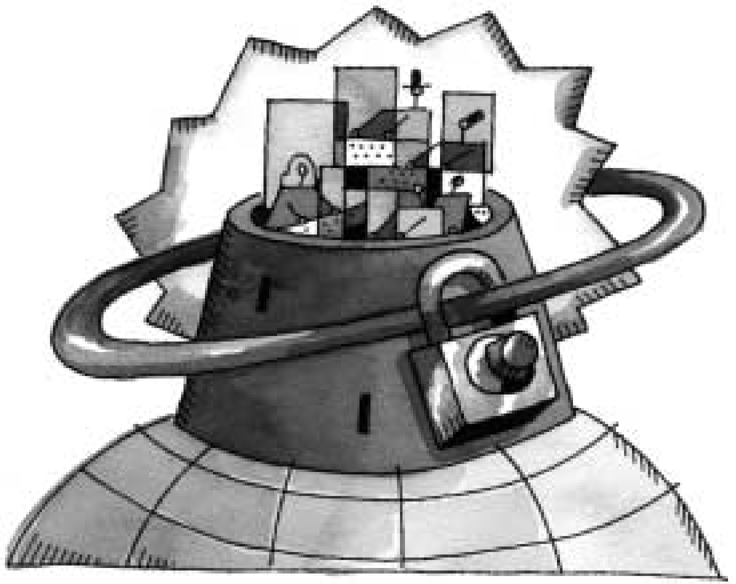
;login:

THE MAGAZINE OF USENIX & SAGE
December 2002 • volume 27 • number 6

Focus Issue: Security

Guest Editor: Rik Farrow



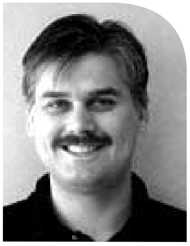
inside:

Dietrich: Active Network Defense

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

active network defense

by Sven Dietrich



Sven Dietrich is a Member of the Technical Staff at the Carnegie Mellon Software Engineering Institute in Pittsburgh, PA. He is the project manager for the Active Network Defense project, and focuses his research on security and survivability.

spock@cert.org

Some Concepts and Techniques

We define Active Network Defense (AND) as the defensive side of computer network operations. The defensive side contains active and passive aspects, and we will focus on the active techniques available for defensively engaging the attacker.

The threats that AND is attempting to address include:

- Denial-of-service (DoS) attacks, including distributed DoS attacks, which cause unusable or crashed systems, clogged networks, or untimely responses for critical missions.
- Worms, recognized as a threat to network security when the Morris worm spread in 1988. Worms such as Code Red and Nimda continue to pose a real threat through their propagation and their embedded mission.
- Viruses which attach themselves to electronic mail messages and local documents.
- Malicious code, including various attack techniques not covered above, such as penetration via exploits, external information gathering such as scanning, etc.

In order to address the threats, one must consider different approaches. The goals set forth are to limit access for the attacker by reducing the attacker's ability to do damage by changing:

- the network path from the attacker to the victim
- what the attacker can see and/or do.

Active Deception Techniques

Typical deceptions include concealment of information, false and planted information, and camouflage to mislead the attacker with respect to characteristics or contents of a host or network. The attacker's strength is focused on an interesting object, such as a host, a set of hosts, or an entire network of attractive hosts. Fred Cohen's Deception ToolKit[1] is a prime example of this honeypot capability. More recently, highly interacting networks of honeypots have been used in the Honeynet Project[2] to discover presence, tools, tactics, and the intent of the attacker. By actively feeding the attacker more attractive systems in comparison to the systems to be defended, the attacker is encouraged to spend time and energy compromising, exploiting, and contacting these honeypots. Since honeypots are not production machines, should a connection from a honeypot be detected it would indicate the presence of the attacker. A recent development is the concept of a virtual honeypot (honeyd).[3] In this approach, a single host creates virtual hosts on a network, with given characteristics of a chosen operating system, in order to deceive the attacker, including a given topology of virtual hosts. Here is a sample configuration script for honeyd:

```
annotate "AIX 4.0 - 4.2" fragment old
# Example of a simple host template and its binding
create template
set template personality "AIX 4.0 - 4.2"
add template tcp port 80 "sh scripts/web.sh"
add template tcp port 22 "sh scripts/test.sh $ipsrc $dport"
add template tcp port 23 proxy 10.23.1.2:23
set template default tcp action reset

bind 10.21.19.102 template
```

[1] Fred Cohen, Deception ToolKit: *http://www.all.net/dtk/index.html*.

[2] The Honeynet Project: *http://project.honeynet.org/*.

[3] Niels Provos et al., virtual honeypots: *http://www.citi.umich.edu/u/provos//honeyd/*.

The config script simulates an AIX host with the old fragment reassembly policy (to fool scanning tools such as nmap),[4] scripts to handle probes/connects to ports 80 (HTTP) and 22 (SSH), and the capability to proxy port 23 (telnet) connections to another host. Any other TCP scanning attempts will encounter a TCP reset. Even though honeyd is a work-in-progress, it can be downloaded and is usable today.

