Journal of Network and Computer Applications 40 (2014) 307–324

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

# Network attacks: Taxonomy, tools and systems

N. Hoque [a,*], Monowar H. Bhuyan [a], R.C. Baishya [a], D.K. Bhattacharyya [a], J.K. Kalita [b]

[a] Department of Computer Science & Engineering, Tezpur University, Napaam, Tezpur 784028, Assam, India
[b] Department of Computer Science, University of Colorado at Colorado Springs, CO 80933-7150, USA

ARTICLE INFO

*Article history:*
Received 28 January 2013
Received in revised form
11 July 2013
Accepted 5 August 2013
Available online 15 August 2013

*Keywords:*
Network attacks
Tools
Systems
Protocol
DoS

ABSTRACT

To prevent and defend networks from the occurrence of attacks, it is highly essential that we have a broad knowledge of existing tools and systems available in the public domain. Based on the behavior and possible impact or severity of damages, attacks are categorized into a number of distinct classes. In this survey, we provide a taxonomy of attack tools in a consistent way for the benefit of network security researchers. This paper also presents a comprehensive and structured survey of existing tools and systems that can support both attackers and network defenders. We discuss pros and cons of such tools and systems for better understanding of their capabilities. Finally, we include a list of observations and some research challenges that may help new researchers in this field based on our hands-on experience.


**Contents**



* Corresponding author. Tel.: +91 9864967675.
*E-mail addresses:* tonazrul@gmail.com (N. Hoque), mhb@tezu.ernet.in (M.H. Bhuyan), rcb@tezu.ernet.in (R.C. Baishya), dkb@tezu.ernet.in (D.K. Bhattacharyya), jkalita@uccs.edu (J.K. Kalita).

1084-8045/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2013.08.001

## 1. Introduction

Due to the Internet's explosive growth and its all-pervasive nature, users these days rely on computer networks for most day-to-day activities. Network attacks attempt to bypass security mechanisms of a network by exploiting the vulnerabilities of the target network. Network attacks disrupt legitimate network operations and include malfunctioning of network devices, overloading a network and denying services of a network to legitimate users, highly reducing network throughput, scanning maliciously and other similar activities. An attacker may also exploit loopholes, bugs, and misconfigurations in software services to disrupt normal network activities. Network security tools facilitate network attackers as well as network defenders in identification of network vulnerabilities and collection of network statistics. Network attackers intentionally try to identify loopholes based on common services open on a host and gather relevant information for launching a successful attack. Thus it is of considerable interest to attackers to determine whether or not the defenders of a network are monitoring network activities regularly. Network defenders try to reduce abnormal activities from live network traffic. Defenders do not usually hide their identity during observation while attackers do.

A large number of network security tools have been designed to launch, capture, visualize, and detect different types of attacks with multiple objectives. Example tools include LOIC (Pras et al., 2010), HOIC (Mansfield-Devine, 2011), Wireshark (Orebaugh et al., 2006), Gulp (Satten, 2007), Ntop (Deri et al., 2001), etc. These tools can be used for capture of live network traffic, preprocessing, feature extraction, vulnerability analysis, traffic visualization and actual detection of attacks. Thus, network security tools help in network security engineering from the viewpoint of both attackers and defenders.

### 1.1. Motivation

Even though there are several published surveys of network security tools such as Conti and Abdullah (2004), Barber (2001), Pilli et al. (2010), their scopes are limited and they usually discuss only a few tools. In Conti and Abdullah (2004), the authors discuss network attack tools which are specific to visual fingerprinting. In Barber (2001), the authors include a few tools that are commonly used by hackers. An exhaustive survey of network forensics is presented in Pilli et al. (2010). The authors categorize the tools into two major groups, viz., network forensic analysis tools and network security tools. None of the surveys (Conti and Abdullah, 2004; Barber, 2001; Pilli et al., 2010) include a taxonomy, attack launching tools, and information gathering tools. They also do not discuss recent network intrusion detection systems. Hence, in this paper we present a structured and comprehensive survey on network attacks in terms of general overview, taxonomy, tools, and systems with a discussion of challenges and observations. Our paper is detailed with ample comparisons where necessary and intended for readers who wish to begin research in this field.

### 1.2. Prior surveys

Several surveys on network security are available in the literature (Conti and Abdullah, 2004; Barber, 2001; Pilli et al., 2010; Gogoi et al., 2011; Zhou et al., 2010; Lunt, 1993; Peng et al., 2007; Dhanjani and Clarke, 2005; Li et al., 2013). However, only a few surveys cover network security tools in general. Garcia-Teodoro et al. (2009) discuss some popular anomaly based intrusion detection techniques and systems. They focus on NIDSs and several detection techniques under three major categories, viz., statistical, knowledge based, and machine learning. Corona et al. (2013) present an overview of adversarial attacks against intrusion detection systems. They provide a general taxonomy of attacks against IDSs, use of IDS weaknesses for attack implementation, and solutions for each attack they include. An overview of IDSs in terms of detection and operation is given in Axelsson (2000). Debar et al. (1999a) present a taxonomy of IDSs from several security aspects. A few fingerprinting attack tools with their detection methodologies are briefly summarized in Conti and Abdullah (2004). Out of the top 75 network security tool list produced by *fyodor*, the creator of nmap, a few tools have been included. Barber (2001) present a few sophisticated attack tools with their usefulness in brief. Our survey differs from these previous surveys in view of the following points.

(a) Unlike (Sherif and Dearmond, 2002), we present tools and attack detection systems under two main categories viz., tools for network defenders and tools for attackers. Our survey also includes a comparison of tools with parameters that are useful to the network traffic analyzer.
(b) A survey of network forensic analysis methods is reported in Pilli et al. (2010). Similar to this survey, we describe tools for network defenders as well as tools for attackers used during network traffic capture, analysis and visualization. In addition, we also include a discussion on several IDSs, with architectures to improve the reader's understanding of the detection mechanisms.
(c) Unlike Corona et al. (2013), we include a taxonomy of network attacks, tools and detection systems. Tools are categorized into two classes: tools for network defenders and tools for attackers. The tools and systems are evaluated using parameters that may help in choosing a tool or a system for experiment or for specific applications.

A comparison of the existing surveys on network security tools and systems is given in Table 1.

**Table 1**
Comparison with existing surveys.

| References | Tools | | | Systems | | |
|---|---|---|---|---|---|---|
| | Attack | Defense | Both | Host | Network | Hybrid |
| Conti and Abdullah (2004) | √ | | | | | |
| Barber (2001) | √ | | | | | |
| Pilli et al. (2010) | | | √ | √ | √ | √ |
| Zhou et al. (2010) | | | | √ | √ | |
| Lunt (1993) | | | | √ | √ | |
| Peng et al. (2007) | | | | √ | √ | |
| Dhanjani and Clarke (2005) | √ | | | | | |
| Debar et al. (1999a) | | | | √ | √ | |
| Axelsson (2000) | | | | √ | √ | √ |
| Sherif and Dearmond (2002) | | | | √ | √ | √ |
| Lazarevic et al. (2005) | | | | √ | √ | |
| Chandola et al. (2009) | | | | √ | √ | √ |
| Bhuyan et al. (2013) | √ | | | √ | √ | √ |

### 1.3. Contributions

This paper provides a structured and comprehensive survey on network security tools and systems that are needed by network security researchers. The major contributions of this survey are the following.

1. Like the taxonomy of network security tools and systems given in Pilli et al. (2010), we classify the tools and systems into a number of categories. In addition, we also provide an analysis of tools in terms of their capabilities, performance, details of parameters and input/output.
2. Most existing surveys do not fully cover launching and information gathering tools, but we do.
3. In addition to discussing network security tools, we present a few popular NIDSs with architecture diagrams including components and functions, and also present a comparison among the NIDSs.
4. Finally, we list several important observations from both technical and practical viewpoints.

### 1.4. Organization

The rest of the paper is organized as follows. Network attacks and related concepts are discussed in Section 2. Section 3 presents a taxonomy of network security tools, a description of these tools, in addition to comparisons among them, whereas Section 4 is dedicated to attack detection systems and architectures. Finally, we present our observations and conclusions in Section 5.

## 2. Network attacks and related concepts

An attack can take many forms such as a Trojan attack, DoS/DDoS attack, or a scan attack. A DDoS attack is very catastrophic to any information system since it uses a large number of compromised hosts and it is very difficult to detect the original source of such an attack. Protecting against system compromise is a good way to defend against DDoS attacks. We discuss different types of anomalies, steps in launching attacks and their detection mechanisms.

### 2.1. Anomalies in network

Anomalies are non-conforming interesting patterns compared to the well-defined notion of normal behavior. Traffic anomalies in computer networks are categorized as network operation anomaly, flash crowds and network abuse anomaly. All these anomalies can be detected by analyzing the traffic volume transmitted from station to station. In addition to these anomalies, other anomalies such as ALPHA, DoS/DDoS, scan, worm, outage, ingress shift, information gathering, passive attacks, spoofing attacks, man-in-middle attacks, and DNS cache poisoning, can cause damage and destruction to the network environment. Anomalies can have large impacts on both performance and security. For example, network anomalies cause service degradation and impact on network speed, as a result of which, network performance may suffer considerably.

### 2.2. Steps in launching an attack

Generally, an attacker executes four basic steps (Bhuyan et al., 2011a) to launch an attack. The steps are given below:

(i) *Information gathering*: The attacker attempts to gather vulnerability information from the network with the hope that some of the information can be used to aid in the ensuing attack.
(ii) *Assessing vulnerability*: Based on the vulnerabilities learned in the previous step, the attacker attempts to compromise some nodes in the network by exploiting malicious code, as a precursor to the launching of attack(s).
(iii) *Launching attack*: The attacker launches the attack on the target victim machine(s) using the compromised nodes.
(iv) *Cleaning up*: Finally, the attacker attempts to eliminate the attack history by cleaning up all the registry or log files from the victim machine(s).

### 2.3. Launching and detecting attacks

Before launching an attack, an attacker first attempts to gather vulnerability information about the target system that may help in attack generation. An attacker scans the network using information gathering tools like nmap and finds loopholes in the system. Based on the gathered information, the attacker exploits some malicious code, possibly available on the network. The malicious code may be used to first compromise hosts in the network or it may be used to directly launch an attack and disrupt the network. There are many methods for launching an attack. For example, one may use Trojans or worms to generate an attack on a system or a network. Scanning or information gathering may be coordinated with an attack and performed simultaneously. One can also use attack launching tools such as Dsniff (Danielle, 2002), IRPAS (Yeung et al., 2008), Ettercap (Norton, 2004) and Libnet (Schiffman, 2000) to generate MAC attacks, ARP attacks or VLAN attacks. The main purpose of the attacker in many cases is to disrupt services provided by the network either by consuming resources or consuming bandwidth. These types of attacks can be launched using flooding of legitimate requests as in TCP SYN flooding, ICMP flooding and UDP flooding.

To detect an attack, one must know the characteristics of an attack and its behavior in a network. The network administrator needs a visualization or monitoring system to observe differences between the characteristics of abnormal traffic and the normal. An attack can be detected from the traffic volume based on the packet header or network flow information. However, such detection usually requires processing huge volumes of data in near real-time. Obviously, designing a real-time defense mechanism that can identify all attacks is a challenging and quite likely impossible task. Most detection methods need some prior information about attack characteristics to use during the detection process. The evaluation of these intrusion detection mechanisms or systems is performed using misclassification rate or false alarm rate. To obtain satisfactory results, an IDS designer needs to be careful in choosing an approach, matching mechanism or any heuristic or in making assumptions. Approaches that have been able to obtain acceptable results include statistical (Ye et al., 2002), soft computing (Chen et al., 2005), probabilistic (Jemili et al., 2007), knowledge-based (Debar et al., 1999b) and hybrid (Aydın et al., 2009). A detailed discussion of these approaches is available in Bhuyan et al. (2011c, 2011b).

