Honeypots for Distributed Denial of Service Attacks

Nathalie Weiler
Computer Engineering and Networks Laboratory (TIK),
Swiss Federal Institute of Technology ETH Zürich, Switzerland
weiler@tik.ee.ethz.ch

Abstract

Distributed Denial-of-Service attacks are still a big threat to the Internet. Several proposals for coping with the attacks have been made in the recent past, but neither of them are successful on themselves alone. In this paper, we present a system that helps in the defence in depth of a network from DDoS attacks. In addition to state-of-art active and passive security defences, we propose a honeypot for such attacks. The goal is to convincingly simulate the success of the compromise of a system to a potential DDoS attacker. Thereby, we can implement the lessons learned by the honeypot in our other systems to harden them against such attacks. On the other hand, we protect the rest of our network infrastructure form the impact of such an attack.

Keywords: *Distributed Denial of Service Attacks, Honeypot, Security Practices.*

1 Introduction

Since February 2000, even laymen know what a devasting effect a Denial of Service (DoS) can have: Ebay, Amazon and Buy.com were out of business for hours due to new generation of DoS attacks, the so called Distributed Denial of Service (DDoS) attack [8]. Denial of Service Attacks are as old as the Internet.

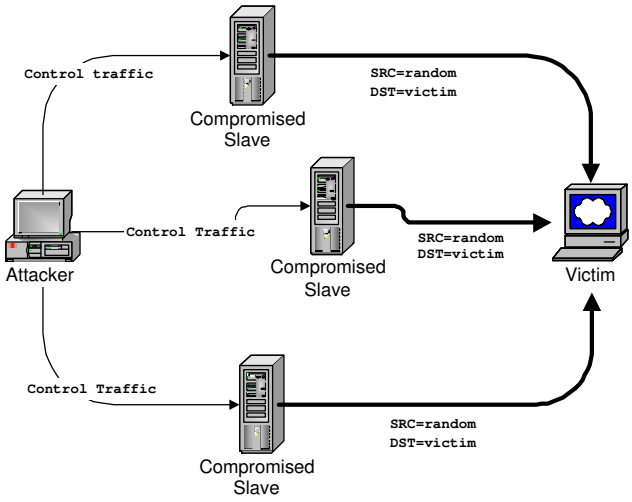


Figure 1. Overview of an DDoS Attack Scenario.

In fact, the first link in the ARPAnet, the mother of the Internet, crashed while receiving data from the sender, the fault being a bug in the communication software [14].

The danger of becoming a victim in a DoS or DDoS attack increases with the availability of attacking tools for downloads in the World Wide Web. The threats to services and users in the Internet becomes stronger. The yearly survey of Computer Crime and Survey of 2001 names denial of service attacks as one of the four major attacks seen in 2001. This type of attack has seen a major increase from 24% of companies reporting such incidents in 1998 to 36% in 2001 [13]. A simple experiment confirmed this trend: we set up a misconfigured system with several unfixed security holes, and announced is presence by trivial postings from the root account in some newbie Newsgroups for system administrators.The log of our intrusion detection system reported 25 different presumed DDoS preparing attacks on this system vs. 1 the day before.

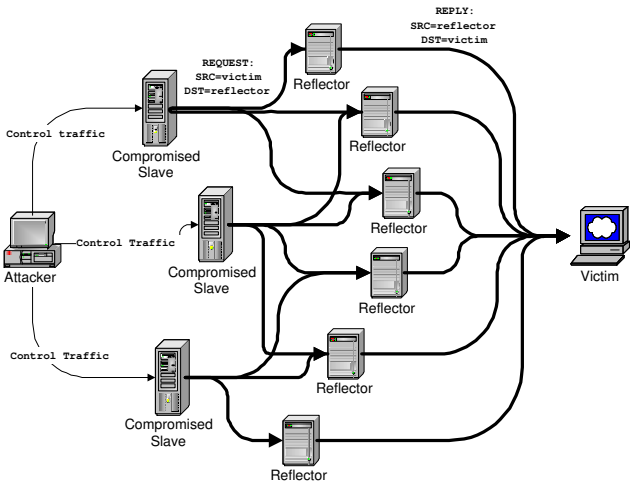


Figure 2. DDoS Attack using reflectors.

