INDUSTRY PROBLEM-1

PORT SCANS AND NETWORK ATTACKS USING SCAPY

What is a Port Scan?

A port scan is a common technique hackers use to discover open doors or weak points in a network. A port scan attack helps cyber criminals find open ports and figure out whether they are receiving or sending data. It can also reveal whether active security devices like firewalls are being used by an organization.
When hackers send a message to a port, the response they receive determines whether the port is being used and if there are any potential weaknesses that could be exploited.
Businesses can also use the port scanning technique to send packets to specific ports and analyse responses for any potential vulnerability. They can then use tools like IP scanning, network mapper (Nmap), and Netcat to ensure their network and systems are secure.

Port scanning can provide information such as:

1. Services that are running
2. Users who own services
3. Whether anonymous logins are allowed
4. Which network services require authentication

How to Prevent Port Scan Attacks?

Port scanning is a popular method cyber criminals use to search for vulnerable servers. They often use it to discover organizations' security levels, determine whether businesses have effective firewalls, and detect vulnerable networks or servers. Some TCP methods also enable attackers to hide their location.

Cyber criminals search through networks to assess how ports react, which enables them to understand the business's security levels and the systems they deploy.
Preventing a port scan attack is reliant on having effective, updated that is in line with the evolving threat landscape. Businesses also require strong security software, port scanning tools, and security alerts that monitor ports and prevent malicious actors from reaching their network. Useful tools include IP scanning, Nmap, and Netcat.

Other defence mechanisms include:

1.A strong firewall: A firewall can prevent unauthorized access to a business's private network. It controls ports and their visibility, as well as detects when a port scan is in progress before shutting it down.

2.TCP wrappers: These enable administrators to have the flexibility to permit or deny access to servers based on IP addresses and domain names.

3.Uncover network holes: Businesses can use a port scanner to determine whether more ports are open than required. They need to regularly check their systems to report potential weak points or vulnerabilities that could be exploited by an attacker.

1) **TCP XMAS scan**

Description:
An adversary uses a TCP XMAS scan to determine if ports are closed on the target machine. This scan type is accomplished by sending TCP segments with all possible flags set in the packet header, generating packets that are illegal based on RFC 793. The RFC 793 expected behaviour is that any TCP segment with an out-of-state Flag sent to an open port is discarded, whereas segments with out-of-state flags sent to closed ports should be handled with a RST in response. This behaviour should allow an attacker to scan for closed ports by sending certain types of rule-breaking packets (out of sync or disallowed by the TCB) and detect closed ports via RST packets.

Extended Description:
In addition to its relative speed when compared with other types of scans, its major advantage is its ability to scan through stateless firewall or ACL filters. Such filters are configured to block access to ports usually by preventing SYN packets, thus stopping any attempt to 'build' a connection. XMAS packets, like out-of-state FIN or ACK packets, tend to pass through such devices undetected. Because open ports are inferred via no responses being generated, one cannot distinguish an open port from a filtered port without further analysis. For instance, XMAS scanning a system protected by a stateful firewall may indicate all ports being open. Because of their obvious rule-breaking nature, XMAS scans are flagged by almost all intrusion prevention or intrusion detection systems.

Typical Severity:
Severity is Low.

Domain of Attack:
- Software
- Communication

Mechanism of Attack:
- Collect and Analyse information.

Resources Required:
This attack can be carried out with a network mapper or scanner, or via raw socket programming in a scripting language. Packet injection tools are also useful for this purpose. Depending upon the method used it may be necessary to sniff the network in order to see the response.

Execution Flow:
1. An adversary sends TCP packets with all flags set but not associated with an existing connection to target ports.
2. An adversary uses the response from the target to determine the port's state. If no response is received the port is open. If a RST packet is received then the port is closed.

