with ARP Response via the ARP Poisoner module.

Next, if the port is closed, the SUT will reply back with "ICMP Port unreachable" message. This message is parsed by Sniffer module and corresponding port is marked as ‘closed’. If the port is open and SUT replies back, the port is marked as ‘open’.

VI. Performance Evaluation

For performance evaluation, the proposed scanner was tested against three servers running Linux flavors [12], [13], [14] and one server running Windows [15]. The proposed scanner was coded [16] in Python 2.7 [17] and executed from Kali Linux [18]. Firewall was disabled from the servers. The servers were again tested using nmap from same scanner machine. The tests were conducted three times on each server. Table I shows the time taken by standard nmap and our solution.

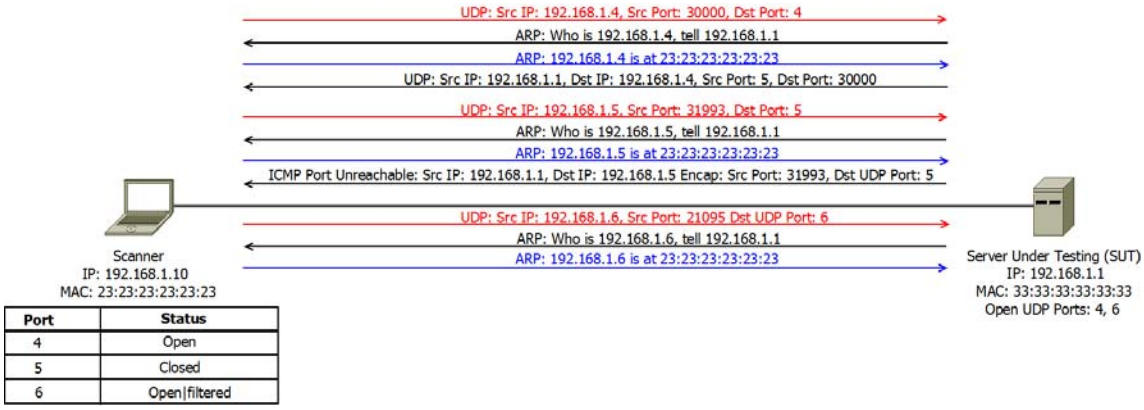

Fig. 2. Scanner in action.

TABLE I: Results of Scanning

Operating System	Time taken (seconds)		Efficiency Boost (%)
	Nmap	Proposed scanner	
CentOS 6.0	65665	344	19088
Debian 6.0	65691	339	19377
Ubuntu 12.10	65675	346	18981
Windows 7	1030	342	301

In Fig. 3, we can see that the nmap scan starts with a high packet burst, but within 300 seconds, the rate drops down to 1 packet per second. The graph captures nmap scanning Ubuntu 12.10 server. The graph has been created using wireshark [19]. The black line denotes UDP packets sent by scanner and red line denotes ICMP packets sent by SUT.

In Fig. 4, the black line represents UDP probe packets sent by the scanner to the SUT. The red dots represent the ICMP unreachable messages received back by the scanner. As we can observe that the rate of scanning remained 200 ports/sec.

The 301% efficiency boost in case of Windows 7 is due to the asynchronous mechanism of port scanning which allows it to send next packet without waiting for replies of previous packet.

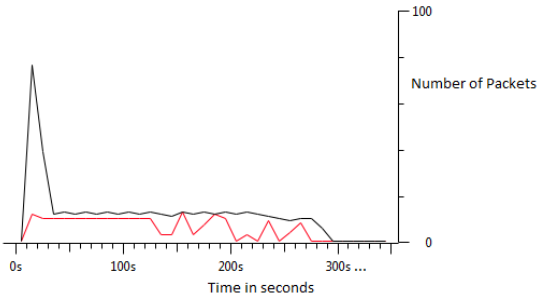

Fig. 3. Graph showing UDP packets during nmap scan.

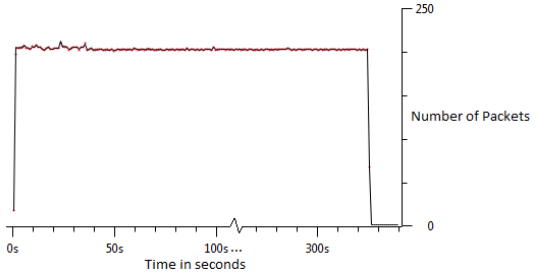

Fig. 4. Graph showing UDP packets during proposed scanning.

VII. Conclusion

The proposed scanner is 191 times faster than traditional scanners against Linux flavors and three times faster against Windows OS. Given that fingerprinting is one of the first steps to find out open ports and services that are running on any system/device, speeding up port scan directly improves performance. Our technique requires special configuration in which it must either be directly connected to the SUT or a network in which there is no NAT/PAT. This is a great advantage, since the unique configuration requirements are unlikely to be available to any hacker/attacker at large, but only to trusted internal users.