In a typical DDoS attack, the attacker subverts a number of servers on the Internet by exploiting well-known security flaws. These compromised servers become the slaves of the attacker by the installation of flooding tools for the real attack. Figure 1 illustrates the procedure of the attack. The attacker sends control traffic to his compromised slaves

that instructs them to generate high volume traffic toward the victim, typically with a faked source address to prevent backtracing to the slaves (that might be used in another attack).

In this paper, we present the design of a system that allows to disrupt the attacker's chain. The goal of the system is to convince the attacker that he successfully compromised the slave. In reality, the system is a kind of a honeypot that lures him to believe so. Thereby, the operator of the honeypot learns the tactics of the attacker and can implement efficient defences in the rest of his network. On the other hand, the attack on the victim is of course successfully inhibited and the recording of the compromise may help in a legal action against the attacker.

The remaining of the paper is organised as follows: First, Section 2 characterises DDoS attacks. Actual methods of defences are discussed in Section 3. Section 4 details the design of our system. We conclude with a short evaluation and further work in Section 5.

2 Characteristics of Distributed Denial of Service Attacks

2.1 Definitions of DoS and DDoS

A **denial of service (DoS) attack** is commonly characterised as an incident in which a user or organisation is deprived of the services of a resource they would normally expect to have. Typically, the loss of service is the inability of a particular network service, such as e-mail, to be available or the temporary loss of all network connectivity and services. In the worst case, for example, a Web site accessed by millions of people can occasionally be forced to temporarily cease operation. A denial of service attack can also destroy programming and files in a computer system [3, 2].

This attack works well if the attacker and the target are equally well equipped in bandwidth and in computing resources. Distributed DoS are used in order to magnify the effect on the victim. Thereby, the attacker can for instance successfully flood a high-end web server consisting of a cluster of web servers served by a powerful load balancer. The WWW Security FAQ identifies such attacks as one of the most dangerous because of their impact on web servers. [19] defines: A **Distributed Denial of Service (DDoS) attack** uses many computers to launch a coordinated DoS attack against one or more targets. Using client/server technology, the perpetrator is able to multiply the effectiveness of the Denial of Service significantly by harnessing the resources of multiple unwitting accomplice computers which serve as attack platforms. Typically a DDoS master program is installed on one computer using a stolen account. The master program, at a designated time, then communicates to any number of "agent" programs, installed on computers anywhere on the Internet. The agents, when they receive the command, initiate the attack. Using client/server technology, the master program can initiate hundreds or even thousands of agent programs within seconds.

A further improvement on the impact of the attack is achieved by using a second type of intermediate system called a reflector [12]. The usage of such reflectors is illustrated in Figure 2: The compromised servers send spoofed requests to the reflectors. The reflectors then reply to the seen source of the request: the victim of the DDoS attack

2.2 Characterisation

Both types of attacks have in common that they typically use a limited number of well know attacks sometimes in different combinations. A DoS attack's main characteristics is that an attacker attempts to prevent one or more legitimate users of a service from the use of the required resources. Therefore, he attempts (1) to inhibit legitimate network traffic by flooding the network with useless traffic. (2) to deny access to a service by disrupting connections between two parties, (3) to block the access of a particular individual to a service, or (4) to disrupt the specific system or service itself.

DDoS attacks follow the same path, but they become more effective and difficult to prevent because of the intermediary systems that add a many-to-one dimension to the whole attack. In the following we will explain by examples the different characteristics of common DoS and DDoS attacks.

Software bugs are frequently used by attackers to compromise a system. One of the first seen DoS attacks, while not widely spread, relied on a software bug in the IOS/700 software release version 4.1. It allowed to crash some Cisco 7xx routers by connecting with telnet and typing very long password strings. By exploiting this software bug, attackers could reboot the 7xx routers and deny service to legitimate users during the reboot period [5].

Another example of a well exploited software bug is the "myriad escaped characters vulnerability" in the Microsoft Internet Information Server[1]. [11] defines it as a denial of service vulnerability allowing a malicious user to overload a web server by a request for a file via a specially-malformed URL for some period of time. The vulnerability does not cause the server to fail, or cause any data to be lost, and the server eventually would resume normal operation, given enough time. Microsoft's IIS implements the decoding of escaped characters[2] in URL strings very inefficiently. So that's why the server was overloaded with the processing of long strings with a large number of escaped characters. Microsoft does of course provide a patch for this vulnerability on its technical support web page.