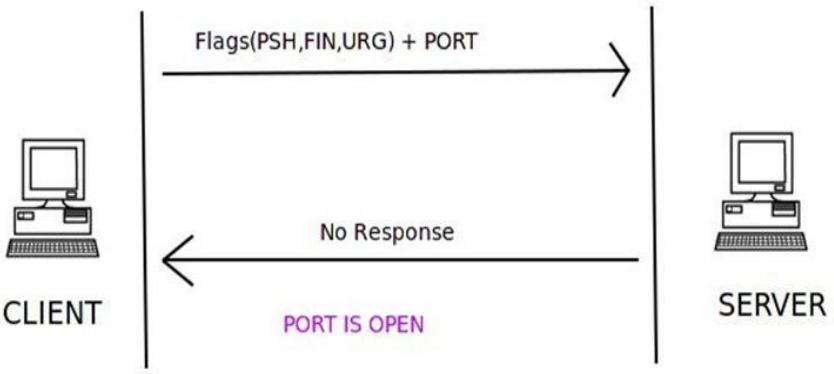
Weakness:

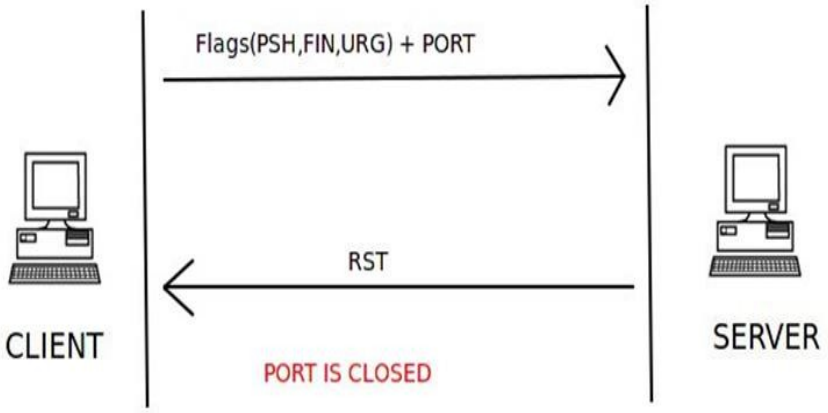
Exposure of Sensitive Information to an Unauthorized Actor

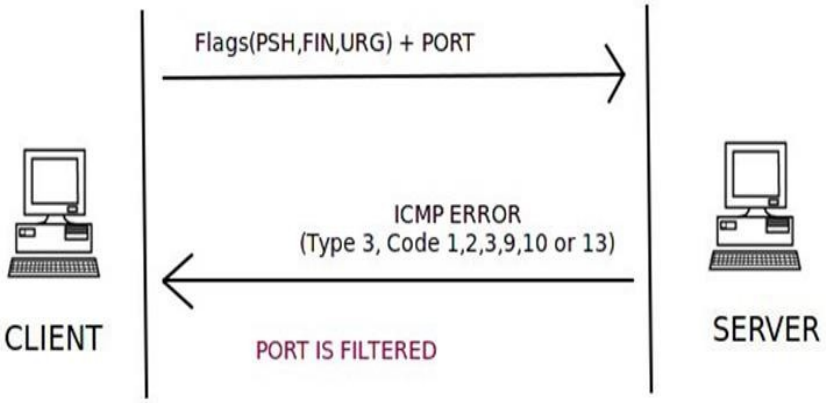
Consequences:

Scope	**Impact**
Confidentiality	Other
Confidentiality access control authorization	Bypass protection Mechanism Hide Activities
Availability	Unreliable Execution

Structure:


Flags(PSH,FIN,URG) + PORT
No Response
CLIENT
PORT IS OPEN
SERVER


Flags(PSH,FIN,URG) + PORT
RST
CLIENT
PORT IS CLOSED
SERVER


Flags(PSH,FIN,URG) + PORT
ICMP ERROR
(Type 3, Code 1,2,3,9,10 or 13)
CLIENT
PORT IS FILTERED
SERVER

Script:

xmas.py

```python
from scapy.all import *

host = 'http://scanme.nmap.org'
ip = socket.gethostbyname(host)

openp = []
filterdp = []
common_ports = { 79,80,81,82
                }
def is_up(ip):
    icmp = IP(dst=ip)/ICMP()
    resp = sr1(icmp, timeout=10)
    if resp == None:
        return False
    else:
        return True

def probe_port(ip, port, result = 1):
    src_port = RandShort()
    try:
        p = IP(dst=ip)/TCP(sport=src_port, dport=port, flags='FPU')
        resp = sr1(p, timeout=2) # Sending packet
        if str(type(resp)) == "<type 'NoneType'>":
            result = 1
        elif resp.haslayer(TCP):
            if resp.getlayer(TCP).flags == 0x14:
                result = 0
            elif (int(resp.getlayer(ICMP).type)==3 and int(resp.getlayer(ICMP).code) in [1,2,3,9,10,13]):
                result = 2

    except Exception as e:
        pass

    return result


if __name__ == '__main__':
    conf.verb = 0
    if is_up(ip):
        for port in common_ports:
            print (port)
            response = probe_port(ip, port)
            if response == 1:
                openp.append(port)
            elif response == 2:
                filterdp.append(port)

        if len(openp) != 0:
            print ("Possible Open or Filtered Ports:")
            print (openp)
        if len(filterdp) != 0:
            print ("Possible Filtered Ports:")
            print (filterdp)
        if (len(openp) == 0) and (len(filterdp) == 0):
            print ("Sorry, No open ports found.!!")
    else:
        print ("Host is Down")
```

Output:

```
PORT — -zsh — 80×24
Last login: Mon May  2 16:19:45 on ttys000
adityasundarraj@Adityas-MBP PORT % Python3 xmas.py
WARNING: No IPv4 address found on anpi0 !
WARNING: No IPv4 address found on anpi2 !
WARNING: more No IPv4 address found on anpi1 !
80
81
82
79
Possible Open or Filtered Ports:
[80, 81, 82, 79]
adityasundarraj@Adityas-MBP PORT %
```

2) **TCP FIN scan**

Description:
An adversary uses a TCP FIN scan to determine if ports are closed on the target machine. This scan type is accomplished by sending TCP segments with the FIN bit set in the packet header. The RFC 793 expected behaviour is that any TCP segment with an out-of-state Flag sent to an open port is discarded, whereas segments with out-of-state flags sent to closed ports should be handled with a RST in response. This behaviour should allow the adversary to scan for closed ports by sending certain types of rule-breaking packets (out of sync or disallowed by the TCB) and detect closed ports via RST packets.

Extended Description:
In addition to its relative speed in comparison with other types of scans, the major advantage a TCP FIN Scan is its ability to scan through stateless firewall or ACL filters. Such filters are configured to block access to ports usually by preventing SYN packets, thus stopping any attempt to 'build' a connection. FIN packets, like out-of-state ACK packets, tend to pass through such devices undetected. FIN scanning is still relatively stealthy as the packets tend to blend in with the background noise on a network link.

Typical Severity:
Severity is Low.

Domain of Attack:
- Software
- Communication

Mechanism of Attack:
- Collect and Analyse information.

Resources Required:
This attack pattern requires the ability to send TCP FIN segments to a host during network reconnaissance. This can be achieved via the use of a network mapper or scanner, or via raw socket programming in a scripting language. Packet injection tools are also useful for this purpose. Depending upon the method used it may be necessary to sniff the network in order to see the response.

Execution Flow:
1. An adversary sends TCP packets with the FIN flag but not associated with an existing connection to target ports.
2. An adversary uses the response from the target to determine the port's state. If no response is received the port is open. If a RST packet is received then the port is closed.

