Large-scale coordinated attacks: Impact on the cloud security

Damien Riquet, Gilles Grimaud, Michaël Hauspie

2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing

Large-scale coordinated attacks: Impact on the cloud security

Damien Riquet Gilles Grimaud Michaël Hauspie

L.I.F.L. CNRS UMR 8022
Université de Lille 1, Cité Scientifique
F-59655 Villeneuve d'Ascq Cedex, France

Email: {firstname.lastname}@lifl.fr

***Abstract*—Cloud Computing has emerged as a model to process large volumetric data. Though Cloud Computing is very popular, cloud security could delay its adoption. Security of the cloud must provide data confidentiality and protection of resources. Such architecture seems to be vulnerable when confronted to distributed attacks also known as large-scale coordinated attacks.**

In this paper, we study the impact of large-scale coordinated attacks on Cloud Computing and its current security solutions. We experiment the open-source IDS Snort and a commercialized firewall using distributed portscan. Our results show that these security solutions are not designed to detect distributed attacks. Indeed, an attacker who controls about 32 hosts can easily achieve a distributed portscan without being detected.

***Index Terms*—Cloud Computing ; Security ; Firewall ; Intrusion Detection System ; Distributed attacks ; Portscan.**

I. Introduction

Cloud Computing is currently a popular model to process large data set. This model provides several layers of services according to the needs of customers. Most of Cloud Computing customers use these services either to store confidential data or to employ powerful resources in order to process large data set. This is why Cloud Computing security has to be reliable.

Security of a network is mainly provided by security devices such as Intrusion Detection Systems (IDS) or Firewalls. These complementary devices can detect intrusions and block attacks from or targeting the cloud. Actually, such devices are not formerly designed to detect distributed attacks. These, also known as large-scale coordinated attacks, could be led by splitting attacks so that security devices do not detect intrusion attempts.

In this paper, we study the impact of these attacks on Cloud Computing. We propose an overview of security devices possibly used in this context. Our work focuses on two devices, the open-source IDS Snort and a commercialized firewall[1].

We led experiments using distributed portscan with different environments : various number of scanners or targets, several TCP portscan techniques and two distribution methods. Results indicate that these devices can easily be evaded when attacks are well distributed. For example, depending on portscan techniques and distribution methods, 32 or 64 scanners are enough to achieve a distributed portscan without being detected.

[1]The company which manufactured the experimented firewall wanted to remain anonymous.

Nowadays, such resources are easily available, through botnets or after a worm outbreak or simply by renting servers on a commercial cloud solution (like Amazon EC2). This is why we are interested in detection of large-scale coordinated attacks, one of the issues that delay Cloud Computing adoption.

The paper is structured as follow. First, Section II introduces the background of this paper and presents current security solutions. Then, Section III describes distributed portscan, the large-scale coordinated attack used to experiment security solutions. Section IV and V present respectively the experimental protocol and results of experimentations. Finally, Section VI concludes this paper.

II. Background and motivation

A. Cloud computing

Lately, actors of domains like industry or research raise a common difficulty : processing of large amount of data. Cloud computing has emerged as a popular model to meet such needs.

Rimal *et al.* present Cloud Computing in [1] as the concept that addresses the next evolutionary step of distributed computing. Also, they add that Cloud Computing deals with different fundamentals like virtualization, scalability, interoperability, quality of service and fail over mechanism. We define in the following parts what is Cloud Computing and how it is secured.

1) Characteristics of the cloud: in [1], Rimal *et al.* define Cloud Computing as a structure designed to provide services to external users. These services let users concentrate on what they want to deploy rather than how to deploy it. Services can be classified into three main categories: software, platform and infrastructure services.

Application service or Software-as-a-Service (SaaS) delivers software over the internet, simplifying support and maintenance. Platform-as-a-Service (PaaS) facilitates deployment of applications without the cost and complexity of buying and managing the underlying hardware. Infrastructure-as-a-Service (IaaS) provides a platform virtualization environment along with storage and networking.

Other characteristics about the cloud are virtualization management, fault tolerance, load balancing, and security.