Detection systems are designed to protect the network from different types of vulnerabilities, which may crash the network or may capture private or secure information. Deployment of an accurate and efficient anomaly detection system demands appropriate design as per standard security requirements and risk analysis. The detection system can be either host based or network based. A typical network structure with a protected LAN, a demilitarized zone and a deployed IDS console is shown in Fig. 1. A demilitarized zone (sometimes referred to as a perimeter network) is a physical or logical subnetwork that contains and exposes an organization's external-facing services to a larger untrusted network, usually the Internet. As shown in the figure,

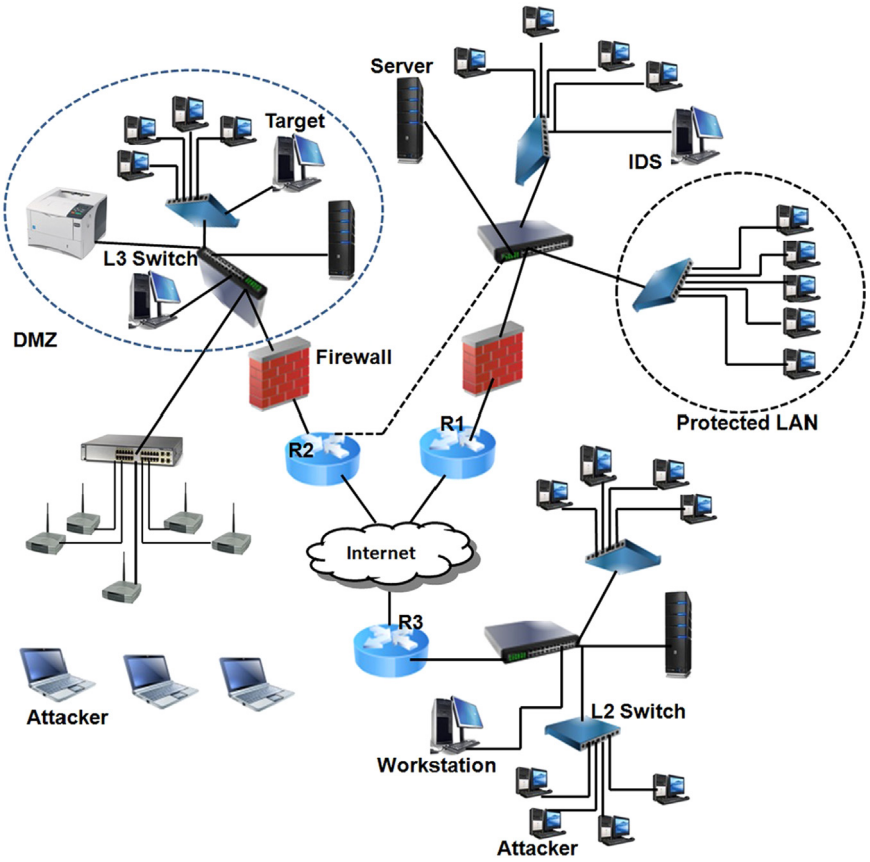


**Fig. 1.** A typical network structure with protected LAN, DMZ and IDS deployment.

an attacker may launch an attack from various machines connected to the network either via wired or wireless media.

The increasing number of highly sophisticated attacks of complex and evolving nature has made the task of defending networks challenging. The appropriate use of tools and systems can simplify the task significantly. This necessitates an awareness of the characteristics and relevance of these tools and systems, and their usage.

## 3. Network security tools

People use different attack tools to disrupt a network for different purposes. As mentioned earlier, attackers generally target Web sites or databases as well as enterprise networks by gathering information based on their weaknesses. In general, attackers use relevant tools for the class of attack they desire to launch. A large number of defense tools also have been made available by various network security research groups as well as private security professionals. These tools have different purposes, capabilities and interfaces.

We categorize existing tools into two major categories: tools for attackers and tools for network defenders. A taxonomy of the tools used in network security is shown in Fig. 2. For each basic category, we show subcategories considering their general characteristics.

### 3.1. Information gathering tools

Before launching an attack, attackers need to understand the environment where the attack is to be launched. To do so, attackers first gather information about the network such as the port numbers of machines and services, operating systems, and so forth. After gathering information, attackers find weaknesses in the network using various tools. Information gathering tools are further classified as sniffing tools and network mapping/scanning tools.

#### 3.1.1. Sniffing tools

A sniffing tool aims to capture, examine, analyze and visualize packets or frames traversing across the network. To support extraction of additional packet features and for their subsequent analysis, it also requires the protocol information during visualization. Some packet sniffing tools are discussed below.

(i) *Tcpdump*: Tcpdump is a premier packet analyzer for information security professionals. It enables one to capture, save and view packet data. This tool works on most flavors of the Unix operating system. One can also use third party open source software, e.g., wireshark to open and visualize tcpdump captured traffic.

(ii) *Ethereal*: Ethereal is a sniffing and traffic analyzing software tool for Windows, Unix and Unix-like OSs, released under the GNU license scheme. It includes two primary library utilities, (a) *GTK+*, a GUI based library, and (b) *libpcap*, a packet capture and filtering library. Ethereal is also capable of reading the output of tcpdump and can apply tcpdump filters to select and display records satisfying certain parameters. Ethereal offers decoding options for a large number ($\geq 400$) of protocols and is useful in network forensics. It supports preliminary inspection of attacks in the network.

(iii) *Net2pcap*: It is a simple tool to read packet traffic from an interface and to transform into a pcap file. Net2pcap is a Linux tool which does not use any library during the transformation. However, it is partially dependent on *libc*, a Linux library utility. The command *%tcpdump -w capfile*

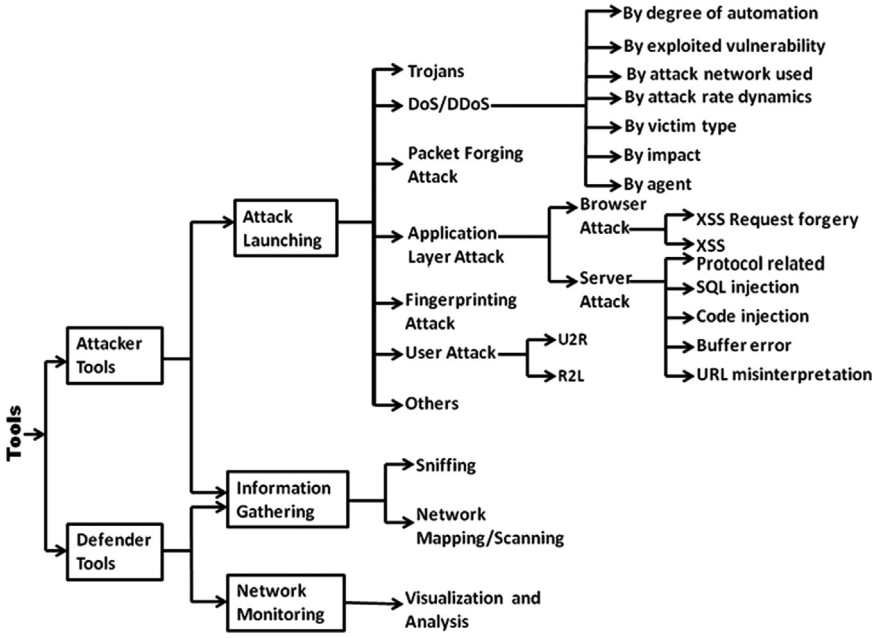


**Fig. 2.** Taxonomy of network security tools.

almost does the same task as Net2pcap. However, Net2pcap is usually used to capture and represent network traffic in a hostile environment to support subsequent analysis.

(iv) *Snoop*: It is a Linux based tool, that works like tcpdump. However, the format of a snoop file is different from the *pcap* format and is defined in RFC 1761.[1] An important feature of this tool is that when writing to an intermediate file, it reduces the possibility of packet loss under busy trace conditions. Snoop allows one to filter, read as well as to interpret packet data. To observe the traffic between two systems, say *X* and *Y*, the following command is used to execute the tool: *% snoop X, Y*

(v) *Angst*: Angst runs on Linux and OpenBSD and is an active packet sniffer that can capture data on switched networks by injecting data into the network. Angst is able to flood a network using random MAC addresses, by causing switches to transmit packets towards all ports.

(vi) *Ngrep*: Ngrep provides filtering facility on packet payloads. It also supports sniffing with the help of tcpdump and libpcap.

(vii) *Ettercap*: Ettercap is a very good sniffer that runs on almost all platforms. More of an active hacking tool, Ettercap uses an ncurses interface and is able to decode several protocols. Ettercap operates in multipurpose mode: sniffer, and interceptor or logger mode for switched LANs. Ettercap can collect passwords for multiple applications, kill connections, inject packets, inject commands into active connections, and has additional plugins.

(viii) *Dsniff*: Dsniff is a collection of tools that enable active sniffing on a network. It can perform man-in-the-middle attacks against SSHv1 and HTTPS sessions. It can also sniff switched networks by actively injecting data into the network and redirecting traffic.

(ix) *Cain and Able*: It is a multipurpose sniffer tool that runs on Windows NT, 2000 and XP and allows for password recovery for a number of protocols, including MSN messenger, and RADIUS shared keys. It can also launch man-in-the-middle attacks for SSHv1 traffic.

(x) *Aimsniff*: It is a simple tool to capture the IP address of an AOL Instant Messenger user while a direct connection is established between the user with others. Once the connection is established, one is able to simply click on the sniff button to capture the IP address.

(xi) *Tcptrace*: It is a powerful tool to analyze tcpdump files and to generate various types of outputs including connection specific information, such as the number of bytes and segments sent and received, elapsed time, retransmissions, round trip times, window advertisements and throughput. It accepts a wide range of input files generated by several capture tools such as tcpdump, snoop, etherpeek, HP Net Metrix, and WinDump. It also provides a graphical presentation of traffic characteristics for further analysis.

(xii) *Tcptrack*: It can sniff and display TCP connection information, as seen in the network interface. Tcptrack can perform the following functions: (a) watch passively for connections on the network interface, (b) keep track of their state and (c) display a list of connections. It displays source IP, destination IP, source port, destination port, connection state, idle time, and bandwidth usage.

(xiii) *Nstreams*: It is a tool to display and analyze network streams generated by users between several networks, and between networks and the outside. Nstreams also can output optionally the *ipchains* or *ipfw rules* matching these streams. It parses outputs generated by tcpdump or files generated using tcpdump with $-w$ option.

(xiv) *Argus*: Argus runs on several operating systems such as Linux, Solaris, Mac OS X, FreeBSD, OpenBSD, NetBSD, AIX, IRIX, Windows and OpenWrt. It can process either live traffic or captured traffic files and can output status reports on flows detected in the stream of packets. It reports reflect flow semantics. This tool provides information on almost all packet parameters, such as reachability, availability, connectivity, duration, rate, load, loss, jitter, retransmission, and delay metrics for all network flows.

(xv) *Karpski*: This is a user-friendly tool with limited sniffing and scanning capabilities. It provides flexibility to include protocol

[1] http://snoopwpf.codeplex.com/

definitions dynamically and also can serve as an attack launching tool against addresses on a local network.

(xvi) *IPgrab*: This packet sniffing tool provides facility for network debugging at multiple layers, such as data link, network and transport layers. It outputs detailed header field information for all layers.

(xvii) *Nast*: Nast uses libnet and libpcap to sniff packets in normal mode or in promiscuous mode to analyze them. It captures packet header parameters, payload information, and saves them in a file in ASCII or ASCII-hex format.

(xviii) *Aldebaran*: It is an advanced libpcap-based sniffing and filtering tool for the TCP protocol. It provides basic information about the source and destination addresses and ports, but no information regarding flags. One can use it to monitor data sent by connections as well as to sniff passwords. Based on libpcap rules, one can use it to sniff packet headers as well as payload contents, and can transmit captured traffic to another host via UDP. Aldebaran also allows one (a) to encrypt the contents saved in dump files, (b) to analyze interface traffic and (c) to report packet statistics, viz., packet count, size, and average speed in HTML or as a plain text file.

(xix) *ScoopLM*: This is a Windows 2000 based sniffing tool for capturing LM/NTLM authentication information on the network. Such information can later be used by a tool such as BeatLM to crack authentication data.

(xx) *Gulp*: It is used to capture very high volume network traffic efficiently from the network firehose. It overcomes the packet loss problem of tcpdump and records a large amount of data that are stored in secondary storage for further processing. It can capture packets from multiple CPUs for better performance and writes the captured data in pcap files.

(xxi) *Nfsen*: Nfsen is used to visualize NetFlow flow data and allows one to display the captured data such as flows, packets and bytes using a graphical interface. One can also visualize protocol specific flows in a graphical format using Nfsen.

(xxii) *Nfdump*: It is used to collect and process NetFlow data. It reads NetFlow data from the files created by nfcapd. It organizes captured data in a time based fashion, typically every 5 min and stores them for further processing. Analysis of data can be performed for a single file, or by concatenating several files in a single run. The output is either ASCII text or binary data, when saved into a file, ready to be processed again with the same tool.

We note that the sniffing tools we have discussed above are not equally useful for all purposes all the time. Their usefulness and importance depend on the user's requirements and purpose at a certain point in time. For example, one cannot use the Cain & Able to capture live network traffic since it performs only password cracking. Most people use tcpdump and libpcap as network sniffing tools to capture all information in packets and store them in a file. One can use the Nfsen and Nfdump tools for NetFlow traffic capture whereas Gulp is used for packet level traffic capture. However, these tools also use tcpdump as an implicit tool for packet as well as NetFlow capture.