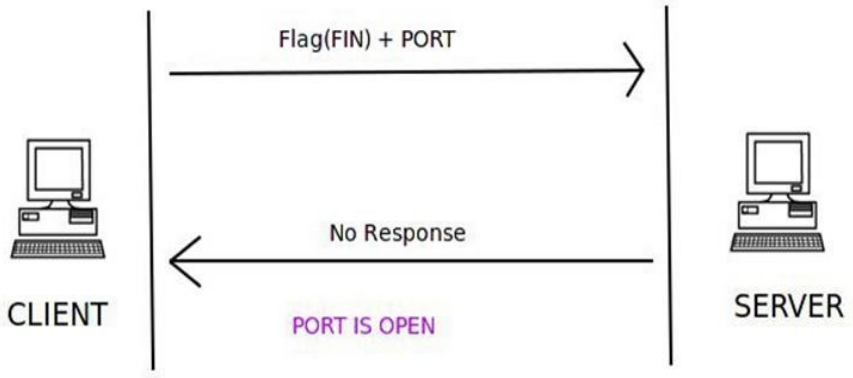
Weakness:

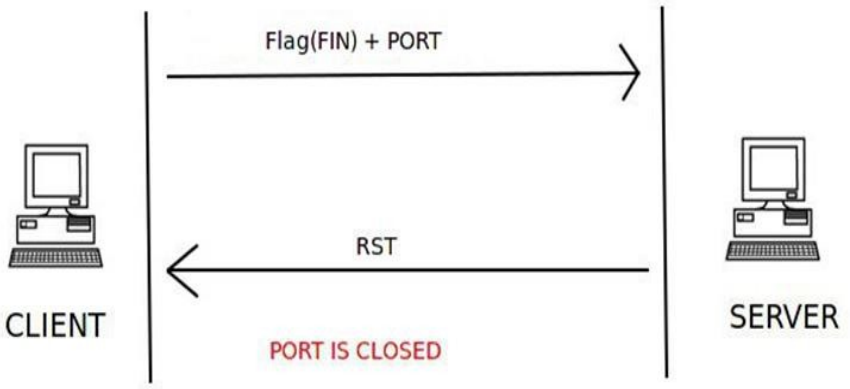
Exposure of Sensitive Information to an Unauthorized Actor

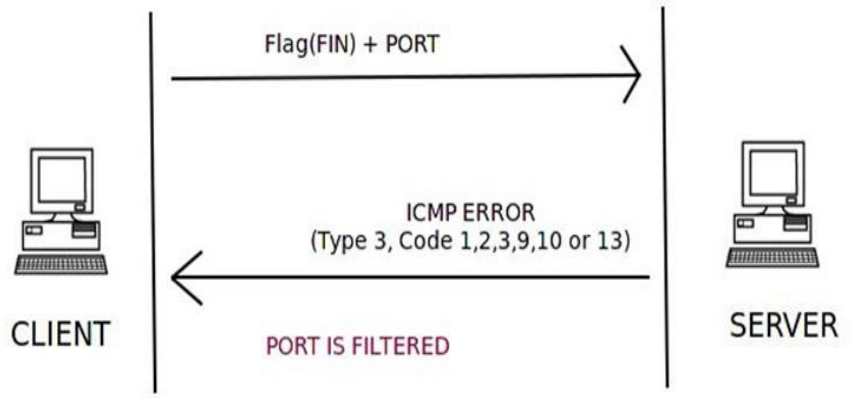
Consequences:

Scope	**Impact**
Confidentiality	Other
Confidentiality access control authorization	Bypass protection Mechanism Hide Activities

Structure:


Flag(FIN) + PORT
No Response
CLIENT
PORT IS OPEN
SERVER


Flag(FIN) + PORT
RST
CLIENT
PORT IS CLOSED
SERVER


Flag(FIN) + PORT
ICMP ERROR
(Type 3, Code 1,2,3,9,10 or 13)
CLIENT
PORT IS FILTERED
SERVER

Script:

fin.py

~/Desktop/PYTHON/PORT/fin.py

```
from scapy.all import *

host = 'http://scanme.nmap.org'
ip = socket.gethostbyname(host)

openp = []
filterdp = []
common_ports = { 79,80,81,82
                 }
def is_up(ip):
    icmp = IP(dst=ip)/ICMP()
    resp = sr1(icmp, timeout=10)
    if resp == None:
        return False
    else:
        return True

def probe_port(ip, port, result = 1):
    src_port = RandShort()
    try:
        p = IP(dst=ip)/TCP(sport=src_port, dport=port, flags='F')
        resp = sr1(p, timeout=2) # Sending packet
        if str(type(resp)) == "<type 'NoneType'>":
            result = 1
        elif resp.haslayer(TCP):
            if resp.getlayer(TCP).flags == 0x14:
                result = 0
            elif (int(resp.getlayer(ICMP).type)==3 and int(resp.getlayer(ICMP).code) in [1,2,3,9,10,13]):
                result = 2

    except Exception as e:
        pass

    return result


if __name__ == '__main__':
    conf.verb = 0
    if is_up(ip):
        for port in common_ports:
            print (port)
            response = probe_port(ip, port)
            if response == 1:
                openp.append(port)
            elif response == 2:
                filterdp.append(port)

        if len(openp) != 0:
            print ("Possible Open or Filtered Ports:")
            print (openp)
        if len(filterdp) != 0:
            print ("Possible Filtered Ports:")
            print (filterdp)
        if (len(openp) == 0) and (len(filterdp) == 0):
            print ("Sorry, No open ports found.!!")
    else:
        print ("Host is Down")
```

L: 8 C: 29 Python Unicode (UTF-8) Unix (LF) Saved: 24/04/22, 4:44:36 PM 1,548 / 178 / 59 100%

Output:

```
[adityasundarraj@Adityas-MBP PORT % Python3 fin.py
WARNING: No IPv4 address found on anpi0 !
WARNING: No IPv4 address found on anpi2 !
WARNING: more No IPv4 address found on anpi1 !
80
81
82
79
Possible Open or Filtered Ports:
[80, 81, 82, 79]
adityasundarraj@Adityas-MBP PORT %
```

3) **TCP ACK scan**

Description:
An adversary uses TCP ACK segments to gather information about firewall or ACL configuration. The purpose of this type of scan is to discover information about filter configurations rather than port state. This type of scanning is rarely useful alone, but when combined with SYN scanning, gives a more complete picture of the type of firewall rules that are present.

Extended Description:
When a TCP ACK segment is sent to a closed port, or sent out-of-sync to a listening port, the RFC 793 expected behaviour is for the device to respond with a RST. Getting RSTs back in response to a ACK scan gives the attacker useful information that can be used to infer the type of firewall present. Stateful firewalls will discard out-of-sync ACK packets, leading to no response. When this occurs the port is marked as filtered. When RSTs are received in response, the ports are marked as unfiltered, as the ACK packets solicited the expected behaviour from a port. When combined with SYN techniques an attacker can gain a more complete picture of which types of packets get through to a host and thereby map out its firewall rule-set. ACK scanning, when combined with SYN scanning, also allows the adversary to analyse whether a firewall is stateful or non-stateful (described in notes). TCP ACK Scans are somewhat faster and more stealthy than other types of scans but often requires rather sophisticated analysis by an experienced person. A skilled adversary may use this method to map out firewall rules, but the results of ACK scanning will be less useful to a novice.

Typical Severity:
Severity is Low.

Domain of Attack:
- Software
- Communication

Mechanism of Attack:
- Collect and Analyse information.

Resources Required:
This attack can be achieved via the use of a network mapper or scanner, or via raw socket programming in a scripting language. Packet injection tools are also useful for this purpose. Depending upon the method used it may be necessary to sniff the network in order to see the response.

Execution Flow:
1. An adversary sends TCP packets with the ACK flag set and that are not associated with an existing connection to target ports.
2. An adversary uses the response from the target to determine the port's state. If a RST packet is received the target port is either closed or the ACK was sent out-of-sync. If no response is received, the target is likely using a stateful firewall.