2) Security of the Cloud: in [1], Rimal *et al.* add that there is a growing concern about security. Users store confidential information in these architectures, and in wrong hands, it could

978-0-7695-4684-1/12 $26.00 © 2012 IEEE
DOI 10.1109/IMIS.2012.76

create a civil liability. This issue may delay Cloud Computing adoption.

Main components of security structures are firewalls and intrusion detection systems. Bellovin *et al.* describe firewalls in [2] as components placed between two networks. All the traffic between these networks must pass through the firewall. Also, firewalls filter authorized traffic, defined by local security policies. Debar *et al.* outline in [3] that the main task of Intrusion Detection System (IDS) is to monitor the usage of systems and to detect insecure states. IDSs detect attempts and active misuse by legitimate users or external parties to abuse their privileges or exploit security vulnerabilities.

Though Cloud Computing relies on datacenter structures, its security is not just about firewalls and intrusion detection systems placed more or less randomly. Cloud Computing security must be adaptable to dynamic architecture (users on the cloud may change during an attack). Moreover, cloud users may be malicious or an attacker could have the control of one of several internal hosts. Cloud security must ensure applications availability and data integrity. Finally, it must prevent DDoS, worms spreading and such large-scale coordinated attacks (even inside Cloud Computing).

Due to their location (at the border of the network), firewalls would not be able to detect those attacks. Indeed, firewalls would not see attacks inside the network. IDSs, well-placed in the network, are likely to be components that detect coordinated attacks. That is why our work focuses on IDSs. Our experiments study current IDSs faced to large-scale coordinated attacks.

B. Intrusion detection system

Security management can be split in three main parts: prevention, detection and correction. IDSs deal with detection part.

In [4], Peddisetty sums up definitions of [5] and [6]. He describes intrusion detection as the process of monitoring events and analyzing them for sign of intrusions or attempts to compromise the confidentiality, integrity, availability or to bypass the security mechanisms of a computer or network.

Peddisetty classifies IDSs according to their location, detection method and response. Following sections present location and detection method categories, which are interesting for our study.

1) Location: The most common way to classify IDSs is to group them by location. Network-based IDSs (NIDS) operate intrusion detection directly on the network; they detect attacks by capturing and analyzing network packets. Host-based IDSs (HIDS) operate on information collected from within an individual computer system. They analyze system logs and critical system files to detect intrusion. Finally, Distributed-IDSs (DIDS) [5][6] use several IDSs (whatever the location) to correlate events from different places of the network.

2) Detection methods: Core engines in detecting malicious activities are detection methods. They function automatically, analyze the information they monitor and raise alarms whenever they detect intrusion. Peddisetty sums up detection methods in [4]. Most used techniques are Pattern Matching and Anomaly-Based detection.

Pattern Matching (or Signature-Based Detection) consists in scanning information and looking for known patterns into it. Whenever the IDS has found a similarity, it raises an alarm. Signature-Based Detection can detect known intrusions, so the main weakness of this method is the need of a constant update of the database containing known patterns (also called signatures).

Anomaly-Based Detection adopts a simple approach : ignore everything that is normal and raise an alarm if it deviates from the normality. This kind of detection method can be effective in detecting unknown attacks, but it may also generates a huge amount of false alarms.

When an attacker behaves like a normal user, Heuristic-Based Detection can be used. This method uses algorithmic logic to detect intrusion attempts using statistical evaluations of the traffic.

C. Large-scale coordinated attacks

In this paper, we focus on coordinated attacks, such as distributed scans, worm outbreaks and distributed denial-of-service attacks. Such attacks are very difficult to detect, according to Zhou *et al.* [7], because IDSs are only monitoring a limited portion of the network. Zhou *et al.* refer to large-scale coordinated attacks as attacks that target a large number of hosts which are spread over a wide geographical area.

In order to detect these types of large-scale coordinated attacks, they need the ability to combine the evidence of suspicious network activity from multiple, geographically distributed networks. To do that, they introduce CIDS, Collaborative Intrusion Detection System. This type of IDS involves several IDSs that collaborate through the network, to identify threats.

