Event-driven computing, port scanning and nmap tool

Antonio Paolacci

May 30, 2011

1 Event-driven computing

Event-driven architecture are implemented to detect and react to an event occurs during the execution of a software application. It's possibile to use this architecture when there is the need to disseminate the right information to the right receiver in the right granularity and at right time, or to do active diagnostics observing subsequent events, or to identify malicious behaviours before they are really happened. An event means a programming entity that represents such an occurrence. There are different types of events: *normal event* that is normally managed, *anticipated abnormal event* that anticipate some abnormal event may occur: for example a system failure and *unanticipated event* that is an unexpexted event. There are many examples of software application based on event-driven computing, for example: financial fraud detection, RFID technologies, emergency control system, intrusion detection system(IDS). Such kind of application is composed by part of *Event-based programming* designing and coding application that makes use of events and a part of *Event processing* performing operations when an event occurs. A simplest type of software could follow the paradigm of *sense-response* application. They are constructed out of 3 principal blocks of code:

- Sense, this block gathers data from within and outside of the application domain.
- Analyse, collected data are analyzed.
- Respond, determines actions that are to be timely taken in response to what has been sensed.

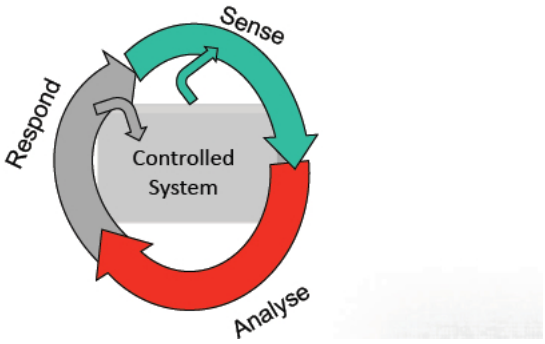

Figure 1: Sense-response loop

An example of sense-response application could be about port scanning(to detect the event open,closed,filtered port for example) and how to detect port scanning activities on our server. In this case **sensing phase** collect traffic of the networks to monitor, **analyzing phase** correlate data of all network domains to find out connections that can be incomplete or failed matching a defined pattern, **responding phase** put IP addresses in a blacklist or block them.// In such way work several port scan detection mechanisms. A port scanning detection system could be a part of IDSs. Many of this type of detectors are easy to evade since they use simple rules that classify a port scan as more than X distinct probes in Y seconds from a single IP source. Typically, the length of Y is severely limited, to keep the amount of state manageable by the system.

2 Port scanning

Port scanning is an activity aimed at discovering the status of TCP/UDP ports of a server or host. Port Scanning performed by an administrator or an attacker gives information about which ports listen on a machine. Every open port is potentially vulnerable and could be used as a vector for a future attack,expecially buffer overflow attack. Also more advanced port scanning techniques try to figure out what kind of software (OS vendor and version) is installed. Port scanning is made with specific software application: **port scanner** designed to probe server or host, called **target** of a specific internet domains for open ports. People who use port scanner are called **scanners** and want to remain hidden during their port scan activities. The base of port scanning is to send internet requests to ports. Internet is based on the Internet Protocol Suite, commonly also called TCP/IP. In this system, hosts and host services are referenced using two components: an address and a port

number. There are 65536 distinct and usable port numbers. Most common services use a limited and specific range of numbers port. There is no "easy way" to detect and stop someone from port scanning an host while it is on the Internet because just accessing an Internet server opens a door to reach that host. Depending on what is the goal, there are three different types of port scan:

- Horizontal scan: a single port over a range of hosts.
- Vertical scan: a range of ports on a single host.
- Block scan: hybrid solution based on a range of ports on a range of hosts.

Depending on the result of a scan usually ports are generalized into one of three categories:

- Open or accepted: the host sent a reply indicating that a service is listening on the port.
- Closed or denied or not listening: the host sent a reply indicating that connections will be denied to the port.
- Filtered, dropped or blocked: there was no reply from the host; a firewall rule prevented packet from reaching target port(filtering it) or target host is unreachable.

As mentioned before open ports present two vulnerabilities: security and stability associated with the program responsible for the service, security and stability associated with the operating system that is running on the host.