A Smurf DoS uses the Internet Control Message Protocol (ICMP) that handles errors and exchange control messages [10]. It sends an ICMP_ECHO_REQUEST packet with a spoofed source address (the IP address of the victim) to one or several subnet broadcast addresses. All machines on the subnet reply directly to the victim's address. The normal effect this attack causes is congestion either in the victim's network connection or in the access ISP network. The

[1] Better known as the URL parsing bug.

[2] Escaped characters are way for users to specify non-printing or special characters in URLs. For example, if a percent sign is followed by two hexadecimal digits, they are replaced by the equivalent ASCII character.

most effective countermeasure against this attack is to disable broadcast pings from outside a subnet. Many people believe that a subnet that still allows such pings have a faulty configuration. Therefore, Smurf like attacks belong to the class of attacks based on misconfigurations.

Another popular attack misuses the standard three-way handshake of the Transport Control Protocol (TCP) [4]. This handshake requires an exchange of three messages between client and server before the service can be used: (1) The client indicates that it wants to start a connection with the server by sending a synchronise (SYN) request. (2) The server replies with a message indicating its readiness: a SYN/ACK (ACKnowledgment) reply. (3) The connection can be used after the final ACK of the client. The so called SYN flood attacks the server – the victim in this case – by sending a large amount of SYN requests without fulfilling the third step of the handshake. Typically, the client's address is also spoofed. Because today's TCP/IP implementations only handle a limited number of connections, the server will discard new connections as long as its backlog queue is full with semi-open connections. This attack is an example of a protocol attack. Analogue attacks exist for other network protocols such as UDP and ICMP.

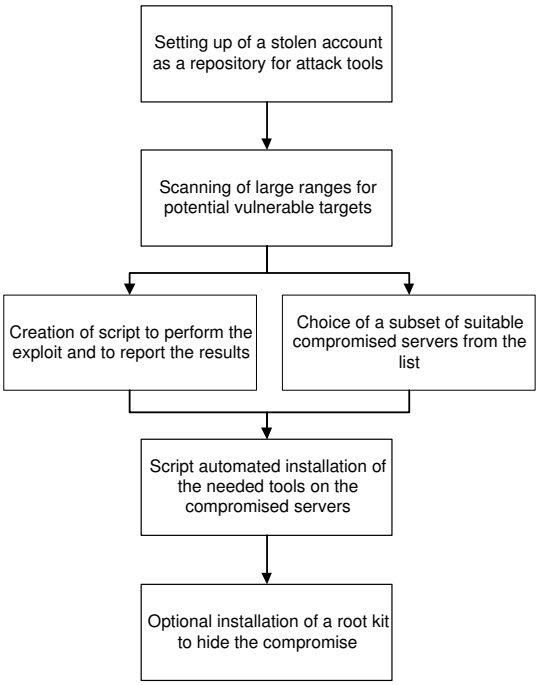


Figure 3. DDoS Pattern.

DDoS attacks follow the simple pattern illustrated in Figure 3. We list the variations on this pattern in chronological order on an example basis.

Trinoo was the first widely known DDoS tool. It uses TCP to exchange control data between the attacker and the master attack host. The compromised slaves are controlled though UDP messages. These then operate an UDP flooding attack on the victim.

Tribe Flood Network (TFN) uses command line interfaces (telnet or ssh) to deliver the control messages to the slaves. Communication between the slave and the attack daemon is done by ICMP echo reply packets because these typically pass through firewalls and are harder to detect than UDP packets. In addition to Trinoo's UDP flooding attack, TFN supports TCP SYN and ICMP floods as well as Smurf attacks.

Stacheldraht is a variant of TFN that uses encrypted TCP connections for control traffic. Furthermore, attack daemons may be updated automatically.

TFN2K further enhances TFN: communications are encrypted to make it more difficult to detect the attack. Slaves and attack daemons communicate through ICMP, UDP or TCP selected randomly. .