In this paper, we experiment large-scale coordinated attacks on NIDS to confirm the need of CIDS. The next section describes distributed portscan, a coordinated attack used as a reconnaissance phase.

III. Distributed Portscan

We chose to experiment large-scale coordinated attacks using distributed portscan. Fyodor, original contributor of the audit tool Nmap, considers in [8] that port scanning lets attackers discover exploitable communication channels. In this section, we describe what is a portscan and how to distribute it. Moreover, we explain why we chose that kind of attacks to confirm whether or not security systems prevent coordinated attacks.

A. Portscan

In this section, we define what is a portscan. Then, we introduce some variables we used for our experiments and how an IDS can detect this kind of attack. Eventually, we describe how an attacker can evade an IDS when portscanning a network.

1) Definition: M. Roesch, who wrote Snort IDS Manual [9], describes portscan as a first phase in network attack, a reconnaissance phase. In that phase, an attacker determines what type of network protocols or services a host supports. Portscan is also used by administrators to evaluate the security of their networks.

Scanning a port consists in an exchange of network packets ; depending on the reaction of the remote host, one can conclude whether or not a port is open, closed or filtered.

2) Variables of a portscan: when an attacker wants to perform a portscan on a network, he can adjust some variables. These variables let the attacker be more discrete or know specific information about remote scanned hosts.

a) Techniques: depending on the goals of the attacker, he would use a specific technique rather than another. Most of portscan techniques give information about state of the targeted ports whereas other techniques give information about the service or the operating system.

These techniques are explained in [8], the manual of Nmap. For example, TCP Connect scanning consists in establishing a TCP connection. This is done by exchanging several packets between the source and the destination. After an attacker has initiated a TCP connection, he knows whether or not the port is open, following the answer the remote host gives him.

Another example is the FIN scanning technique. This one use a weakness of the specifications of the TCP protocol. Indeed, the RFC 793 [10] reads : *"If the destination port state is closed [...] an incoming segment not containing a RST causes a RST to be sent in response."* As a result, when an attacker does not receive a response from a remote host after sending this type of packet, he knows that the port is open.

b) Timing: an attacker can adjust the speed rate of portscan attacks. By doing this, he can easily evade IDSs, because most of them do not detect portscans when they are executed very slowly.

c) Targeted ports: Lee describes different types of targeted port in [11]. Her paper introduces vertical, horizontal and block scans, presented below.

Vertical scans are portscan that target numerous ports on a singular remote host. That type of portscan tries to discover a weakness on a particular host. Contrary to vertical scans, horizontal scans target only one port on several remote hosts. This lets an attacker search a specific weakness on a given network. Horizontal scans are commonly used by worms, that are aware of a particular vulnerability. A block scan is a combination of vertical and horizontal scans ; it consists in a portscan of several ports on several remote hosts.

d) Parallelization and distribution: another variable of portscans is the parallelization. An attacker can divide portscans into sub-tasks that scan only a part of the set. A portscan using local parallelization has more chance to be detected quickly by IDSs because only one host generates malicious traffic. That's why distributed portscan is a good idea to evade IDS. Distributed portscans are described in the Section III-B.

3) Portscan detection: one of the features of IDSs is to detect portscans. This section describes how IDSs succeed to do this.

Firstly, we have to differenciate portscan that mimic legit traffic and portscan that use weaknesses of protocols. The latter is very easy to detect when such weaknesses are described and known by IDSs. For example, FIN scanning (or the alternative Null scanning) generates unexpected traffic. An IDS keeping connection state can detect such malicious traffic. Snort stores a huge set of signatures, that describes such attacks.

Detecting a portscan that generates legit traffic is very difficult. IDSs use counters to attribute a score to each host that tries to address an host of the internal network. These counters are incremented each time an event occurs. When a counter reaches a fixed threshold, an alert is raised.