### 3.1.2. Scanning tools

A network scanning tool aims to identify active hosts on a network, either (a) to attack them, or (b) to assess vulnerabilities in the network. It provides an overall status report regarding network hosts, ports, IPs, etc. The four possible types of port scans are (i) one-to-one, (ii) one-to-many, (iii) many-to-one, and (iv) many-to-many as shown in Fig. 3. Below, we present a few network vulnerability scanning tools.

(i) *Nmap*: This network mapping tool facilitates network exploration and security auditing. It can scan large networks fast, especially against single hosts. It is effective in using raw IP packets to identify a large number of useful parameters, such as available hosts, services offered by the hosts, OSs running, and use of packet filters or firewalls. In addition to its use in security audits, network administrators can use it for routine tasks such as maintaining network inventory, managing service upgrade schedules, and monitoring host or service uptime.

(ii) *Amap*: Amap detects an application protocol without depending on the TCP/UDP ports it is bound to. It identifies applications running on a specific port by sending trigger packets, which are typically involved in an application protocol handshake. Most network daemons only respond to the correct handshake (e.g., SSL). Amap takes the responses and looks for matches. This tool supports TCP and UDP protocols, regular and SSL-enabled ASCII and binary protocols and has a wide list of options to control its behavior. It accepts an nmap machine readable output file and logs to a file and a terminal.

(iii) *Vmap*: This version mapper tool allows one to identify the version of a daemon by fingerprinting its characteristics, based on its responses to bogus commands.

(iv) *Unicornscan*: This is an asynchronous scanner as well as a payload sender. This scalable and flexible tool gathers and collects information quickly. For fast response, it uses a distributed TCP/IP stack and provides a user-friendly interface to introduce a stimulus into a TCP/IP enabled device or network, and measure the response. The main features of this tool include asynchronous protocol specific UDP scanning, asynchronous stateless TCP scanning with wide variations in TCP flags and asynchronous stateless TCP banner grabbing.

(v) *TTLscan*: This tool uses libnet and libpcap utilities to identify a host by sending TCP SYN packets to each port of the host. It sniffs the response from the host and uses it to identify

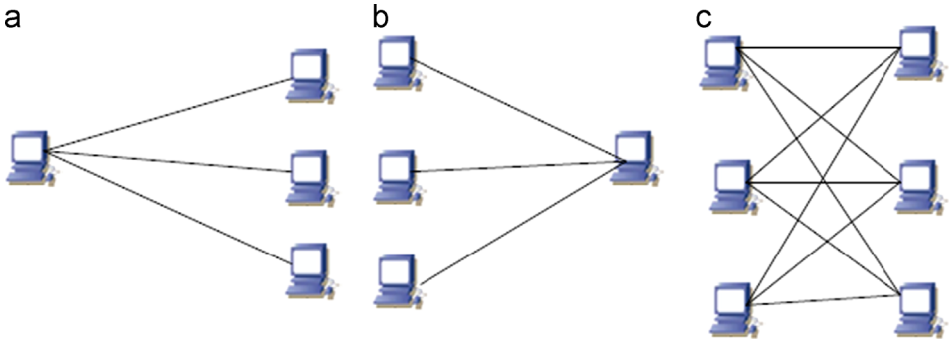


**Fig. 3.** Different types of port scans. (a) Single source port scan (one-to-many), (b) distributed port scan (many-to-one) and (c) distributed port scan (many-to-many).

**Table 2**
Comparison of sniffing tools.

| Tool's name | Input | Protocol | Purpose | Effectiveness | Sources |
|---|---|---|---|---|---|
| Ethereal | I | C/HT/S | Packet capturing | Powerful, user friendly | www.ethereal.com. |
| Tcpdump | I | C/U/IC | Packet capturing | Less intrusive than Ethereal | www.tcpdump.org |
| Net2pcap | I | C/U/IC | Packet capturing | Linux based, auditable | www.secdev.org |
| Snoop | N | C/U/IC/ Te/ F | Packet capturing | No packet loss, supports more than 12 options | www.softpanorama.org |
| Angst | H/p | C | Sniffing | Easy to use, aggressive | www.angst.sourceforge.net |
| Ngrep | I | C/U/IC | Capturing packet | Multi-platform, handles large data | www.ngrep.sourceforge.net |
| Ettercap | I | C/U | Man-in-the-middle attack | Efficient, supports more that 35 options | www.ettercap.sourceforge.net |
| Dsniff | I | F/Te/ S/HT/P | Password sniffing | Unix based | www.naughty.monkey.org |
| Cain & able | I | | Password recovery | Easy to use | www.oxid.it |
| Aimsniff | H | C/U/HT | Capturing packet | Linux based | www.sourceforge.net |
| Tcptrace | F | C | Analysis of traffic | Most commonly used | www.tcptrace.org |
| Tcptrack | I/p | C | TCP connection analysis | Linux based | www.rhythm.cx |
| Argus | F | C/U | Analysis of audit data | Muti-platform, real-time processing | www.qosient.com/argus |
| Karpski | I | C/U | Packet analyzer | Limited applicability | www.softlist.net |
| IPgrab | I/p | | Display packet header | Displays packet details | www.ipgrab.sourceforge.net |
| Nast | I | C/U | Traffic analysis | Supports more than 12 options | www.nast.berlios.de |
| Gulp | I | C/U/IC | Packet capturing/visualization | Very efficient, easy to use | staff.washington.edu/corey |
| Libpcap | I | C/U/IC | Packet capturing | High performance | www.tcpdump.org |
| Nfsen | I | C/U | Flow capturing/ visualization | Easy navigation of NetFlow data | www.nfsen.sourceforge.net |
| Nfdump | I | C/U | Flow capturing/visualization | Powerful packet analyzer | www.nfdump.sourceforge.net |

Here, I – interface ID, N – network IP, H – host IP, F – traffic captured file, p – port, C – TCP, U – UDP, IC – ICMP, IG – IGMP,HT – HTTP, S – SMTP, F – FTP, P – POP, Te – Telnet.

**Table 3**
Comparison of scanning tools.

| Tool's name | Input | Protocol | Purpose | Effectiveness | Sources |
|---|---|---|---|---|---|
| Nmap | IP/p | C/U | Scanning | Very powerful, easy to use | www.insecure.org |
| Amap | IP/p | C/U | Scanning | Powerful application mapper | www.freeworld.thc.org |
| Vmap | T | C/U | Version mapping | Few options, easy to use | www.tools.l0t3k.net |
| Unicornscan | T/p | C/U | Scanning | Supports more than 15 options | www.unicornscan.org |
| Ttlscan | T/p | C | Scanning | Linux based | www.freebsd.org |
| Ike-scan | T | C/U | Host discovery | Supports more than 50 options | www.stearns.org |
| Paketto | IN | C | Scanning | Very fast scanner | www.packages.com |

Here, IP – IP address(es), T – target IP, p – port, IN – interface ID, C – TCP, U – UDP.

hosts with services by forwarding packets to another host behind a firewall. It can detect the OS and its version running on a host behind the firewall by reading specific header parameters such as TTL, window size and IP ID.

(vi) *IKE-scan*: This tool assists in discovery, fingerprinting and testing of IPSec VPN servers based on the IKE protocol. IKE-scan works on Linux, Unix, Mac OS and Windows environment under the GPL license.

(vii) *Paketto*: It is a set of tools to assist in manipulating TCP/IP networks based on non-traditional strategies. These tools provide tapping functionality within the existing infrastructure and also extend protocols beyond their original purpose. Example tools include (a) *scanrand*, which facilitates fast discovery of network services and topologies, (b) *minewt*, which serves as a user space NAT/MAT router, (c) *linkcat*, which offers an Ethernet link to stdio, (d) *paratrace*, which helps trace network paths without spawning new connections, and (e) *phentropy*, which uses *OpenQVIS* to render arbitrary amounts of entropy from data sources in 3-D phase space. (Table 2)

Table 3 shows the purposes and effectiveness of these tools and the sources from where they can be obtained. Almost all these tools are Linux based.

For scanning a large network, one can use nmap as the most effective tool. Nmap has the ability to scan a large network to determine multiple parameters such as active hosts and ports, host operating systems, protocols, timing and performance, firewall/IDS evaluation and spoofing, and IPv6 scanning. Due to its multiple functionalities, network administrators find it very useful to monitor a large network. Amap and Vmap do not support many of the functionalities performed by nmap. Attackers use namp to find the vulnerabilities in a host to compromise it for constructing BotNets during DDoS attack generation using the agent handler architecture.

### 3.2. Attack launching tools

A large number of network security tools that use cryptographic mechanisms to launch attacks are available on the Web. People can freely download these tools and can use them for malicious activities such as Trojan propagation, network mapping, probe attacks, buffer overflow attacks, DoS/DDoS attacks, and application layer attacks. Such tools can be used to launch layer specific and protocol specific attacks, such as HTTP, SMTP, FTP or SNMP related attacks. Other tools can be used to launch DoS/DDoS attacks, which can disrupt the services of a network or a Website very quickly. Some tools are used in wired networks to capture and exploit valuable information while others are used in wireless networks.

#### 3.2.1. Trojans

Trojans are malicious executable programs developed to break the security system of a computer or a network. A Trojan resides in a system as a benign program file. Once the user attempts to open the file, the Trojan is executed, and some dangerous action is

performed. Victims generally unknowingly download the Trojan from multiple sources such as (i) the Internet, (ii) an FTP archive, (iii) via peer-to-peer file exchange using IRC, and (iv) Internet messaging. Typically, Trojans are of seven distinct types: (a) Remote access Trojans (b) Sending Trojans (c) Destructive Trojans (d) Proxy Trojans (e) FTP Trojans (f) Security software disable Trojans and (g) DoS Trojans.

*Remote access Trojans* are malware programs that use backdoors to control the target machine with administrative privilege. These type of Trojans are downloaded invisibly with a user request for a program such as a game or an email attachment. Once the attacker compromises a machine, the Trojan uses this machine to compromise more machines to construct a BotNet for launching a DoS or DDoS attack. An example of remote access Trojan is *danger. Sending Trojans* are used to capture and provide sensitive information such as passwords, credit card information, log files, e-mail addresses, and IM contact lists to the attacker. In order to collect such information, such Trojans attempt to install a keylogger to capture and transmit all recorded keystrokes to the attacker. Examples of this type of Trojans are *Badtrans.B email virus*, and *Eblast. Destructive Trojans* are very destructive for a computer and often programmed to delete automatically some essential executable programs such as configuration and dynamic link library (DLL) files. Such Trojans act either (i) as per the instructions of a back-end server, or (ii) based on pre-installed or programmed instructions, to strike on a specific day, at a specific time. Two common examples of this type are *Bugbear virus* and *Goner worm. Proxy Trojans* attempt to use a victim's computer as a proxy server. A Trojan of this kind compromises a computer and attempts to perform malicious activities such as fradulent credit card transactions, and launching of malicious attacks against other networks. Examples of proxy Trojans are *TrojanProxy:Win32*, *Paramo.F. FTP Trojans* attempt to open port 21 and establish a connection from the victim computer to the attacker using the File Transfer Protocol (FTP). An example of FTP Trojan is *FTP99cmp. Security software disable Trojans* attempt to destroy or to thwart defense mechanisms or protection programs such as antivirus programs or firewalls. Often such a Trojan is combined with another type of Trojan as a payload. Some examples are *trojan.Win32.KillAV.ctp* and *trojan.Win32.Disable.b. DoS Trojans* attempt to flood a network instantly with useless traffic, so that it cannot provide any service. Some examples of this category of Trojan are *ping of Death*, and *teardrop*.

### 3.2.2. DoS/DDoS attacks

*Denial of service* (DoS) is a commonly found, yet serious class of attack caused due to an explicit attempt of an attacker to prevent or block legitimate users of a service from using desired resources. Such an attack occurs in both distributed as well as in a centralized setting. Some common examples of this class of attack are *SYN flooding*, *smurf*, *fraggle*, *jolt*, *land*, and *ping-of-death*.

A *Distributed Denial of Service* (DDoS) attack is a coordinated attempt on the availability of services of a victim system or a group of systems or on network resources, launched indirectly from a large number of compromised machines on the Internet. Typically, a DDoS attacker adopts an $m : 1$, i.e., many compromised machines to a single victim machine or an $m : n$ approach that makes it very difficult to detect or prevent. A DDoS attacker normally initiates such a coordinated attack using either an architecture based on agent handlers or Internet relay chat (IRC). The attacking hosts are usually personal computers with broadband connections to the Internet. These computers are compromised by viruses or Trojan programs called *bots*. These compromised computers are usually referred to as *zombies*. The actions of these zombies are controlled by remote perpetrators often through (a) *BotNet* commands and (b) a control channel such as IRC. Generally, a DDoS attack can be launched using any one of the following ways.