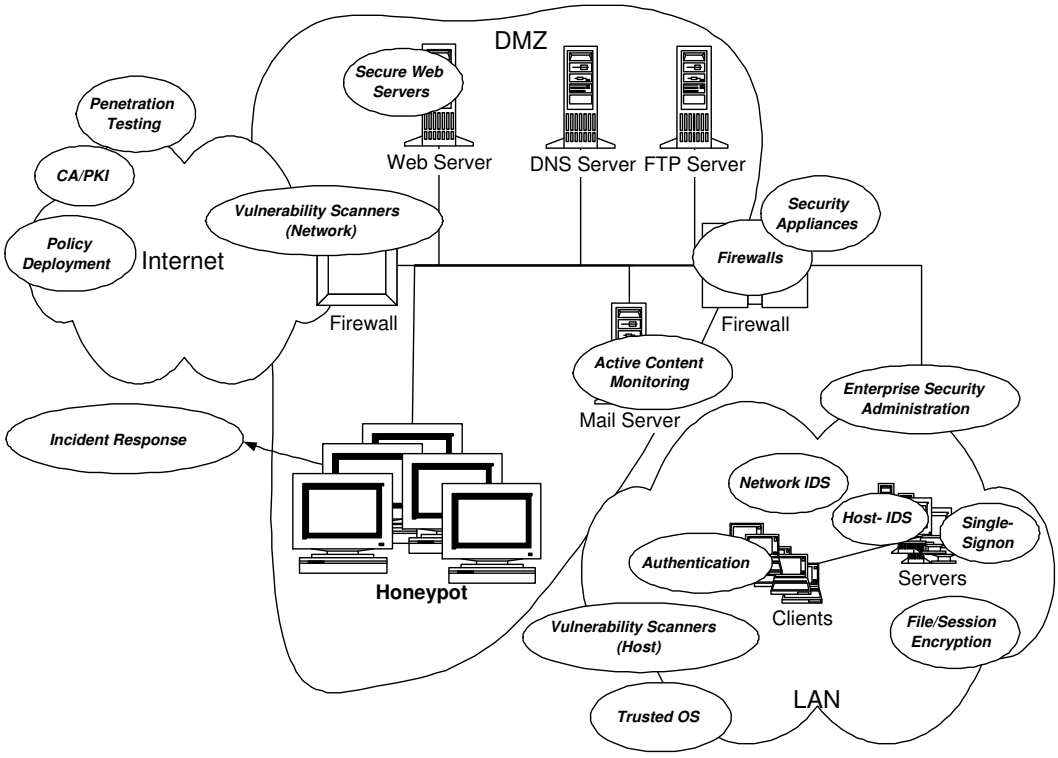
3 Defenses against Attacks

Several proposals have been made to cope with DDoS attacks even though neither of them solves the issue completely. Broadly the approaches can be categorised into two broad categories: mitigation of the impact/detection the attack and identification of the source of the attack. The first category includes measures as (a) filtering packets [6], (b) disabling broadcasts and unused services, and (c) applying security patches [7]. The second area of proposals focuses on identifying the source of the DDoS attack. This problem of tracing back of such packets of data received considerable attention in the past: Bellovin's ITRACE [1] uses ICMP packets to verify the path of a small subset of selected forwarded packets. So, the victim may be able to locate the compromised slave. [15] use a packet marking scheme to enable the victim to traceback the real source of the packet. [18] enhance the scheme by reducing the number of markings by employing network topology maps. Finally, [16] propose a source path isolation engine in strategical located routers in order to enable victims to request the taken path of a given packet. However, all these mechanisms suppose that a large enough amount of networks implements them. How realistic this assumption is for the future cannot be fully answered currently.