Our technique is particularly useful in conditions where there is a need to quickly perform a port scanning under testing environment where ARP poisoning and multiple IPs do not cause interference with the normal working of SUT.

References

[1] J. Postel *et al*. Rfc 792: Internet Control Message Protocol. InterNet Network Working Group. [Online]. Available: http://www. tools.ietf.org/html/rfc792
[2] Nmap - Free Security Scanner for Network Exploration and Security Audits. [Online]. Available: http:/www. /nmap.org/
[3] M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "Surveying port scans and their detection methodologies," *The Computer Journal*, vol. 54, no. 10, pp. 1565–1581, 2011.
[4] H. Liang and Z. Qiansheng, "The design of port scanning tool based on tcp and udp," in *Proc. the 2012 International Conference of Modern Computer Science and Applications*. Springer, 2013, pp. 179–183.
[5] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure*, 2009.
[6] C. Nachreiner. Anatomy of an Arp Poisoning attack. [Online]. Available: http://www.watchguard.com/infocenter/editorial/135324.asp
[7] S. Y. Nam, D. Kim, and J. Kim, "Enhanced arp: preventing arp poisoning-based man-in-the-middle attacks," *Communications Letters*, IEEE, vol. 14, no. 2, pp. 187–189, 2010.
[8] S. Kumar and S. Tapaswi, "A centralized detection and prevention technique against arp poisoning," in *Proc*. the *2012 International Conference on. Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*, 2012, pp. 259–264.
[9] M. De Vivo, E. Carrasco, G. Isern, and G. O. D. Vivo, "A review of port scanning techniques," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 2, pp. 41–48, 1999.
[10] Nessus Vulnerability Scanner. [Online]. Available: http://www.tenable.com/products/nessus
[11] D. Kaminsky, "Explorations in namespace: white-hat hacking across the domain name system," *Communications of the ACM*, vol. 49, no. 6, pp. 62–69, 2006.
[12] Centos Project. [Online]. Available: http://www.centos.org
[13] Debian–the Universal Operating System. [Online]. Available: http://www.debian.org
[14] Ubuntu: The world's Most Popular Free Os. [Online]. Available: http://www.ubuntu.com
[15] Microsoft Windows–Microsoft Windows. [Online]. Available: http://www. windows.microsoft.com
[16] Source Code: [Online]. Available: https://www. github.com/sumit-1/screamer_scanner
[17] Welcome to Python.Org. [Online]. Available: https://www.python.org
[18] Kali Linux - Rebirth of Backtrack, the Penetration Testing Distribution. [Online]. Available: http://www.kali.org.
[19] Wireshark Go Deep. [Online]. Available: http://www.wireshark.org

Sumit Kumar was born in 1987. He received his master's degree in computer science with information security as specialization from IIIT Gwalior India in 2012 and He received his bachelor's degree in computer science and engineering from Apeejay College of Engineering, Gurgaon, India in 2010.

He is working in Device Security Assurance Center, ABB Global Industries and Services Limited, Bangalore, India as Cyber Security Analyst. He has previously worked in Directi Internet Solutions Pvt. Ltd as a system administrator. His research interests include network security, information security and python. He is an offensive security certified professional and has two papers to his credit.

Sithu D. Sudarsan is a member of IACSIT and he was born in 1969. He received his bachelor's in electronics and communication engineering from Madurai Kamaraj University, India in 1990. He received his master's in systems and information from Birla Institute of Technology and Science (BITS), Pilani, India in 1993 and He received his doctoral degree in applied science–applied computing from University of Arkansas at Little Rock (UALR), USA in 2009.

He has over two decades of research and development experience. He spent about 5 years each with the Centre for Electronics Design and Technology in India, Eletronics Research and Development Center of India (India), Bharat Electronics Limited (BEL India) and US Food and Drug Administration (USA) respectively as a design engineer, project manager, member (senior research staff) and visiting scientist. He also spent a year each at St. Michael's Polytechnic as a associate lecturer (1990-1991) and UALR as a graduate research assistant (2007-2008). He is currently leading the software research group at the India Corporate Research Center, Bangalore as part of ABB Corporate Research. He has over 70 publications to his credit and is invited regularly to various conferences as a speaker. His research areas include cyber physical systems, security, big data, real-time systems, computer and communication networks, intellectual property rights.

Dr. Sudarsan has received several awards including chief scientist's citation (BEL 2002), Outstanding Ph. D. Graduate Award (UALR 2010), and Outstanding Service Award (USFDA 2012). He received research grants to develop information security protocol suite for embedded systems and devices including wireless devices. He has participated actively in several standards committees at international/national levels.