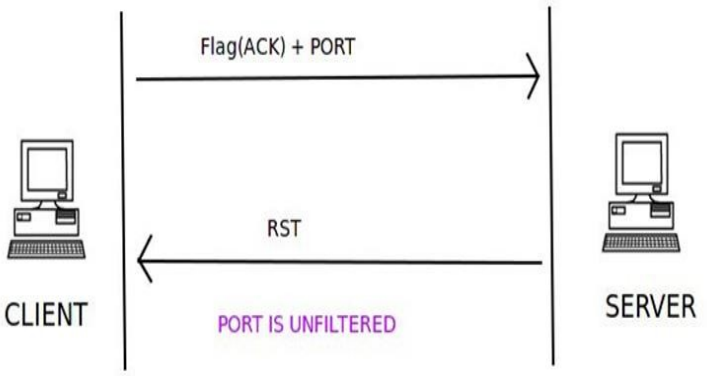
Weakness:

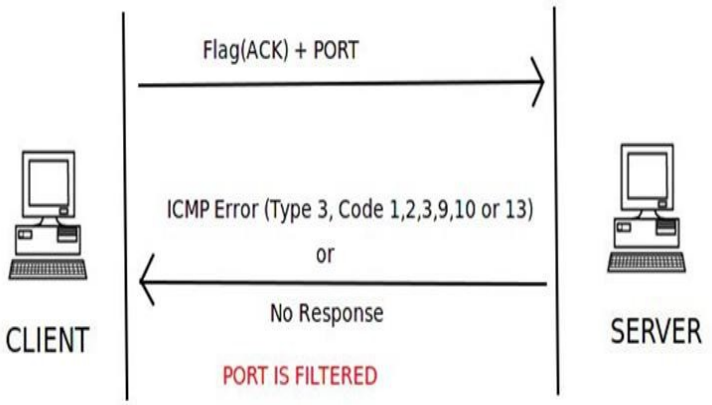
Exposure of Sensitive Information to an Unauthorized Actor

Consequences:

Scope	**Impact**
Confidentiality	Other
Confidentiality access control authorization	Bypass protection Mechanism Hide Activities

Structure:


Flag(ACK) + PORT
RST
CLIENT
PORT IS UNFILTERED
SERVER


Flag(ACK) + PORT
ICMP Error (Type 3, Code 1,2,3,9,10 or 13)
or
No Response
CLIENT
PORT IS FILTERED
SERVER

Script:

ack.py

~/Desktop/PYTHON/PORT/ack.py

```python
from scapy.all import *

host = 'http://scanme.nmap.org'
ip = socket.gethostbyname(host)
port = 80

def is_up(ip):
    icmp = IP(dst=ip)/ICMP()
    resp = sr1(icmp, timeout=10)
    if resp == None:
        return False
    else:
        return True

def probe_port(ip, port, result = 1):
    src_port = RandShort()
    try:
        p = IP(dst=ip)/TCP(sport=src_port, dport=port, flags='A', seq=12345)
        resp = sr1(p, timeout=2) # Sending packet
        if str(type(resp)) == "<type 'NoneType'>":
            result = 1
        elif resp.haslayer(TCP):
            if resp.getlayer(TCP).flags == 0x4:
                result = 0
            elif (int(resp.getlayer(ICMP).type)==3 and int(resp.getlayer(ICMP).code) in [1,2,3,9,10,13]):
                result = 1

    except Exception as e:
        pass

    return result


if __name__ == '__main__':
    conf.verb = 0
    if is_up(ip):
            response = probe_port(ip, port)
            if response == 1:
                 print ("Filtered | Stateful firewall present")
            elif response == 0:
                 print ("Unfiltered | Stateful firewall absent")
    else:
        print ("Host is Down")
```

Output:

```
[adityasundarraj@Adityas-MBP PORT % Python3 ack.py
WARNING: No IPv4 address found on anpi0 !
WARNING: No IPv4 address found on anpi2 !
WARNING: more No IPv4 address found on anpi1 !
Filtered | Stateful firewall present
adityasundarraj@Adityas-MBP PORT %
```