(i) *By degree of automation*: The attack generation steps such as recruit, exploit, infect, and use phase can be performed in three possible ways: manual, automatic, and semi-automatic.
(ii) *By exploited vulnerability*: The attacker exploits the vulnerability of a security system to deny the services provided by that system to legitimate users. In semantic attacks, it exploits a specific feature or implementation bug of some protocols or applications installed in the victim machine to overload the resources used by that machine. An example of such attack is the TCP SYN attack.
(iii) *By attack network used*: To launch a DDoS attack, an attacker may use either an agent handler network or an IRC network.
(iv) *By attack rate dynamics*: Depending on the number of agents used to generate a DDoS attack, the attack rate may be either a constant rate or a variable rate attack. Besides these, an increasing rate attack and a fluctuating rate attack can also be mounted using a rate change mechanism.
(v) *By victim type*: DDoS attacks can be generated to paralyze different types of victims. Example include application attacks, host attacks, network attacks, and infrastructure attacks.
(vi) *By impact*: Based on the impact of a DDoS attack, it may be either a disruptive or a degrading attack.
(vii) *By agent*: A DDoS attack can be generated by a constant agent set or a variable agent set.

Some statistics on DDoS attacks are shown in Fig. 4. Out of many DoS/DDoS attack generation tools, a few are discussed below.

(i) *Jolt*: This DoS attack tool sends a large number of fragmented ICMP packets to a target machine running Windows 95 or NT in such a manner that the target machine fails to reassemble them for use, and as a result, it freezes up and cannot accept any input from the keyboard or mouse. However, this attack does not cause any significant damage to the victim system, and the machine can be recovered with a simple reboot.
(ii) *Burbonic*: This DoS exploit attempts to victimize a Windows 2000 machine by sending a randomly large number of TCP packets with random settings with the purpose of increasing the load on the machines so that it leads to a crash.
(iii) *Targa*: Targa is a collection of 16 different DoS attack programs. One can launch these attacks individually as well as in a group and can damage a network instantly.
(iv) *Blast20*: This TCP service stress tool is able to identify potential weaknesses in the network servers quickly. An example for

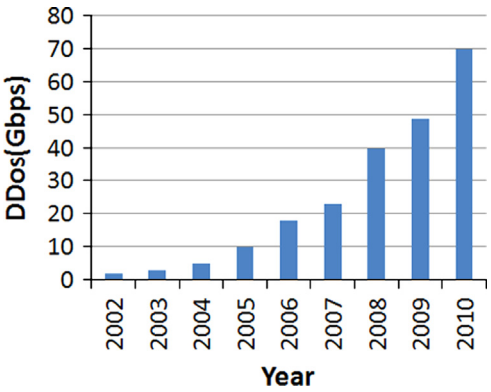


**Fig. 4.** DDoS attack graph.

the use of this tool is *% blast targetIP port start_ size end_ size /b (i.e., begin text) "GET/SOME TEXT" /e (i.e., end text) "URL"* The command is used to send attack packets of size minimum *start_size* bytes to maximum *end_size* bytes to a server with a specified target IP.

(v) *Crazy Pinger*: It attempts to launch an attack by sending a large number of ICMP packets to a victim machine or to a large remote network.

(vi) *UDPFlood*: This tool can flood a specific IP at a specific port instantly with UDP packets. The flooding rate, the maximum duration and the maximum number of packets can be specified in this tool. It can also be used to test the performance of a server.

(vii) *FSMax*: It is a server stress testing tool. To test a server for buffer overflows that may be exploited during an attack, it accepts a text file as input and executes a server through a sequence of tests based on the input.

(viii) *Nemsey*: The presence of this tool implies that a computer is insecure, and infected with malicious software. It attempts to launch an attack with a specified number of packets of specified sizes, including information such as protocol and port.

(ix) *Panther*: This UDP based DoS attack tool can flood a specified IP at a specified port instantly.

(x) *Slowloris*: It creates a large number of connections to a target victim Web server by sending partial requests, and attempts to hold them open for a long duration. As a consequence, the victim servers maintain these connections as open, consuming their maximum concurrent connection pool, which eventually compels them to deny additional legitimate connection attempts from clients.

(xi) *BlackEnergy*: This Web based DDoS attack tool, an HTTP-based BotNet, uses IRC based command and control method.

(xii) *HOIC*: It is an HTTP based DDoS tool that focuses on creation of high speed multi-threads to generate HTTP flood traffic. It is able to flood simultaneously up to 256 Web sites. The built-in scripting system in this tool allows the attacker to deploy boosters, which are scripts designed to thwart DDoS counter measures.

(xiii) *Trinoo*: This is an effective DDoS attack tool, that uses a master host and several broadcast hosts. It issues commands using TCP connection to the master host, and the master instructs the broadcast hosts via UDP to flood specific target IPs at random ports with UDP packets. To launch an attack using this tool, an attacker should possess prior access to the host to install a Trinoo master or broadcast, either by passing or by compromising the existing security system.

(xiv) *Shaft*: It is a variant of Trinoo, which provides statistics on TCP, UDP and ICMP flooding attacks. These help attackers identify the victim machine's status (i.e., completely down or alive), or to decide on the termination of zombies in addition to the attack.

(xv) *Knight*: This IRC-based tool can launch multiple DDoS attacks to create SYN flood, UDP flood, and urgent pointer flood on Windows machines.

(xvi) *Kaiten*: This is an IRC-based attack tool and is able to launch multiple attacks, viz., UDP flood, TCP flood, SYN flood, PUSH+SYN flood attacks. It uses random source addresses.

(xvii) *RefRef*: RefRef is used to exploit existing SQL injection vulnerabilities using features included in MySql such as SELECT permissions to create a DoS attack on the associated SQL server. It sends malformed SQL queries carrying payloads which force servers to exhaust their own resources. It works with a Perl compiler to launch an attack.

(xviii) *LOIC*: It is an anonymous attacking tool launched via IRC. It operates in three modes based on the three protocols: TCP, UPD, and HTTP. It exists in two versions: binary and Web-based. It uses multiple threads to launch an attack.

(xix) *Hgod*: This is a Windows XP based tool that spoofs source IPs and specifies protocols and port numbers during an attack. By default, it is used for TCP SYN flooding. An example of TCP SYN flooding attack command against 192.168.10.10 on port 80 with a spoofed address of 192.168.10.9, is *%hgod 192.168.10.10 80 -s 192.168.10*

(xx) *TFN*: Like Trinoo, TFN requires a client host and several daemon hosts. It is very effective in launching DDoS attacks, viz., ICMP flood, UDP flood, SYN flood, and smurf attacks. TFN2K, a variant of TFN, also includes some special features, such as encryption and decryption, the ability to launch stealth attacks, and DoS attacks to crash a specified target host, and to communicate shell commands to the daemons.

(xxi) *Stacheldrath*: This DDoS attacking tool is a hybridization of TFN and Trinoo, with some additional features, such as encrypted transmission between the components, and automatic updation of the daemons.

A large number of attack generation tools are available on the Internet and most are very powerful, and can be easily used to crash networks and Websites. Among these, we found that LOIC and HOIC are very adept at launching DDoS attacks within a very short time. LOIC supports TCP, UDP, and HTTP protocols to construct attack packets whereas HOIC supports only the HTTP protocol. Although TFN, Trinoo, and Stachaldraht can be used to launch a DDoS attack, they are not as powerful as LOIC. It should be noted that the use of LOIC to launch an attack in a public network is a crime.

#### 3.2.3. Packet forging attack tools

Packet forging tools are useful in forging or manipulating packet information. An attacker can generate traffic with manipulated IP addresses based on this category of tools. We describe some commonly used packet forging tools.

(i) *Packeth*: Packeth is a Linux based tool with a graphical user interface. It can send any packet or sequence of packets using raw sockets on the Ethernet. It provides a large number of options such as being able to create incorrect checksum, and wrong header length.

(ii) *Packit*: This network auditing tool allows one to customize, inject into, monitor, and manipulate IP traffic. It cab be used in various ways such as (a) to test an NIDS, (b) to evaluate the performance of a firewall, (c) to scan a network, (d) to simulate network traffic, and (e) in TCP/IP auditing.

(iii) *Packet excalibur*: This forging tool allows one to (a) sniff packets, (b) build and receive custom packets, and (c) to spoof packets. It has a graphical interface to help a user build scripts as a text file and to specify additional protocols.

(iv) *Nemesis*: This Unix-like and Windows based network packet crafting and injection tool is useful for testing any NIDS, firewall or IP stack and a variety of other tasks. This command-line driven tool also provides an option for scripting. Nemesis allows an attacker to craft and inject a large variety of packets. Especially in IP and the Ethernet injection modes, it allows one to craft and inject almost any custom packets.

(v) *Tcpinject*: This forging tool allows one to transmit a wide variety of TCP/IP packets by specifying multiple parameters such as source IPs, destination IPs, source ports,

destination ports, packet size, payload, TCP control flags and TCP window size.

(vi) *Libnet*: This tool provides many facilities to the application programmer including the ability to construct and inject network packets through a portable and high-level API. To support underlying packet creation and injection functionality, it uses the libnet utility.

(vii) *SendIP*: This command-line forging tool allows one to send arbitrary IP packets with a large number of options to specify the content of every header of a specific packet. Any data can be added to the packet during transmission.

(viii) *IPsorcery*: This TCP/IP packet generating tool has the ability to send TCP, UDP and ICMP packets using a GTK+ interface.

(ix) *Pacgen*: This Linux based Ethernet IP TCP/UDP packet generating tool allows an attacker to generate custom packets with configurable Ethernet, IP, TCP and UDP layers as well as custom payloads. It also includes additional features such as the ability to generate a *packet count*, and a *programmable time interval* between packets sent.

(x) *ARP-SK*: ARP Swiss Knife (ARP-SK) allows one to create totally arbitrary ARP requests to manipulate ARP packets, and to test network security and connectivity.

(xi) *ARPspoof*: This tool is also known as ARP Cache Poisoning. It allows one to spoof the contents of an ARP table on a remote computer on the LAN. Two addresses are used to establish connection between two computers on an IP/Ethernet network: (a) the MAC address, which is used on a local area network before packets go out of the gateway, and (b) the IP address, which is used to surf the Internet through a gateway.

(xii) *Libpal*: This user-friendly packet assembly library provides utilities to build and send forged Ethernet, IP, ICMP, TCP and UDP packets. It uses a structure to represent a packet.

(xiii) *Aicmpsend*: This ICMP packet sending tool supports several features including ICMP flooding and spoofing. It allows one to implement all the ICMP flags and codes.

Based on our study of packet forging tools, we note that Nemesis is widely used to generate custom packets using different protocols. It supports most protocols such as ARP, DNS, ICMP, IGMP, IP, OSPF, RIP, TCP and UDP. This makes it very effective compared to other tools. Other advantages of this tool are that: (a) anyone can generate custom packets from the command prompt or using shell scripts in a system, and (b) attackers find it very useful to generate attack packets.

#### 3.2.4. Application layer attack tools

In an application layer attack, the attacker uses legitimate application layer HTTP requests from legitimately connected network machines to overwhelm a Web server (Xie and Yu, 2009). The application layer attack may generate a session flooding attack, request a flooding attack or an asymmetric attack (Ranjan et al., 2006; Yu et al., 2007). Application layer DDoS attacks are more subtle than network layer attacks and the detection of application layer attacks is difficult because they use legitimate protocols and legitimate connections. Application layer attacks are of four types given below.

(i) *HTTP-related attacks*: In this attack, a massive amount of HTTP requests are sent to overwhelm the target site in a very short time frame. Some commonly used tools of this category are Code Red Worm and its mutations, Nimda Worm and its mutations, Cross site scripting attacks, Malicious URLs and AppDDoS.

(ii) *SMTP-related attacks*: SMTP protocol is used to transmit emails over Internet. The attacker tries to attack a mail server using this Internet mail transfer protocol. Some example attack tools of this category are SMTP mail flooding, SMTP worms and their mutations, extended relay attacks, and firewall traversal attacks.

(iii) *FTP-related attacks*: The first step in this attack is to initiate a legitimate FTP connection and then send some attack packets to the victim. Examples include FTP bounce attacks, FTP port injection attacks, passive FTP attacks, and TCP segmentation attacks.

(iv) *SNMP-related attacks*: The main goal of an SNMP attack is to change the configuration of a system and then monitor the state of availability of the system. Examples of this category of attacks include SNMP flooding attacks, default community attacks, and SNMP put attacks.

#### 3.2.5. Fingerprinting attack tools

Fingerprinting tools are used to identify specific features of a network protocol implementation by analyzing its input and output behavior. The identified features include protocol version, vendor information and configurable parameters. Fingerprinting tools are used to identify the operating system running on a remote machine and can also be used for other purposes. Existing fingerprinting tools show that implementations of most key Internet protocols such as ICMP, TCP, TELNET and HTTP (Beverly, 2004; Shah, 2004; Yarochkin, 1998) have bugs. Network administrators can use remote fingerprinting to collect information to facilitate management, and an intrusion detection system can capture the abnormal behavior of attackers or worms by analyzing their fingerprints (Singh et al., 2004).