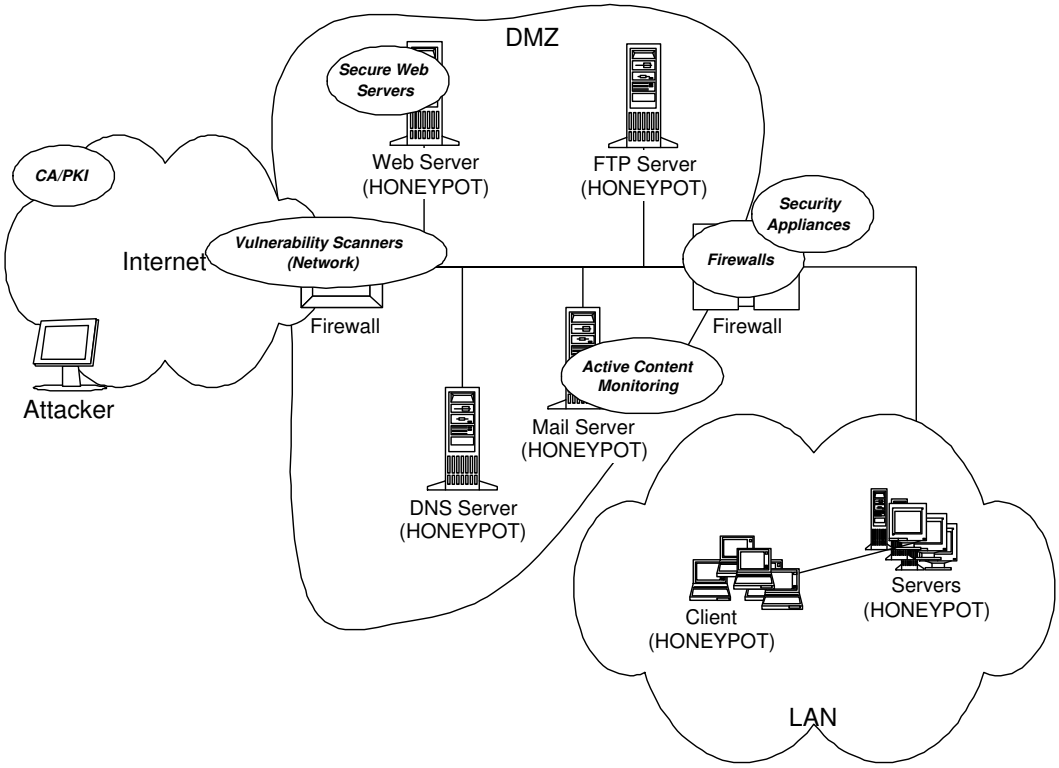
4 A Honeypot for DDoS

In our analysis of existing research in the coping with DDoS, we looked for a generally applicable solution. The response to an incident should be independent of platform as the attack occurs across many different platforms within a network and should not assume the changes in a large part of the backbone network. Therefore, we propose a system that can be generally applied in each organisation and relies on state-of-the-art technology. The vantages of this system are two-fold: First we can defend our operational network with a high probability against known DDoS and against new, future variants. Second, we trap the attacker so that recording of the compromise can help in a legal action against the attacker.

The devised system is a honeypot that lures the attacker to believe that he successfully compromised a slave for his needs. In reality, the honeypot learns the tools, tactics, and



(a) Implementation in the Organisation.



(b) View of the Attacker.

Figure 4. Two Different Views of the Honeypot.

motives of the blackhat. The lessons learned are then implemented in the rest of the network as defensive mechanisms.

The honeypot provides an organisation information on their own security risks and vulnerabilities. It should consist of similar systems and applications than the one used by the organisation for its productive environment so to give the attacker a real world feeling and to be able to implement the learned lessons in the productive environment.

Figure 4(a) illustrates the implementation in the organisation. The depicted organisation runs a well set up and maintained security infrastructure with classical elements and recent developments: Services such as web, mail, ftp services and DNS that should be accessible form the outside are situated in a demilitarised zone (DMZ). The local internal network (LAN) of the organisation is in another zone protected by a firewall with adequate, up-to-date security appliances; even inside the LAN file transmission are always encrypted, the clients run trusted operating systems, the services are managed by a indirect authentication method[3]. Furthermore, detection systems are running: host based intrusion detection systems (IDS) and vulnerability scanners ,and network IDS together with network vulnerability scanners at the borders of the organisation's network. The organisation might operate virtual private networks (VPN) with local subsidiaries and a public key infrastructure (PKI) for intra-business corporation. Standard mechanisms are used for protection of web and mail servers.