Counters need to be reinitialized or decreased ; Kang *et al.* introduce in [12] some amnesty policies. The first one, Positive-Reward-based method, consists in decreasing the counter whenever the host is acting normally. A normal event could be a successful connection or a connection to a highly visited host. The second one, Timeout-based method, decreases counters when related events expire.

4) IDS evasion: the main goal of an attacker is to be unnoticed so that he can perform an attack after. To remain discreet, he has to evade IDSs.

There are several ways to evade IDSs. First, an attacker has to use portscan techniques that adopt a legit behaviour. For IDSs using threshold detection technique, an attacker can avoid Positive-Reward-based method by connecting regularly to known open ports. To evade Timeout-based method, an attacker just has to slow down the portscan attack.

Another way to evade IDS is to split the portscan through several hosts. This is the topic of the next section.

B. Distributed Portscan

The easiest and fastest way to scan while evading IDS is to distribute attacks. IDSs are most of the time defenseless when several hosts, also called scanners, are leading an attack.

We chose to experiment distributed portscan because portscan is a phase where attackers are trying to acquire networks knowledge. If IDSs can stop them at this phase, following attacks would not be that efficient. Moreover, distributed portscan is one of the most common large-scale coordinated attacks.

This section describes basic methods to distribute an attack in order to portscan some hosts of a network. We only describe two distribution techniques because we implemented them for the experimentations. We suppose, for these experimentations, that attackers know when hosts achieving attacks are detected. Actually, this is not true for real attacks. This assumption let us compute rates introduced in Section V.

1) Naive distribution: Kang *et al.* describe in [12] the naive distribution. It consists in a sequential distributed scan: the attacker selects one of the scanners he controls and starts scanning the target network with it. When the scanner is detected, the attacker selects a different scanner and resume the portscan. The process continues until the portscan is completed. We expect this type of distributed portscan to be linear: if a

scanner can scan x ports, two scanners should be capable to scan the double, $2x$.

2) Parallel Distribution: This distribution consists in splitting the whole set containing targets and ports between scanners. Each scanner has a sub-task to perform, then he communicates the results to a coordinator. We expect this technique to overrule IDSs detection due to the generated traffic.

IV. Experimentations

This section presents the experimentations that have been made to confirm that actual security solutions are not efficient against large-scale coordinated attacks. First, we introduce the different tested solutions then we explain how the solutions were tested.

A. Security solutions

We experimented large-scale coordinated attacks on two security solutions : Snort and a commercial firewall. We chose to experiment these attacks on both a commercial and a non-commercial solution, in order to compare devices coming from different environments.

1) Snort: This open-source IDS, according to [9], is able to analyze network traffic in real-time. Snort (version 2.9.1) uses a set of signatures (written by the community) that identify known attacks. In our case, this type of detection (signature-based) can only target portscans that are known and not legit. Snort uses also several modules, including sfPortscan that does statistical analysis and detects portscan that could seem legit.

2) Commercial Firewall: This product is also called a United Threat Management. According to its specifications, it includes a network firewall and an IDS. As it is a commercialized product, we do not know the mechanisms used by the manufacturer to detect attacks and intrusions. However, experimentations should give us an idea of how this device works.

B. Experimental protocol

Experimentations have a common process that we present in this section along with the network architecture and benchmark configuration.

1) Network architecture: the network architecture used for experimentations is pictured in Figure 1. The security solution, at the center, is the gateway between cloud computing hosts (on the left) and internet hosts (on the right). All traffic goes through the security solution.

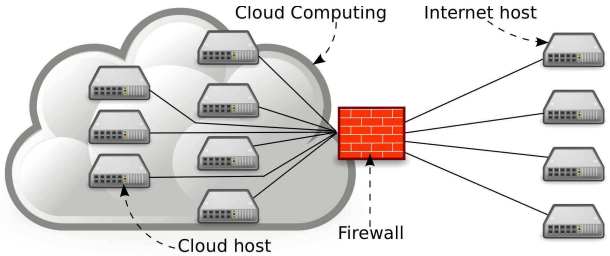


Fig. 1: Network architecture

All hosts (from the cloud or the internet) were virtualized within OpenVZ[2] containers. Snort was also virtualized whereas the commercial firewall was a physical device.