(i) *Nmap*: Nmap is one of the best fingerprinting tools for both Unix and Windows operating systems. It is very useful in network mapping as well as information gathering from a remote machine on a network as described in Section 3.1.2.

(ii) *P0f*: P0f is an OS fingerprinting tool that uses passive fingerprinting in contrast to active fingerprinting performed by Nmap. Passive fingerprinting simply sniffs the network and classifies the host based on the observed traffic. This is more difficult than active fingerprinting, since one has to accept whatever communication happens rather than designing custom probes.

(iii) *Xprobe*: This OS fingerprinting tool is used to find the operating system run by a remote machine. Xprobe is similar to Nmap and it exploits the ICMP protocol in its fingerprinting approach.

(iv) *CronOS*: This fingerprinting tool is used to determine the operating system of a target machine. This tool is embedded in Namp-CronOS and it has three options to perform operations. The *S* option guesses the timeout period of SYN_RCVD states, the *I* option determines the last ACK state timeout and the *f* option uses FIN_WAIT_1 state timeout for fingerprinting.

(v) *Queso*: This utility runs on Linux and Solaris operating systems. It is used to remotely determine the operating system's version and manufacturer information by analyzing network packets. It provides precise information about a network or a system by scanning the network.

(vi) *AmapV4.8*: The Amap fingerprinting tool identifies applications and services by creating bogus communication without listening on default ports. It maintains a database of all the known applications, including non-ASCII based applications and enterprise services.

(vii) *Disco*: The Disco fingerprinting tool is used to discover unique IP addresses on a network. In addition to IP discovery, it also fingerprints TCP SYN packets.

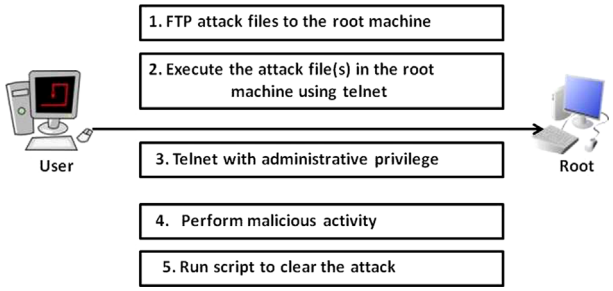


**Fig. 5.** Steps in U2R attack.

(viii) *Sprint*: This fingerprinting tool is used to identify the operating system running on a machine. In addition, sprint also has the ability to calculate up times and contains an advanced banner grepping functionality. Sprint, when run with $-n$ switch, simulates netcraft.

Among the fingerprinting tools discussed above, Nmap is the best due to its multiple functionalities and superior effectiveness compared to the others. Nmap provides detailed information on a network or a host with a maximum amount of vulnerable information. P0f supports passive scanning whereas Xprobe and Queso[2] are used for remote operations.

### 3.2.6. User attack tools

In user attacks (Lippmann and Cunningham, 2000), either the attacker (a) attempts as a normal legitimate user to gain the privileges of a root or superuser, or (b) attempts to access a local machine by exploiting its vulnerabilities without having an account on that machine. Both types of attempts are very difficult to detect because their behavior resembles normal characteristics. We discuss these attacks by category along with launching tools.

(i) *U2R attack*: In this attack, as shown in Fig. 5, the attacker initially attempts to gain access to the local victim machine as a legitimate user. The means may be a password sniffing attempt, dictionary attack, or any social engineering approach. The attacker then explores possible vulnerabilities or bugs associated with the operating system running on the victim machine to perform the transition from user to superuser or root level. Once root privileges are acquired, the attacker possesses full control of the victim machine to install backdoor entries for future exploits, manipulate system files to gather information, and other damaging actions. Two well-known U2R attack tools are described next.
   (a) *Yaga*: This tool is used to create a new administrator account by compromising registry files. The attacker edits the registry file to crash some system services on the victim machine and create a new administrator account.
   (b) *SQL attack*: Here, the attacker creates a TCP connection with an SQL database server on a Unix machine. The database shell exits when a special escape sequence is issued and the root shell of the machine is started by running the Perlmagic[3] script.

(ii) *R2L Attack*: In this attack, a remote attacker, without an account on a local machine, attempts to send packets to that machine by gaining local access based on the vulnerabilities of that machine. To gain access to the local machine, the attacker attempts various ways as shown in Fig. 6. Two such ways are (a) using online and offline dictionary attacks to acquire the password to access the machine, and (b) making repeated guesses at possible usernames and passwords. The attacker also attempts to take advantage of those legitimate users who are often casual in choosing their passwords. Below are two R2L attack tools.
   (a) *Netcat*: This R2L attack tool uses a Trojan program to install and run Netcat on the victim machine at port number 53. The Netcat program works as a backdoor to access the machine using Netcat port without any username and password.
   (b) *ntfsdos*: The attacker gains the console of a WinNT machine by running ntfsdos. The program mounts the machine's disk drives. Thus the attacker is able to copy secret files on the secondary media.

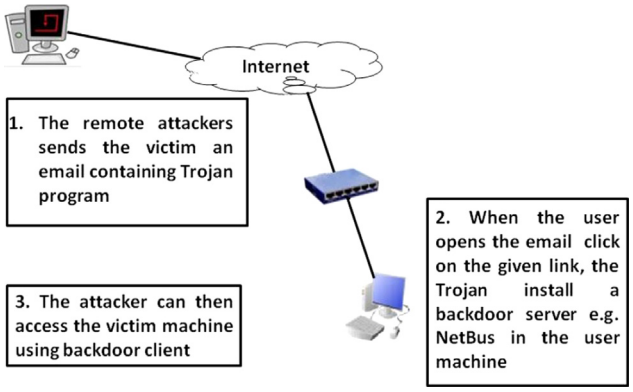


**Fig. 6.** Steps in R2L attack.

### 3.2.7. Other attack tools

In addition to the tools reported in the preceding sections, there are other tools that have direct or indirect use in the attack launching process. We discuss some of these to increase the awareness of learners and security researchers.

(i) *Ping*: This tool is used to test network connectivity or reachability of a host on an IP network. Ping is a pioneering tool developed to check a computer or router, and Internet connectivity. The ping request is sent to a particular host or to a network using command prompt. As a reply it displays the response of the destination host and how long it takes to receive a reply. It uses the ICMP protocol, which has low priority and slower speed than regular network traffic.

(ii) *Hping2*: It is used to send custom TCP/IP packets and display reply messages received from the target. It handles fragmentation and arbitrary packet size, and can also be used to transfer files. It performs firewall rule testing, port scanning, protocol based network performance testing, and path MTU discovery.

(iii) *Hping3*: This tool works almost like Hping2 and can handle fragmentation with arbitrary packet size. It finds the sequence number for reply packets from the source port. It starts with a base source port number and increases this number as packets are sent. The default base source port is random. The source port number may be kept constant for each packet sent.

(iv) *Traceroute*: Traceroute is used to show the route between two systems in a network. It lists all intermediate routers from the source end to the destination end. Using this tool, one determines how systems are connected to each other or how an Internet service provider connects to the Internet to

[2] http://spot-act.heck.in/queso-scanner-v-0-5.xhtml
[3] http://www.perlmagic.org/

provide services. The traceroute program is available in most computers including most Unix systems, Mac OS and Windows OS.

(v) *Tctrace*: Though tctrace is similar to traceroute, it uses TCP SYN packets to trace. This makes it possible for one to trace through firewalls if one knows a TCP service that is allowed to pass from the outside.

(vi) *Tcptraceroute*: Tcptraceroute sends either UDP or ICMP ECHO request packets using a TTL field, which is incremented by one with each hop until the destination is reached. It shows the path that a packet has traversed to reach the destination. However, due to the widespread use of firewall filters it may not be able to complete the path to the destination.

(vii) *Traceproto*: Traceproto is similar to traceroute, but this tool allows the user to choose protocols to be traced. It currently allows TCP, UDP and ICMP protocol trace. It can be used to test and bypass firewalls, packet filters and check if ports are open. Traceproto is actually a traceroute replacement tool written in C.

(viii) *Fping*: Fping uses ICMP protocol to determine whether a host is active or not. Fping is more powerful than ping because it can scan any number of hosts or a file containing the list of hosts. Instead of trying one host until it times out or replies, fping sends out a ping packet and moves to the next host in a round-robin fashion. If a host replies, it is noted and removed from the list of hosts to check. If a host does not respond within a certain time limit and/or retry limit, it is considered unreachable. Unlike ping, fping is meant to be used in scripts and its output is easy to parse.

(ix) *Arping*: The arping tool is used in the Linux platform to send ARP request messages to a destination host in a LAN. It is used to test whether an IP address is in use or not.

### 3.3. Network monitoring tools

Monitoring of network traffic is an essential activity for network defenders in order to observe, analyze and finally identify any anomalies occurring in the network. In support of such activities of network defenders as well as to assist in meaningful interpretation of the outcomes of their analysis, network monitoring and analysis tools play an important role. Rapid incidences of malicious attempts to compromise the confidentiality, integrity and access control mechanisms of a system or to prevent legitimate users of a service from accessing the requested resources have led to an increased demand for developing useful tools to visualize network traffic in a meaningful manner to support subsequent analysis.

#### 3.3.1. Visualization and analysis tools

An effective network traffic (both packet and NetFlow traffic) visualization tool can be of significant help for network defenders in monitoring and analysis tasks. Appropriate visualization not only supports meaningful interpretation of the analysis results, but also assists security managers in identifying anomalous patterns. It also helps in taking appropriate action to mitigate attacks before they propagate and infect other parts of the network. Some visualization tools are discussed below.

(i) *Tnv*: This time-based traffic visualization tool presents packet details and links among local and remote hosts. It assists in learning the normal patterns in a network, investigating packet details, and in network troubleshooting. Tnv provides multiple services to support inspection and analysis activities: (a) opening and reading libpcap files, (b) capturing live packets, and (c) saving captured data in a MYSQL database.

(ii) *Network Traffic Monitor*: This tool provides support in presenting and scanning detailed traffic scenarios since the inception of an application process and also allows analyzing traffic details.

(iii) *Rumint*: This tool enables visualization of live captured traffic. It can also save captured traffic as a pcap file in the Windows environment.

(iv) *EtherApe*: EtherApe allows one to sniff live packets and to monitor captured data in the Unix environment.

(v) *NetGrok*: It is a real-time network visualization tool that presents a graphical layout and a tree map to support visual description of the network data. It can visualize live packets, capture traces, and help in filtering.

(vi) *NetViewer*: This tool not only supports observation of captured live traffic in an aggregated way, but also helps identify network anomalies. In addition, NetViewer supports visualization of useful traffic characteristics to assist in tuning of defense mechanisms.

(vii) *VizNet*: It helps visualize the performance of a network based on bandwidth utilization.

Most visualization tools discussed above support both visualization and analysis of network traffic. To visualize and also to analyze a network, one can use EtherApe in Unix or NetViewer in the Windows platform. For real-time visualization of live traffic for intrusion detection, NetViewer is the best due to its ability to detect anomalous network traffic. Network defenders need real-time visualization tools that can detect abnormal behaviors in network traffic and immediately generate alert messages to inform the administrator.

## 4. Attack detection systems

Attack detection systems or intrusion detection systems are essential to keep enterprise networks secure. Many IDSs have been developed with diverse facilities. However, there is still need for a complete real-time solution with novel attack detection capability. We describe here some popular IDSs with architecture for a selected few. We also present a comparison among them. The intrusion detection systems that we discuss attempt to identify known as well as unknown attacks using statistical, data mining or soft computing approaches.

(i) ADAM (Daniel et al., 2001): Automated Data Analysis and Mining (ADAM) includes a data mining based technique to detect intrusions or attacks within a testbed. It combines association rule mining and classification to discover attacks in tcpdump audit trail data. ADAM trains the classifier to classify suspicious connections as either a known type of attack or an unknown type or a false alarm with respect to the existing rules or profiles.

(ii) MINDS (Ertoz et al., 2004): Minnesota Intrusion Detection System (MINDS) is another popular data mining based system for detecting network attacks or intrusions. The architecture of MINDS is shown in Fig. 7. It takes NetFlow version 5 data collected through flow tools. Before entering the anomaly detection module, a data filtering step is executed to remove non-interesting network traffic patterns. The anomaly detection engine uses an outlier detection algorithm to assign an anomaly score to each network connection. Finally, it reports alarms for any malicious activity based on the anomaly scores.