In this security infrastructure, we introduce a new system: a honeypot that should attract distributed denial-of-service attackers. This virtual system physically corresponds to a set of computing system or a network of such systems following the idea of the Honeynet project[4] [9]. The Honeynet is a conceptually upgrading of traditional honeypots used for intrusion detection. According to the definition,

"*a **Honeynet** is different from traditional honeypots, it is what we would categorise as a research honeypot. This does not make it a better solution then traditional honeypots, merely it has a different purpose. Instead of its value being detecting or deceiving attackers, its value is gaining information on threats. The two biggest design differences from a classic honeypot are: (1) It is not a single system but a network of multiple systems. (2) All systems placed within the Honeynet are standard production systems. These are real systems and applications, the same you find on the Internet. Nothing is emulated nor is anything done to make the systems less secure. The risks and vulnerabilities discovered within a Honeynet are the same that exist in many organisations today.*"

Our DDoS honeypot must fulfil the task to lure the attacker into employing this system as a compromised slave. That's why the attacker's packet – regardless of protocol – should be handled by the honeypot while all other – regular – packets are forwarded to the legitimate destination (web server, mail server, client, e.g.). So, the honeypot should simulate the whole network of the organisation to the attacker as shown in Figure 4(b). Every system in the organisation might be a honeypot. For example, if the attacker's compromise packets to the webserver of the corporation are detected, the packets go to the honeypot for processing. The reply the attacker gets should be indistinguishable from a real reply of the web server.

Three major problems must be solved to successfully project this illusion to the attacker:

1. The attack must be detectable.
2. The attack packets must be actively directed to the honeypot.
3. The honeypot must be able to simulate the organisation's network infrastructure, at least the parts known to the attacker.

The first issue is linked to the solution of the second problem: both should ideally be implemented by a transparent packet forwarder at the border of the corporation's DMZ. Its functionality is to look at each packet and decide if the packet belongs to a DDoS attack. If the test is negative, the packet should go to the given destination inside the DMZ or the LAN. In all other cases the packet forwarder should determine which part of the honeypot system should produce the request. A possible setup of the honeypot could be that each Internet service of the corporation is replicated in one system of the honeypot. First experiments showed that the forwarder is a potential bottleneck in this setup. Therefore, we investigate currently different other setups with next generation routers for this issue.

The detection itself is done by efficiently matching signatures of DDoS packets. Currently, we employ similar signatures as the DDoS signatures of [17], to be able to detect know attacks with a large probability. However, this method has the drawback, although being very efficient,that new attacks are not detected until our system knows the signature.

Finally, the third problem can be solved by employing a variant of the Honeynet approach. Then, it should also be easier to simulate realistic confirmation messages to the attacker as depicted in Figure 5. The depicted warning system of the honeypots to the reflectors and the victim enables to play down eventual probes of the attacker to verify the success of the DDoS attack at these points.

5 Conclusion

In this paper, we described a promising tool for luring attackers into the belief of a successful DDoS attack. We showed how such a system can be used in a defence in depth real-world network environment. We identified different problems with the current realisation and provided first solutions to cope with the scalability of the honeypot. Although our honeypot is still in its' infancy, we achieved first promising results with the presented initial setup. Future work will consist of the development of a honeynet

[3]Kerberos is an example of such a method. An off-line authentication method (i.e. based on certificates) should be used if the management of the clients becomes too burdening and if the organisation requires larger corporation environments than one single LAN.

[4]The Honeynet itself cannot be directly applied to this situation, because it does not deceive the attacker about the real network structure. However our DDoS honeypot needs this illusion as shown below.