2) Benchmark configurations: experimentations we made have been accomplished with different configurations. A configuration is characterized by several variables, such as the number of attacking (or targeted) hosts, the way to scan or the way to distribute the portscan. Below, we list the different values of these variables :

- number of attacking hosts : 2^n, with $1 \leq n \leq 6$,
- number of targeted hosts (Snort) : 2^n, with $1 \leq n \leq 6$,
- number of targeted hosts (Commercial) : 2^n, with $n = 2$,
- distribution portscan technique : naive and parallel,
- portscan technique : Connect, SYN, RPC, FIN, Null, Xmas,
- local portscan timing : insane, aggressive, normal and polite,
- targeted ports : 100 ports (most used ports),
- traffic generated : only by the attacks,
- security solution configuration : default.

Timing values are Nmap default options ; for example, insane timing does a parallelized portscan every 5ms while polite timing does a sequential portscan every 0.4s. We chose not to tweak the configuration, so that a security solution is not favoured over the other one.

3) Benchmark process: Each benchmark was an automated sequence of steps. These steps include turning on the security device, processing the distributed attack, verifying whether or not there is an intrusion according to the security solution, logging information about the current attack, turning off the security device.

The reason to turn on and off the device at the beginning and at the end of experimentations is to avoid interferences from the previous and next benchmark.

V. Results

A. Evaluation

After each experiment, we knew how many ports the attacker succeeded to scan. To compare experiments, we introduce a metric named Attacker Success Rate (ASR), computed as follows:

$n =$ Number of ports successfully scanned before detection
$T =$ Total number of ports to scan

$$ASR = \frac{n}{T}$$

We consider that a portscan is successful when it is not detected. Also, the scanner needs to find the correct port state and the generated traffic must reach targets (according to security policies, firewalls may drop packets).

This rate, that fluctuates between 0 and 1, can also be represented as a percentage. The lower is the rate, the better is the security solution. Indeed, a low rate means that the security solution has detected almost all portscan attempts.

[2]OpenVZ is a container-based virtualization for Linux: http://wiki.openvz.org.

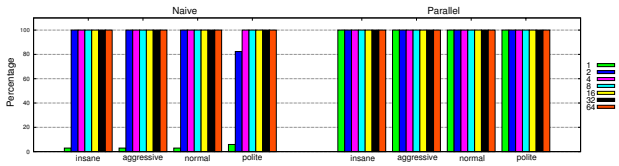
Each configuration has been executed at least 25 times. To avoid false-positives (that may be caused by hardware malfunctioning, network disconnection, etc.), we creamed off results by removing extreme values. Then, we calculated median value for each configurations, that are displayed in the following tables and diagrams.

B. Results

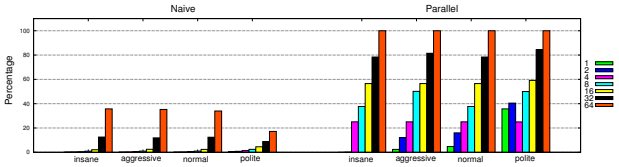
In this section, we only highlight some particularity of the results. The whole set of results and scripts used for these experimentations are available[3] on request.

We present results using diagrams. A diagram represents the ASR on the y-axis (as a percentage) for a given number of scanners and a given portscan technique. Also, a diagram pictures distribution techniques (naive to the left, parallel to the right). The number of scanners is represented with different colors ; they are grouped together according to the timing of the portscan.

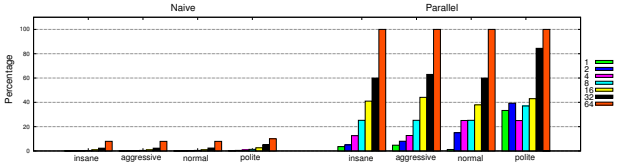
Figure 2 refers to experiments on Snort for the TCP Connect technique. We observe that for 32 (Figure 2b) and 64 targets (Figure 2c), naive distribution is nearly linear. The case where there is only one target (Figure 2a) is specific: Snort does not detect anything because there isn't enough traffic to raise alarms.