(iii) DNIDS (Kuang, 2007): The Dependable Network Intrusion Detection System (DNIDS) uses a combined strangeness and isolation measure with the k-nearest neighbor (CSI-KNN)

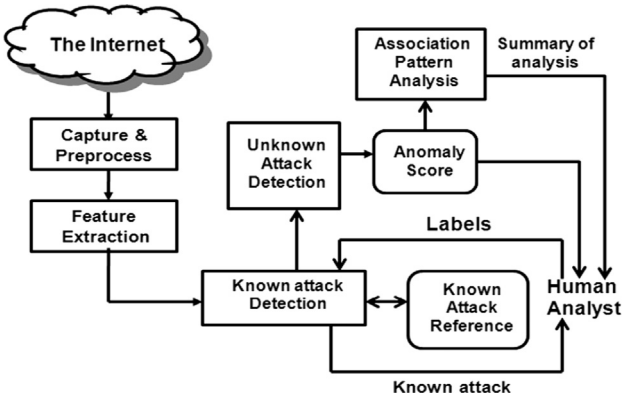


**Fig. 7.** Architecture of MINDS.

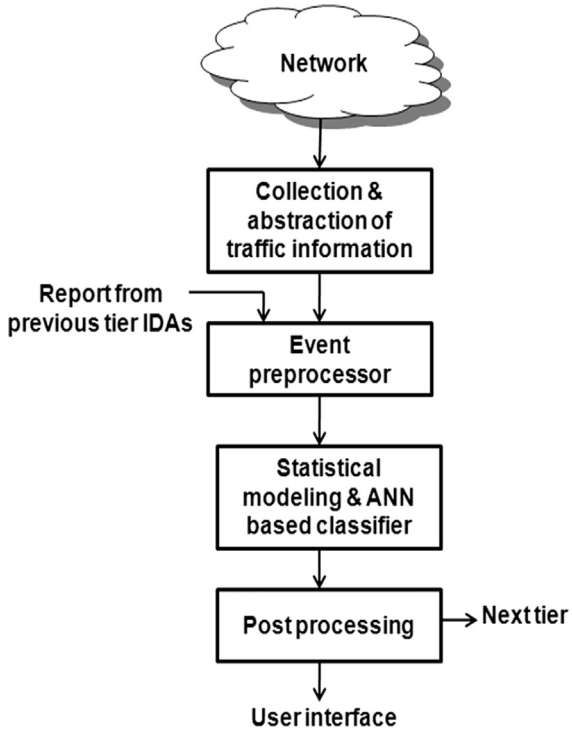


**Fig. 8.** Architecture of HIDE.

algorithm. DNIDS can detect network intrusions while providing continued service even under attacks. The intrusion detection algorithm analyzes characteristics of network data by employing two measures, strangeness and isolation. Based on these measures, a correlation unit raises intrusion alerts with associated confidence estimates. Multiple CSI-KNN classifiers work in parallel to deal with different types of network traffic and report alarm for any abnormal traffic patterns.

(iv) HIDE (Zhang et al., 2001): HIDE is a hierarchical anomaly based system, developed using statistical modeling and neural networks. See Fig. 8 for architecture of the system. It consists of several tiers, each tier containing several Intrusion Detection Agents (IDAs), which are IDS components that monitor activities of a host or a network. The statistical processor maintains a reference model of typical network activities, compares reports from the event preprocessor with the reference model. It forms a stimulus vector to feed into the neural network classifier that analyzes the vector to decide whether the network traffic is normal or attack. The post-processor generates reports for agents at higher tiers.

(v) NSOM (Labib and Vemuri, 2002): Network Self-Organizing Maps (NSOM) is a self-organizing map (SOM) based IDS that attempts to detect anomalies by quantifying the usual or acceptable behavior and by flagging irregular behavior as potentially intrusive. NSOM is used to classify real-time Ethernet network traffic data. It collects network traffic data continuously from a network port, preprocesses and selects suitable features to classify them as attack or normal.

(vi) FSAS (Song et al., 2006): Flow-based Statistical Aggregation Scheme for Network Anomaly Detection (FSAS) has a two-layered architecture containing a feature generator and a flow-based detector. The feature generator collects network traffic data from a host or a network and the event time module periodically calls the feature extraction module to convert the flow statistic information into the format required by the model. The feature scoring metric calculates the probability scores of these features by comparing the features with the reference model generated by past normal and attack users. Higher the maliciousness of a flow, the higher is the possibility of the flow being an attack. FSAS provides 22 significant features relevant for DoS attack detection.

(vii) N@G (Subramoniam et al., 2005): Network at Guard (N@G) is a hybrid IDS that contains both network and host sensors. It analyzes the audit trail using statistical techniques as part of the host sensor. The system has a management console to aggregate alerts from various sensors using a user interface, a middle tier and a data management component. It provides real-time protection to client components against malicious traffic, which can include unsolicited changes to the Windows hosts file, and Windows messenger service. It also provides Layered Service Provider (LSP) and Domain Name Server (DNS) protection. The system can dynamically apply access control to routers (Cisco) to actively block network attacks.

(viii) FIRE (Dickerson, 2000): Fuzzy Intrusion Recognition Engine (FIRE) is a fuzzy logic based anomaly IDS to assess malicious activity. The system combines simple network traffic metrics with fuzzy rules to determine the likelihood of specific or general network attacks. FIRE relies on fuzzy network traffic profiles as inputs to its rule set and uses simple data mining techniques to process the network input data for anomaly detection.

(ix) NFIDS (Mohajerani et al., 2003): NFIDS is a hierarchical neuro-fuzzy anomaly IDS that is composed of several autonomous agents and three tiers. Tier-I contains some Intrusion Detection Agents (IDAs) that monitor activities of a host or a network and report abnormal behavior to Tier-II. Tier-III combines correlation data, higher-level reports and sends alarm to the user interface. This system uses a decision making process to detect intrusions based on fuzzy rules and neural networks. Fuzzy rules are created using expert knowledge of system administrators to represent common types of attacks. Fuzzy rules can be learned by the neural network structure and based on the membership values of the fuzzy rules, different attacks are detected.

(x) D-WARD (Mirkovic and Reiher, 2005): D-WARD is an adaptive source-end DDoS defense system that can detect attacks autonomously and give surgically accurate response using its traffic profiling techniques. The architecture of D-WARD is shown in Fig. 9. D-WARD inflicts very low collateral damage to legitimate traffic, while quickly detecting and severely rate-limiting outgoing attacks.

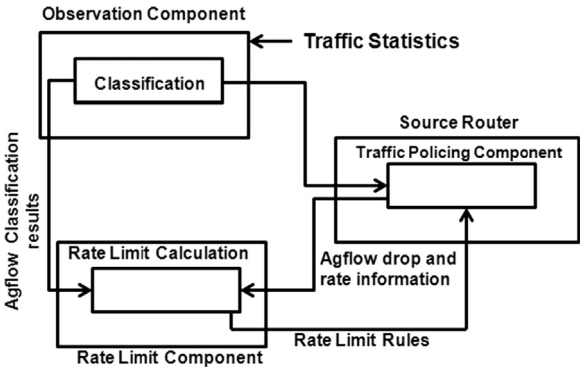


**Fig. 9.** Architecture of D-WARD.

(xi) LADS (Sekar et al., 2006): Large-scale Automated DDoS detection System (LADS) is a triggered multi-stage detection system that addresses both scalability and accuracy in detecting DDoS attacks. LADS uses clustering on NetFlow data to detect DDoS attacks in a Tier-1 ISP.

(xii) ANTID (Lee and Shieh, 2005): ANTID detects and filters DDoS attacks which use spoofed packets to circumvent conventional intrusion detection schemes. The anti-DDoS scheme intends to complement, rather that replace conventional schemes by embedding in each IP packet a unique path fingerprint that represents the route an IP packet has traversed. Thus, ANTID is able to distinguish IP packets that traverse different Internet paths. A spoofed DDoS attack can be detected by observing a surge of spoofed packets. ANTID is lightweight, robust, and incrementally deployable.

(xiii) DCD (Chen et al., 2007): DCD uses Change Aggregation Trees (CAT) to detect distributed flooding attacks at flow-level. The idea is to detect abrupt traffic changes across multiple network domains at the earliest time. The system is built over attack-transit routers, which work together. Each ISP domain has a CAT server to aggregate flooding alerts reported by routers. To resolve policy conflicts at different ISP domains, a new secure infrastructure protocol (SIP) was developed to establish mutual trust or consensus.

(xiv) CERN Investigation of Network Behavior and Anomaly Detection (CINBDS) (Hulboj and Jurga, 2009): The main objective of this project is to detect anomalies in a network and automatically take countermeasures. It works in high speed networks. It uses the sFlow (Li et al., 2013) standard for monitoring a high speed network and reduce occurrences of false positive alarms. However, for unknown anomalies this system generates false positives.

(xv) NetSTAT (Vigna and Kemmerer, 1999): It is a network based intrusion detection system that uses the state transition analysis technique. It operates on a high volume network. State transition analysis describes computer penetrations as a sequence of actions performed by an attacker to compromise a system. This system uses autonomous intrusion detection components known as probes. Probes are used to monitor network traffic and a filter module is used to select the messages that contribute assertions in a state transition scenario. If a single probe can detect all types of attacks, it does not interact with the analyzer. Otherwise, the analyzer decomposes an intrusion scenario into subscenarios so that each one can be detected by a single probe.

(xvi) Bro (Paxson, 1999): It is a real-time stand-alone system used for detecting network intruders. Bro is divided into an "event engine" that reduces a kernel filtered network traffic stream into a series of high level events, and a "policy script interpreter" that interprets event handlers written in a specialized language used to express a site's security policy. It detects intrusions by parsing network traffic to extract application level semantics. The Bro analyzer filters network traffic and removes unnecessary elements. The remaining information is sent to the event engine and Bro interprets the structure of network packets and abstracts them into high-level events that are analyzed by policy script interpreter to detect malicious activities.

(xvii) Snort (Roesch, 1999): Snort is a cross-platform, lightweight network intrusion detection system that can detect a wide variety of traffic anomalies over TCP/IP networks. It is useful when it is not cost efficient to deploy a commercial NIDS. Though Snort operates in a manner similar to tcpdump, it can inspect packet payload information. It decodes the application layer packets and uses rules to collect traffic that has specific patterns in the application layer. The decoded output produced by Snort is more user friendly than tcpdump. To detect network anomalies, Snort uses rules and matching algorithms.

(xviii) PAYL (Wang and Stolfo, 2004): It is a payload based intrusion detection system that uses statistical measures to detect anomaly patterns. It makes a profile of byte frequency distribution and the standard deviation of payloads to a single host or port for normal data. During anomaly detection, it captures incoming payloads and computes the Mahalanobis distance metric from the normal. Any new test payload found to be too distant from the normally expected payload is deemed anomalous and an alert is generated.

(xix) ALERT-ID (Chu et al., 2012): It is a network based intrusion detection system that oversees activities of a network and generates an alarm on detecting malicious patterns. It detects anomaly based on comparison of real-time captured traffic from switch/router logs and profiles constructed from historical data. This system effectively identifies potential intrusions and misuses with an acceptable overall alarm rate.

(xx) MAD-IDS (Brahmi et al., 2010): MAD-IDS is a distributed intrusion detection system that exploits the features obtained from mobile agent methodology. It depends on (a) a misuse detection mobile agent to detect known attacks and (b) an anomaly detection mobile agent to detect unknown attacks. The mobile agents provide high accuracy for predicting different behaviors in network traffic using the data mining techniques.

(xxi) ML-DIDS (Uddin et al., 2013): It is a signature-based multilayer distributed IDS that uses mobile agents. It can detect threats by dynamically creating efficient multiple small databases for signatures. At the same time, it provides a mechanism to update these multiple small signature databases at regular intervals using mobile agents with high detection rate. (Tables 4–9)

A comparison of detection systems based on parameters such as detection type (host based, network based or both), detection approach (misuse, anomaly or both), nature of detection (online or offline), nature of processing (centralized or distributed), data gathering mechanism (centralized or distributed) and the technical approach for analysis, is given in Table 10. We make the following observations regarding attack detection systems.