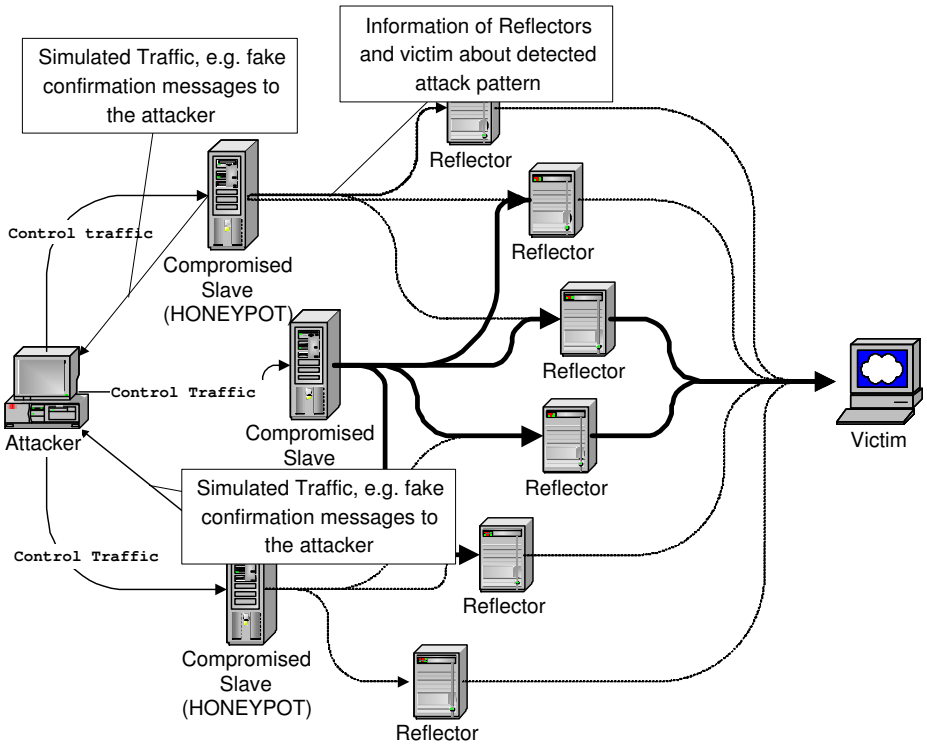


Figure 5. Tracing of the Attacker.

scalable to a middle sized organisation and the investigation of other solution to the re-direction of packets.

References

[1] S. Bellovin. ICMP Traceback Messages. `http://www.research.att.com/~smb/papers/draft-bellovin-itrace-00.txt`, March 2000.

[2] CERT Cordination Center. CA-1999-17: Denial-of-Service Tools. `http://www.cert.org/advisories/CA-1999-1.html`.

[3] CERT Cordination Center. CA-2000-01: Denial-of-Service Developments. `http://www.cert.org/advisories/CA-2000-01.html`.

[4] CERT Cordination Center. CA-2000-21: Denial-of-Service Vulnerabilities in TCP/IP Stacks. `http://www.cert.org/advisories/CA-2000-21.html`.

[5] Cisco Systems, Inc. Cisco Security Advisory: 7xx Router Password Buffer Overflow. `http://www.cisco.com/warp/public/770/pwbuf-pub.shtml`, June 1998.

[6] Cisco Systems, Inc. Defining Strategies to Protect against TCP SYN Denial of Service Attacks. `http://www.cisco.com/warp/public/707/4.html`, July 1999.

[7] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial-of-service Attacks which Employ IP Source Address Spoofing. RFC 2267, January 1998.

[8] A. Harrison. The Denial-of-service Attack Aftermath. `http://www.cnn.com/2000/TECH/computing/02/14/dos.aftermath.idg`, 2000.

[9] HoneyNet Project. `http://project.honeynet.org/`.

[10] C. A. Huegen. The latest in Denial-of-Service Attacks: "Smurfing" Description and Information to Minimize Effects. White Paper, February 2000.

[11] Microsoft Technical Support. Myriad Escaped Characters Vulnerability. `http://www.microsoft.com/technet/security/bulletin/fq00-023.asp`, April 2000.

[12] V. Paxson. An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks. *Computer Communication Review*, 31(3), July 2001.

[13] R. Power. 2001 CSI/FBI Computer Crime and Security Survey. Technical report, Computer Security Institute, 2001.

[14] P. H. Salus. *Cating the Net: From ARPANET to INTERNET and Beyond*. Addison-Wesley, 1995.

[15] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. In *Proceedings of ACM SIGCOMM 2000*, August 2000.

[16] A. Snoeren, C. Partridge, L. Sanchez, W. Strayer, C. Jones, and F. Tchakountio. Hash-Based IP Traceback. In *Proceedings of ACM SIGCOMM 2001*, August 2001.

[17] Snort Signature Database. `http://www.snort.org/snort-db`.

[18] D. Song and A. Perrig. Advanced and Authenticated Marking Schemes for IP Traceback. In *Proceedings of ACM INFOCOM 2001*, April 2001.

[19] L. D. Stein and J. N. Stewart. The World Wide Web Security FAQ – Version 3.1.2. `http://www.w3.org/Security/Faq/`, February 2002.