(a) 1 target



(b) 32 targets



(c) 64 targets

Fig. 2: Snort – ASR for the TCP Connect portscan technique according to number of targeted hosts

Also, we observe that the more scanners there are, the faster the attack is detected. Indeed, for 32 scanners, for the polite timing, we have these rates, according to the distribution technique presented in Table I.

[3]Address an email to authors or go to the website : www.lifl.fr/∼riquetd

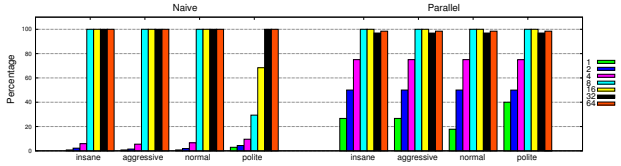
Number of targets	ASR Naive	ASR Parallel
1	100%	100%
8	41%	96%
32	9%	84%
64	5%	84%

TABLE I: Snort – TCP Connect technique with 32 scanners

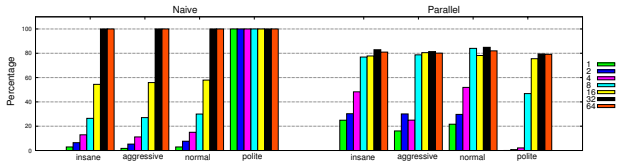
Besides, we notice that the best way to evade IDS detection is to use distribution rather than slowing the attack. For example, for 8 targets and parallel distribution, success rate is respectively 10% and 37% for insane and polite timing. On the contrary, when we vary the number of scanners, success rate is respectively 10% and 95% for 1 and 64 scanners.

Figure 3 presents distributed TCP Connect Portscan targeting 4 hosts. As soon as there are at least 8 scanners, the commercial firewall is better. Indeed, Snort is completely overrunned, with an ASR about nearly 100% for each cases. This is not true for the polite timing, where the commercial firewall seems to reach his limits. Distributed attacks are slow enough to bypass detection system.

The commercial firewall has better results when there are few scanners but becomes inoperative when attacks are slow enough.



(a) Snort



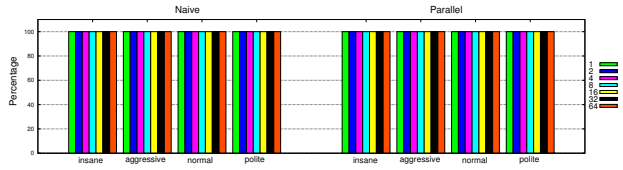
(b) Commercial firewall

Fig. 3: ASR for the TCP Connect portscan targeting 4 hosts

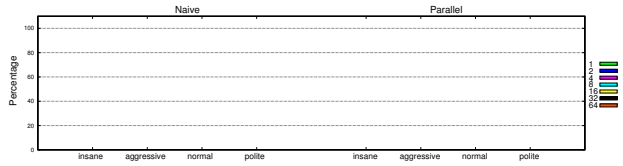
Figure 4 presents results for the Null scanning technique, which use a illegit behaviour. This kind of technique can be identified and described as a signature. Surprisingly, default configuration of Snort does not include this signature. Snort does not detect Null portscan attacks, whatever the experimental configuration is.

About the commercial firewall, the device integrates a TCP automaton ; this let the device drops any packet that is not in a right state. In this case, traffic generated by a scanner never reaches targets, because it does not generate a correct exchange of data. We encountered similar results with FIN and Xmas scan techniques (illegit portscan method). In these cases, portscan (or distributed portscan) is never considered successful (this implies that ASR is always equal to 0). Indeed, when a firewall drops that type of packet, portscan usually

concludes unsuccessfully port states based on firewall answers. Figure 4b and Figure 5b present these results.