- A detection system with high accuracy and low false alarm rate may often fail to operate in real-time. However, if one

**Table 4**
Comparison of attacking tools.

| Tool's name | Input | Protocol | Purpose | Effectiveness | Sources |
|---|---|---|---|---|---|
| Jolt | T | IC | DoS | Uses 100% CPU time | www.flylib.com |
| Burbonic | T | C | DoS | Multi-platform, easy to use | www.packetstormsecurity.org |
| Targa | T | C/U/IC | DoS | Very efficient | www.packetstormsecurity.org |
| Blas20 | T | C | DoS | Multi-platform, performs quick damage to a system | |
| Crazy Pinger | V | IC | DoS | Multi-platform, easy to use | www.softwaretopic.informer.com |
| UDPFlood | V | U | DoS | Windows based, less powerful | www.foundstone.com |
| FSMax | F | | DoS | Windows based, efficient for server testing | www.brothersoft.com |
| Nemsey | T/p | C | DoS | Windows based | packetstormsecurity.org |
| Panther | T/p | U | DoS | Easy to use | www.bestspywarescanner.net |
| Land & LaTierra | T | C | DoS | Powerful for land attack | |
| Slowloris | T | HT | DoS | Powerful for HTTP attack | www.ha.ckers.org/slowloris |
| Blackenergy | S | C/U/ IC | DDoS | Simple and powerful for DDoS | www.airdemon.net |
| HOIC | T | HT | DDoS | Very effective for DDoS | www.rapidshare.com |
| Shaft | V | U/C/IC | DDoS | Multi-platform, commonly used | |
| Knight | V | C/U | DDoS | Less powerful | www.cert.org |
| Kaiten | V | U/C | DDoS | Windows based | www.mcafee.com |
| RefRef | T | | DDoS | Effective for DDoS | www.hackingalert.net |
| Hgod | T/p | C/U/IC | DDoS | Easy to use | www.flylib.com |
| LOIC | T/p | C/U/ IC | DDoS | Very effective, powerful for flooding attack | www.sourceforge.net |
| Trinoo | T/p | U | DDoS | Multi-platform, easy to use | www.nanog.org |
| TFN | T | U/C/IC | DDoS | Multi-platform, effective for flooding attacks | www.codeforge.com |
| TFN2K | T | U/C/IC | DDoS | Simple and easy to execute | www.goitworld.com |
| Stachaldraht | T | C | DDoS | Multi-platform, supports more features | www.packetstormsecurity.org |
| Mstream | T | C | DDoS | Multi-platform and more primitive | www.ks.uiuc.edu |
| Trinity | T | C/U | DDoS | Very effective to compromise hosts | www.garykessler.net |

Here, T – target IP, V – victim IP, S – server IP, C – TCP, U – UDP, IC – ICMP, F – input text file, p – port, HT – HTTP.

**Table 5**
Comparison of packet forging attack tools.

| Tool's name | Input | Protocol | Purpose | Effectiveness | Sources |
|---|---|---|---|---|---|
| Packeth | S/D | U/C/IC | Packet generator | Supports many features | www.sourceforge.net |
| Packit | I | C/U/IC | Packet analysis and injection | Supports more than 60 options | www.packetfactory.openwall.ne |
| Packet Excalibur | | | | Multi-platform | www.freecode.com |
| Nemesis | H | IC/IG/ C/U | Packet crafting/injection | Multi-platform, powerful | www.sourceforge.net |
| Tcpinject | H | C | Packet generator | Linux based, easy to use | www.packetstormsecurity.org |
| Libnet | H | C | Packet injection | Portable, efficient and easy to use | www.packetfactory.net/libnet |
| SendIP | H | C/U | Packet generator | Supports many options | www.softpedia.com |
| IPsocery | H | C/U/ IC/IG | Packet generator | Easy to use | www.tools.l0t3k.net |
| Pacgen | H | C/U | Packet generator | Simple packet generator | www.sourceforge.net |
| Arp-sk | H | A | ARP packet generator | Sensitive for ARP attack | www.arp-sk.org |
| ARP-SPOOF | T | A | Packet intercept | Efficient for ARP cache poisoning | www.sourceforge.net |
| Libpal | H | C/U/IC | Packet intercept | User friendly | www.sourceforge.net |
| Aicmspend | T | IC | ICMP packet flooding | Supports many features | www.packetstormsecurity.nl |

Here, S – source IP, D – destination IP, I – interface ID, H – host IP, T – target IP, C – TCP, U – UDP, IC – ICMP, IG – IGMP, A – ARP.

**Table 6**
Comparison of fingerprinting tools.

| Tool's name | Input | Protocol | Purpose | Effectiveness | Sources |
|---|---|---|---|---|---|
| Nmap | H/N | C/U | Network scanning | Easy to use, powerful, widely used | www.nmap.org |
| P0f | H | C | Remote network identification | Passive fingerprinting, difficult to use | www.lcamtuf.coredump.cx |
| Xprobe | T | C/U | Port scanning | Simple as nmap, powerful | www.sourceforge.net |
| CronOS | T | C | OS identification | Linux based | pen-testing.sans.org |
| Queso | H | C/U | OS fingerprinting | Multi-platform, widely used | www.tools.l0t3k.net |
| AmapV4.8 | T/p | C/U | Host and port scanning | Powerful application mapper | www.linux.softpedia.com |
| Disco | | C | IP discovery and Fingerprinting | Support large number of features | www.tools.l0t3k.net |
| Sprint | H | C | OS identification | Simple and efficient | www.safemode.org |

Here, H – host IP, N – network IP, T – target IP, C – TCP, U – UDP, p – port.

compromises on the false alarm rate, one can build effective real-time systems.

- A detection system should be capable of not only detecting unknown attacks, but also be able to modify the normal as well as attack profiles dynamically.

## 5. Observations and conclusions

Based on our extended study of network security tools and systems available for anomaly based NIDS, we make the following observations.

**Table 7**
Comparison of other tools.

| Tool's name | Input | Protocol | Purpose | Effectiveness | Sources |
|---|---|---|---|---|---|
| Ping | H | IC | Host discovery | Commonly used, Easy to use | www.download.cnet.com |
| Hping2 | T | IC/C/U | Port scanning | Supports many features | www.hping.org. |
| Hping3 | T | IC/C/U | Port scanning | Powerful for network testing | www.hping.org. |
| Traceroute | H | IC/C/U | Route discovery | Multi-platform, easy to use | www.brothersoft.com |
| Tctrace | H | C | Route discovery | Multi-platform, easy to use | www.tcptrace.org |
| Tcptrace route | I | C | DNS lookup | Powerful for route discovery | www.michael.toren.net |
| Traceproto | H | C/U/IC | Route discovery | Effective for firewall testing | www.traceproto.sourceforge.net |
| Fping | T | IC | Target host discovery | More powerful than ping | www.softpedia.com |
| Arping | S | A | Send ARP request | Linux based, efficient | www.linux.softpedia.com |

Here, H – host IP, T – target IP, I – interface ID, S – source IP, C – TCP, U – UDP, IC – ICMP, A – ARP.

**Table 8**
Comparison of visualization tools.

| Tool's name | Input | Protocol | Purpose | Effectiveness | Sources |
|---|---|---|---|---|---|
| Tnv | F | C/U/ IC | Traffic visualization | Supported by all OSs | www.tnv.sourceforge.net |
| Network Traffic Monitor 1.02 | H | C/U/ IC | Live traffic monitoring | Easy to use, efficient for visualization | www.monitor-network-traffic.winsite.com |
| Rumint | H | C/U/ IC/IG | Visualize life traffic | Extremely flexible | www.rumint.org |
| EtherApe | H | C | Traffic flow visualization | Very simple and powerful | www.brothersoft.com |
| Netgrok | H | C/U/ IC | Real-time traffic visualization | Multi-platform and easy to use | www.softpedia.com |
| Netviewer | H | C/U | Traffic analysis | Powerful defense tool | www.brothersoft.com |
| VizNet | H | C/U | Traffic analysis and visualization | Efficient for visualization | www.viznet.ac.uk |

Here, H – host IP, F – captured data file, C – TCP, U – UDP, IC – ICMP, IG – IGMP.

**Table 9**
Tools by category.

| Category | Tool name | Source |
|---|---|---|
| Trojans | NukeNabbler | http://community.norton.com |
| | AIMSpy | http://www.securitystronghold.com |
| | NetSpy | http://www.netspy-trojan-horse.downloads |
| Information gathering tools | ASS | http://www.manpages.ubuntu.com |
| | NMap | http://www.nmap.org |
| | p0f | http://www.lcamtuf.coredump.cx/p0f.shtml |
| | MingSweeper | http://www.hoobie.net/~mingsweeper |
| | THC Amap | http://www.freeworld.thc.org/thc-amap |
| | Angry IP Scanner | http://www.angryziber.com/w/Download |
| DoS attack tools | Targa | http://www.security-science.com/ |
| | Burbonic | http://www.softpedia.com |
| | Blast20 | http://seomagz.com/2010/03/dos-denial-of-service-attack-tools-ethical-hacking-session-3/ |
| Spoofing attack tools | Engage Packet Builder | http://www.engage-packet-builder.software.informer.com/ |
| | Hping | http://www.hping.org |
| | Nemesis | http://www.nemesis.sourceforge.net |
| | PacketExcalibur | http://www.linux.softpedia.com |
| | Scapy | http://www.softpedia.com |
| TCP session hijacking tools | Firesheep | http://www.codebutler.github.com/firesheep/ |
| | Hunt | http://www.packetstormsecurity.org/sniffers/hunt |
| | Juggernaut | http://www.tools.l0t3k.net/Hijacking/1.2.tar.gz |
| | TTY Watcher | http://www.security-science.com |
| | IP Watcher | http://www.download.cnet.com |
| | Hjksuit-v0.1.99 | http://www.tools.l0t3k.net/~Hijacking/hjksuite-0.1.99.tar.gz |
| Probe attack tools | Solarwind | http://www.solarwinds.com |
| | Network Probe | http://www.softpedia.com |
| | NMap | http://nmap.org |
| Spoofing attack tools in wireless | Kismet | http://www.linux.die.ne |
| | libpcap | http://www.sourceforge.net/projects/libpcap |
| | libnet | http://www.libnet.sourceforge.net |
| | libdnet | http://www.libdnet.sourceforge.net/ |
| | libradiate | http://www.packetfactory.net/projects/~libradiate |
| Application layer attack tools | HOIC | https://www.rapidshare.com |
| | LOIC | http://www.softpedia.com |
| | RefRef | http://www.softpedia.com |

**Table 10**
Comparison of attack detection systems, where P represents the types of detection as host based (H) or network based (Net) or hybrid (H), W indicates the class of detection as misuse (M) or anomaly (A) or both (B), X corresponds to the nature of detection as real-time (R) or non-real time (N), Y and Z represent the nature of processing and data gathering mechanism as centralized (C) or distributed (D) respectively.

| System | Year | P | W | X | Y | Z | Approach |
|---|---|---|---|---|---|---|---|
| NetSTAT (Vigna and Kemmerer, 1999) | 1999 | Net | A | R | C | D | State transition approach |
| Bro (Paxson, 1999) | 1999 | Net | A | R | C | C | Behavior analysis |
| Snort (Roesch, 1999) | 1999 | Net | A | R | C | D | Rule based |
| FIRE (Dickerson, 2000) | 2000 | Net | A | N | C | C | Fuzzy logic |
| ADAM (Daniel et al., 2001) | 2001 | Net | A | R | C | C | Association rule |
| HIDE (Zhang et al., 2001) | 2001 | Net | A | R | C | D | Statistical and neural network |
| NSOM (Labib and Vemuri, 2002) | 2002 | Net | A | R | C | C | Neural networks |
| MINDS (Ertoz et al., 2004) | 2003 | Net | A | R | C | C | Outlier |
| NFIDS (Mohajerani et al., 2003) | 2003 | Net | A | N | C | C | Neuro fuzzy logic |
| N@G (Subramoniam et al., 2005) | 2003 | H | B | R | C | C | Statistical |
| PAYL (Wang and Stolfo, 2004) | 2004 | H | A | R | C | C | Statistical |
| D-WARD (Mirkovic and Reiher, 2005) | 2005 | Net | B | R | D | D | Statistical |
| ANTID (Lee and Shieh, 2005) | 2005 | Net | A | R | C | D | Path fingerprinting |
| FSAS (Song et al., 2006) | 2006 | Net | A | R | C | C | Statistical |
| LADS (Sekar et al., 2006) | 2006 | Net | A | R | C | D | Clustering based |
| DNIDS (Kuang, 2007) | 2007 | Net | A | R | C | C | K-NN |
| DCD (Chen et al., 2007) | 2007 | Net | A | R | C | D | Statistical |
| MAD-IDS (Brahmi et al., 2010) | 2010 | Net | B | R | D | D | Mobile agent based |
| ALERT-ID (Chu et al., 2012) | 2012 | Net | A | R | C | D | Rule based |
| ML-DIDS (Uddin et al., 2013) | 2013 | Net | M | R | D | D | Misuse mobile agent |

- Existing information gathering tools that scan the network work successfully in one-to-one and one-to many scenarios. However, existing tools are unsuitable for coordinated scanning (i.e., $m:1$ and $n:m$ mapping) with varying source and destination IPs, dynamically within a specified time interval.
- An integrated tool with supporting modules for capture, preprocessing, analysis, and visualization of both packet and NetFlow data is lacking. Existing tools (e.g., wireshark, Nfsen, and Nfdump) can support capture of either NetFlow traffic or packet label traffic but not both.
- Existing DDoS attack tools cannot launch attacks at multiple layers. They support launching of only single layer attacks.
- Most existing DDoS attack tools are restricted to a limited number of attack scenarios. Such tools cannot be customized to develop additional attack scenarios.
- Most existing NIDSs are dependent on several user input parameters, and their performance is highly sensitive to these parameters.
- Almost all anomaly based NIDSs perform either near real-time or offline. In addition, most suffer from a large number of false alarms.
- An effective tool to support correlation between packet traffic and NetFlow traffic is still lacking.