2.1 From TCP to SYN port scanning

The TCP port scanning is simplest port scanners and uses the operating system's network functions. If a port is open, the operating system completes the TCP three-way handshake, and then the port scanner immediately closes the connection. Services can log the sender IP address and an IDS(Intrusion detection system) can raise an alarm. But port scan softwares are aimed at keeping the scanner hidden(**stealthy port scans**),so this is no widely used. SYN scan is another form of TCP scanning. Rather than use the operating system's network functions, the port scanner generates raw IP packets itself, and monitors for responses. This scan type is also known as

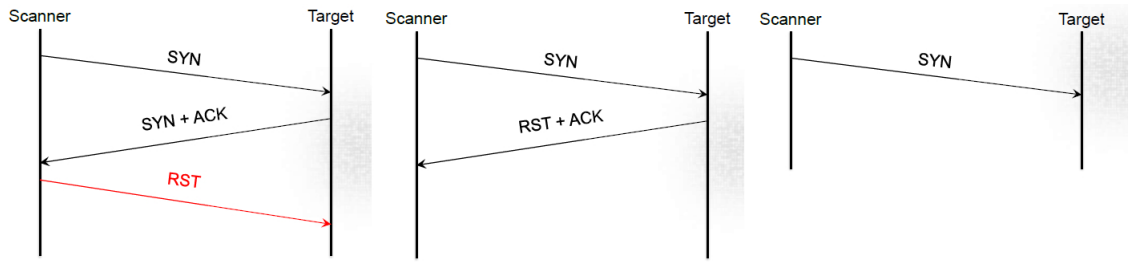

Figure 2: SYN port scanning situations respectively to open,closed,filtered ports

"half-open scanning", because it never actually opens a full TCP connection. The port scanner generates a SYN packet. If the target port is open, it will respond with a SYN+ACK packet. The scanner host responds with a RST packet, closing the connection before the handshake is completed. No trace of any connection at application level at target host. If target will respond with RST+ACK packet, the scanner knows the port is closed, if no response packet is received, the port is filtered. The simplest type of a TCP/SYN scan is made with a call to connect(). The manpage for this system call on Unix/Linux systems has the following prototype for this function:

```
#include <sys/socket.h>
int connect(int socketfd, const struct sockaddr *address,
socklen_t address_len);
```

where the parameter *socketfd* is the file descriptor associated with the internet socket constructed by the client (with a call to three-argument socket()), the pointer parameter *address* that points to a sockaddr structure that contains the IP address of the remote server, and the parameter *address_len* that specifies the length of the structure pointed to by the second argument. A call to connect() if successful completes a three-way hand- shake (that was described in Lecture 16) for a TCP connection with a server. The header file *sys/socket.h* include a number of definitions of structs needed for socket programming in C. When connect() is successful, it returns the integer 0, otherwise it returns -1.

2.2 Dynamic port scanning

Dynamic port scanning is a new technology that aims to dynamically spoof the IP address of the scanner machine. The spoofed IP address of TCP/UDP packets is "dynamically" generated random and at run time by the application. However, that IP address must fall in the local IP subnet of the scanner. Traditionally this type of port scanning was considered unreliable due to the

fact that reply packets would not reach back the scanning machine. A recent implementation that ensures reliability is based on ARP Poisoning integrated with port scanning tools [1].

2.3 Distributed port scanning

Distributed port scann works by dividing the range of IP/ports to probe over multiple attacker machines. In this case each single machine starts to scan a small subset of ports and then sends back the results to the administrator with an increased application overhead. Advantages of distributed scan could be:

- minimizing the time since multiple scanners are working in parallel, but there is overhead to manage the results by the attacker;
- the port scan activity is "quasi-stealthy", tracing back the attacker is a little hard since there are many IPs in the server log file appearing in the logs of the scanned network;

3 The nmap port scanner

Nmap is the best known and documented tool to do port scanning activities. Was presented and developed by Fyodor of *"Insecure.org"*. Now is supported by a big community of open-source programmers. Nmap allows to do many types of port scanning: TCP port scan with *connect()*,TCP SYN port scan (the stealty port scan, already discussed), UDP port scan(note that in UDP we don't have SYN packets, so there is no such thing as a UDP SYN scan), ICMP port scan... Various types of scans can be carried out with a large number of options passed to the command:

```
nmap [Scan Type(s)] [Options] {target specification}
```

- **-sP**: this option, also known as the "ping scanning" option, is for ascertaining as to which machines are up in a network
- **-sV**: this is also referred to as "Version Detection", tries to detect what service is actually running at each of those ports
- **-sT**: a TCP scan with connect() system call.

[1] For more information about visit the website of the project: http://www.securebits.org/projects.html

- **-sU**: a UDP scan
- **nmap -p 1-1024 -sT antoniopaolacci.it**: in this way we can limit the range of examinated ports.
- **nmap -sS -A -P0 antoniopaolacci.it**: by default nmap tries before to detect if the machine is up and then examines the host, so with -P0 option nmap to not use ping in order to decide whether a machine is up.

References

[1] Avinash Kak at Purdue University, "Port Scanning, Vulnerability Scanning, Packet Sniffing, and Intrusion Detection"

[2] Giorgia Lodi,Leonardo Aniello, Slides of the first Distributed Systems Seminar

[3] Andrew J. Bennieston, "NMAP - A Stealth Port Scanner" at http://www.nmap-tutorial.com

[4] Cynthia Bailey Lee, Chris Roedel, Elena Silenok, "Detection and Characterization of Port Scan Attacks",2001