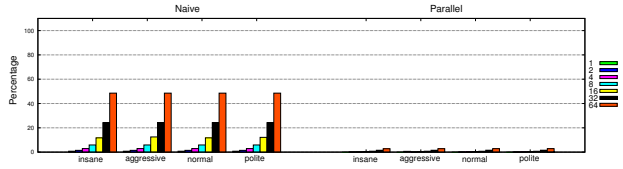
(a) Snort



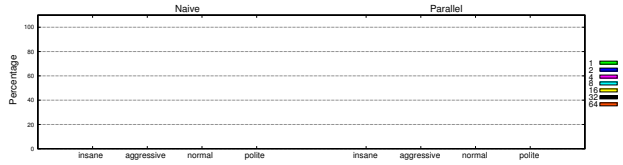
(b) Commercial firewall

Fig. 4: ASR for the Null portscan targeting 4 hosts

Other scan techniques like Xmas [8] scanning, based on a illegit behaviour, are included by default in the signature set of Snort IDS. Snort is very efficient in this case, according to figure 5.



(a) Snort



(b) Commercial firewall

Fig. 5: ASR for the Xmas portscan targeting 4 hosts

To conclude about experimentations, there is always a way to successfully evade detection systems. In our experiments, for most configurations, 32 scanners were enough to evade them. According to results, large-scale coordinated attacks easily evade IDS and firewalls.

VI. Conclusion

Cloud computing is a resourceful architecture. It can provide several layers of services according to the needs. Currently, only security issues delay its massive adoption. In this paper, we studied cloud computing security, more precisely the impact of large-scale coordinated attacks.

Experimentations led in this paper show that distributed attacks can easily be achieved without being detected. Indeed, either the security solution can be obsolete because not updated, or the solution can rely on unsuitable methods. Be that as it may, an attacker who controls enough hosts can accomplish distributed attacks while evading security solutions. In our case, the attacker needs 32 hosts to complete a distributed portscan to avoid detection.

Our experiments only focus on distributed portscans, so other experimentations about different distributed attacks should be led. Also, we point out the fact that no traffic noise was included during our experimentations, so results might be worse in real attacks.

Our future work will focus on the detection of large-scale coordinate attacks using collaborative IDS (CIDS) using reconfigurable devices. These IDS could be scattered across the network as physical or virtual probes.

References

[1] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Fifth International Joint Conference on INC, IMS and IDC, 2009. NCM '09*. IEEE, Aug. 2009, pp. 44–51.

[2] S. M. Bellovin and W. R. Cheswick, "Network firewalls," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 50–57, Sep. 1994.

[3] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, no. 8, pp. 805–822, Apr. 1999.

[4] N. R. Peddisetty, *State-of-the-art Intrusion Detection: Technology, Challenges, and Evaluation*, 2005.

[5] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C.-l. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, D. M. Teal, and D. Mansur, "DIDS (Distributed intrusion detection system) - motivation, architecture, and an early prototype," *in proceedings of the 14th national computer security conference*, pp. 167—176, 1991.

[6] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni, "An architecture for intrusion detection using autonomous agents," in *Computer Security Applications Conference, 1998, Proceedings., 14th Annual*. IEEE, Dec. 1998, pp. 13–24.

[7] C. V. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," *Computers & Security*, vol. 29, no. 1, pp. 124–140, Feb. 2010.

[8] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. USA: Insecure, 2009.

[9] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX conference on System administration*, ser. LISA '99. Berkeley, CA, USA: USENIX Association, 1999, p. 229–238. [Online]. Available: http://dl.acm.org/citation.cfm?id=1039834.1039864

[10] I. S. Institute, "RFC 793 (TCP)," www.ietf.org/rfc/rfc793.txt, 1981. [Online]. Available: www.ietf.org/rfc/rfc793.txt

[11] C. B. Lee, C. Roedel, and E. Silenok, "Detection and characterization of port scan attacks," 2003. [Online]. Available: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.161.7079

[12] M. G. Kang, J. Caballero, and D. Song, "Distributed evasive scan techniques and countermeasures," in *Proceedings of the 4th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, ser. DIMVA '07. Berlin, Heidelberg: Springer-Verlag, 2007, p. 157–174.