Based on the above observations, we have identified the following list of research challenges for network security researchers.

- It is a challenging task to develop an integrated tool to support capture, preprocessing (e.g., filtering, and feature extraction), analysis, and visualization of both packet and NetFlow traffic.
- It is also a challenging task to develop a tool for faster capture, preprocessing and extraction of all types features for network traffic corresponding to all layer protocols.
- It is a non-trivial task to develop a GUI based DDoS attack traffic generation tool that is capable of handling all possible attack scenarios for multiple layers (e.g., application and transport layers).
- Development of an anomaly based NIDS is dependent on a minimum number of user parameters and capable of handling both known as well as unknown attacks in real-time with a minimum number of false alarms is another challenging task.
- Development of a real-time detection system for both low rate and high rate DDoS attacks at the victim end without affecting legitimate users or normal service is another challenging task.

Even though there are many network security tools available in the research community, the proper use of these tools is very important in the real network security infrastructure. In this paper, we have presented three major categories of security tools depicted in Fig. 2. We started this paper with a brief description of network attacks, their characteristics and steps to perform attacks. The paper also presents a large number of tools that have become popular in recent years. We also provide architectures of some popular IDSs and compare them. Finally, we conclude with a list of observations and research challenges.

## Acknowledgments

This work is supported by Department of Information Technology (DIT) and Council of Scientific & Industrial Research (CSIR) Government of India. The research is also partially funded by the National Science Foundation (NSF), USA under Grants CNS-095876 and CNS-0851783. The authors are thankful to the funding agencies.

## References

Axelsson S. Intrusion detection systems: a survey and taxonomy. Technical Report, Chalmers University.
Aydın M, Zaim A, Ceylan K. A hybrid intrusion detection system design for computer network security. Computers and Electrical Engineering 2009;35(3):517–26.
Barber R. Hacking techniques: the tools that hackers use and how they are evolving to become more sophisticated. Computer Fraud and Security 2001;2001(3):9–12.
Beverly R. A robust classifier for passive TCP/IP fingerprinting. Passive and Active Network Measurement 2004:158–67.
Bhuyan MH, Bhattacharyya DK, Kalita JK. Surveying port scans and their detection methodologies. The Computer Journal 2011a;54:1565–81.
Bhuyan MH, Bhattacharyya DK, Kalita JK. Survey on incremental approaches for network anomaly detection. International Journal of Communication Networks and Information Security 2011b;3(3):226–39.
Bhuyan M, Bhattacharyya D, Kalita J. NADO: network anomaly detection using outlier approach. In: Proceedings of the 1st international conference on communication, computing and security. New York, NY, USA: ACM; 2011c. p. 531–6.

Bhuyan M, Bhattacharyya D, Kalita J. Network anomaly detection: methods, systems and tools. IEEE Communications Surveys and Tutorials Early Access 2013;1:1–34.
Brahmi I, Yahia SB, Poncelet P. MAD-IDS: novel intrusion detection system using mobile agents and data mining approaches. In: Proceedings of the Pacific Asia conference on intelligence and security informatics. Berlin, Heidelberg: Springer-Verlag; 2010. p. 73–6.
Chandola V, Banerjee A, Kumar V. Anomaly detection: a survey. ACM Computing Surveys 2009;41(3):15.
Chen W-H, Hsu S-H, Shen H-P. Application of SVM and ANN for intrusion detection. Computer and Operation Research 2005;32(10):2617–34.
Chen Y, Hwang K, Ku W-S. Collaborative detection of DDoS attacks over multiple network domains. IEEE Transactions on Parallel Distributed Systems 2007;18 (12):1649–62.
Chu J, Ge Z, Huber R, Ji P, Yates J, Yu Y-C. Alert-ID: analyze logs of the network element in real time for intrusion detection. In: Research in attacks, intrusions, and defenses. Springer; 2012. p. 294–313.
Conti G, Abdullah K. Passive visual fingerprinting of network attack tools. In: Proceedings of the 2004 workshop on visualization and data mining for computer security. Washington, DC, USA: ACM; 2004. p. 45–54.
Corona I, Giacinto G, Roli F. Adversarial attacks against intrusion detection systems: taxonomy, solutions and open issues. Information Sciences 2013;239:201–25.
Daniel B, Julia C, Sushil J, Ningning W. ADAM: a testbed for exploring the use of data mining in intrusion detection. ACM SIGMOD Record 2001;30(4):15–24.
Danielle L. Introduction to DSNIFF. In: Global information assurance certification paper. SANS Institute; 2002.
Debar H, Dacier M, Wespi A. Towards a taxonomy of intrusion-detection systems. Computer Networks 1999a;31(8):805–22.
Debar H, Dacier M, Wespi A. Towards a taxonomy of intrusion-detection systems. Computer Networks 1999b;31(9):805–22.
Deri L, Carbone R, Suin S. Monitoring networks using ntop. In: Proceedings of the 2001 IEEE/IFIP international symposium on integrated network management. Seattle, WA, USA, IEEE; 2001. pp. 199–212.
Dhanjani N, Clarke J. Network security tools.USA: O'Reilly Media; 2005.
Dickerson JE. Fuzzy network profiling for intrusion detection. In: Proceedings of the 19th international conference of the North American fuzzy information processing society. Atlanta; 2000. p. 301–6.
Ertoz L, Eilertson E, Lazarevic A, Tan P, Kumar V, Srivastava J, et al. MINDS-minnesota intrusion detection system. Next Generation Data Mining 2004:199–218.
Garcia-Teodoro P, Diaz-Verdejo J, Maciá-Fernández G, Vázquez E. Anomaly-based network intrusion detection: techniques, systems and challenges. Computers and Security 2009;28(1):18–28.
Gogoi P, Bhattacharyya D, Borah B, Kalita JK. A survey of outlier detection methods in network anomaly identification. Computer Journal 2011;54(4):570–88.
Hulboj MM, Jurga RE. CERN investigation of network behaviour and anomaly detection. In: Recent advances in intrusion detection. Berlin, Heidelberg: Springer; 2009. p. 353–4.
Jemili F, Zaghdoud M, Ben Ahmed M. A framework for an adaptive intrusion detection system using Bayesian network. In: Proceedings of the IEEE intelligence and security informatics; 2007. p. 66–70.
Kuang LV. DNIDS: a dependable network intrusion detection system using the CSI-KNN algorithm. Master's Thesis, Queen's University Kingston, Ontario, Canada; September 2007.
Labib K, Vemuri R. NSOM: a tool to detect denial of service attacks using self-organizing maps. Technical Report, Department of Applied Science University of California, Davis, California, USA; 2002.
Lazarevic A, Kumar V, Srivastava J. Intrusion detection: a survey. In: Managing cyber threats. Springer; 2005. p. 19–78.
Lee F-Y, Shieh S-P. Defending against spoofed DDoS attacks with path fingerprint. Computers and Security 2005;24:571–86.
Li B, Springer J, Bebis G, Hadi Gunes M. A survey of network flow applications. Journal of Network and Computer Applications 2013;36(2):567–81.
Lippmann RP, Cunningham RK. Improving intrusion detection performance using keyword selection and neural networks. Computer Networks 2000;34 (4):597–603.
Lunt TF. A survey of intrusion detection techniques. Computers and Security 1993;12(4):405–18.
Mansfield-Devine S. Anonymous: serious threat or mere annoyance?. Network Security 2011;2001(1):4–10.
Mirkovic J, Reiher P. D-ward: a source-end defense against flooding denial-of-service attacks. IEEE Transactions on Dependable Secure Computing 2005;2 (3):216–32.
Mohajerani M, Moeini A, Kianie M. NFIDS: a neuro-fuzzy intrusion detection system. In: Proceedings of the 10th IEEE international conference on electronics, circuits and systems at the University of Sharjah in Sharjah, United Arab Emirates, vol. 1; 2003. p. 348–51.
Norton D. An Ettercap primer. SANS Institute InfoSec Reading Room; 2004.
Orebaugh A, Ramirez G, Beale J. Wireshark and Ethereal network protocol analyzer toolkit. Syngress; 2006.
Paxson V. Bro: a system for detecting network intruders in real-time. Computer Networks 1999;31(23):2435–63.
Peng T, Leckie C, Ramamohanarao K. Survey of network-based defense mechanisms countering the DoS and DDoS problems. ACM Computing Surveys (CSUR) 2007;39(1):3.
Pilli ES, Joshi R, Niyogi R. Network forensic frameworks: survey and research challenges. Digital Investigation 2010;7(1):14–27.
Pras A, Sperotto A, Moura GCM, Drago I, Barbosa R, Sadre R, et al. Attacks by "anonymous" Wikileaks proponents not anonymous. Technical Report 10.41, Design and Analysis of Communication Systems Group (DACS), University of Twente, Enschede, The Netherlands; 10 December 2010.
Ranjan S, Swaminathan R, Uysal M, Knightly E. DDoS-Resilient scheduling to counter application layer attacks under imperfect detection. In: Proceedings of the 25th IEEE international conference on computer communications. Barcelona, Spain; 2006. p. 1–13.
Roesch M. Snort – lightweight intrusion detection for networks. In: Proceedings of the 13th USENIX conference on system administration. Washington; 1999. p. 229–38.
Satten C. Lossless gigabit remote packet capture with linux, University of Washington Network Systems; 2007.
Schiffman MD. Libnet 101, part 1: The primer, Guardent security digital infrastructure; 2000. p. 1–10.
Sekar V, Duffield N, Spatscheck O, van der Merwe J, Zhang H. LADS: large-scale automated DDoS detection system. In: Proceedings of the annual conference on USENIX '06 annual technical conference. Berkeley, CA, USA: USENIX Association; 2006. p. 16.
Shah S. An introduction to HTTP fingerprinting. Net-Square Solutions 2004:1–21.
Sherif JS, Dearmond TG. Intrusion detection: systems and models. In: Proceedings of the 11th IEEE international workshops on enabling technologies: infrastructure for collaborative enterprises. Rome, Italy, IEEE; 2002. p. 115–33.
Singh S, Estan C, Varghese G, Savage S. Automated worm fingerprinting. In: Proceedings of the 6th symposium on operating systems design and implementation, vol. 6. Berkeley, CA, USA: USENIX Association; 2004. p. 4.
Song S, Ling L, Manikopoulo C. Flow-based statistical aggregation for network anomaly detection. In: Proceedings of the IEEE international conference on networking, sensing and control. Florida, USA: IEEE, Ft. Lauderdale; 2006. p. 786–91.
Subramoniam N, Pawar PS, Bhatnagar M, Khedekar NS, Guntupalli S, Satyanarayana N, et al. Development of a comprehensive intrusion detection system – challenges and approaches. In: Proceedings of the 1st international conference on information systems security. Kolkata, India; 2005. p. 332–5.
Uddin M, Rehman AA, Uddin N, Memon J, Alsaqour R, Kazi S. Signature-based multi-layer distributed intrusion detection system using mobile agents. International Journal of Network Security 2013;15(1):79–87.
Vigna G, Kemmerer RA. Netstat: a network-based intrusion detection system. Journal of Computer Security 1999;7(1):37–71.
Wang K, Stolfo SJ. Anomalous payload-based network intrusion detection. In: Recent advances in intrusion detection. Springer; 2004. p. 203–22.
Xie Y, Yu SZ. Monitoring the application-layer DDoS attacks for popular websites. IEEE/ACM Transactions on Networking 2009;17(1):15–25.
Yarochkin F. Remote OS detection via TCP/IP stack fingerprinting. Phrack Magazine 1998;17(3):1–10.
Ye N, Ehiabor T, Zhang Y. First-order versus high-order stochastic models for computer intrusion detection. Quality and Reliable Engineering International 2002;18(3):243–50.
Yeung KH, Fung D, Wong KY. Tools for attacking layer 2 network infrastructure. In: Proceedings of the international multiconference of engineers and computer scientists, vol. 2. Hong Kong; 2008. p. 1–6.
Yu J, Li Z, Chen H, Chen X. A detection and offense mechanism to defend against application layer DDoS attacks. In: Proceedings of the 3rd international conference on networking and services. IEEE, National University of Defense Technology, Changsha, 2007. p. 54–60.
Zhang Z, Li J, Manikopoulos C, Jorgenson J, Ucles J. HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In: Proceedings of the 2nd annual IEEE systems, cybernetics information assurance workshop. West Point, NY, USA: IEEE Computer Society; 2001. p. 85–90.
Zhou CV, Leckie C, Karunasekera S. A survey of coordinated attacks and collaborative intrusion detection. Computers and Security 2010;29(1):124–40.