Related to the idea of obfuscation outlined before, there is another deception technique, known as packet scrubbing,[5] which can hide and falsify host, operating system, or other characteristic information for those systems behind the packet scrubber. An attacker looking to fingerprint a target host or network will be faced with false and possibly inconsistent information. Consider a network of mixed hosts, say PCs and Macintoshes. One can make the network look like all PCs, all Macintoshes, all Solaris boxes, or like nothing in particular. The emphasis, however, is on "normalizing" the traffic so that it is indistinguishable from any other operating system, rather than pretending to be a different type of operating system. One approach is already built into the pf packet filter,[6] included in OpenBSD, for example. A simple configuration line added to your pf.conf file will "scrub" your inbound traffic in an effort to thwart exploitation of ambiguities in TCP/IP protocol stacks to perform fingerprinting attacks or worse:

```
ext_if = "kue0"

# normalize all incoming traffic
scrub in on $ext_if all
```

Attacks that are worse include exploiting the IP fragment reassembly techniques of intrusion detection systems for overlapping IP fragments. Using pf with packet scrubbing enabled on a NAT (Network Address Translation) box or firewall will protect hosts on the closed side. OpenBSD itself is already immune to such attacks.

Hogwash takes a more proactive approach. Using snort-like configuration files, Hogwash is built on top of layer 2, also known as the data link layer, and is designed to run on Linux systems without IP networking installed, so as to be completely invisible on the network. The defense philosophy of Hogwash is centered on the theory that a low-level network approach will prevent the packet scrubber from becoming the target of the next attack. Its focus is to drop or sanitize malicious packets only. All other packets travel completely unmodified through the network, since the system does not directly interact with the packet at the protocol level (e.g., Ethernet hardware addresses or time-to-live fields do not get changed or updated). Packets fall into three categories:

- Legitimate or good packets are let into the network.
- Malicious or bad packets are dropped by the system, and an alert is sent to the operator.
- Transient or, sometimes, bad packets are left unaffected, but an alert is still sent to the operator.

For example:

```
drop tcp $EXTERNAL_NET any -> $HOME_NET 80 (content:"/etc/passwd";
msg:"WEB: attemp to request /etc/passwd";)
```

This drops any requests originating on the external network and directed at the Web server on the home network that contain the string /etc/passwd, potentially an attempt to retrieve the UNIX password file. The Hogwash project is still experimental, but its author claims that a Celeron 733-equipped host with two 100Mbps Ethernet

[4] Fyodor, nmap: *http://www.insecure.org/nmap/index.html*

[5] Matthew Smart, Robert Malan, and Farnam Jahanian, "Defeating TCP/IP Stack Fingerprinting," *9th USENIX Security Symposium*, 2000: *http://www.usenix.org/publications/library/proceedings/sec2000/smart.html.*

[6] Daniel Hartmeier, "Design and Performance of the OpenBSD Stateful Packet Filter (pf)," *2002 USENIX Technical Conference*, June 2002: *http://www.benzedrine.cx/pf.html.*

cards can handle the full 100Mbps network, depending on the rule set. Your mileage will vary.

Preemptive Strike

As a preventive measure, network defenders can actively wander the networks in search of potential attackers. Once a potential attacker has been identified, by whatever means, the network defenders can collect intelligence on the capabilities of the attackers. Such intelligence gathering can include host and network characteristics – for example, operating system versions, infrastructure architecture details, and router operating systems. By exploiting this knowledge, the defenders could make a preemptive strike against the attacker, taking advantage of existing vulnerabilities in the remote hosts. This can take several forms. If a potential set of vulnerable hosts is identified and it is known they are not (yet) under the control of the attacker, then a series of "mass patchings" can remove vulnerabilities in the remote machines. If, on the contrary, the hosts have already been compromised, then some cleaning code can be injected to remove the malicious code and possibly patch the system. This was demonstrated in the response to Code Red: a benign version of the original Code Red called Code Green[7] removed Code Red from an attacking system and patched it to resist further Code Red infections. The development of Code Green was assisted by the findings[8] of the Eeye Digital Security team, and by various other "experiments" with the Code Red worm. It is, of course, possible to turn the hostile host against its controller, effectively turning the attacker's agents against the attacker's host(s).

Rather than attacking the enemy's hosts themselves, it is also possible to target the routing infrastructure, resulting in a collapse of the attacker's connectivity. Cooperative ISPs can install access control lists and/or rate-limiting to prevent attack traffic from entering the Internet core. Lacking ISP cooperation, DDoS or crafted BGP attacks against routers can cripple the connectivity of attacking hosts. BGP has been the object of analysis for its instability,[9] but seriously: "Kids, don't try this at home."

Striking Back at the Attacker

In a different scenario, such as during an ongoing attack, network defenders could explore the possibility of retaliating against the attacking hosts. While this is a legally and ethically problematic subject,[10] let us explore what the possibilities are today.

The first option would be to disable the attacking machines, if/when they have been properly identified. Suppose such traceback or other identification has taken place, then fault inducement in the attacker's code, the underlying environment, and/or the operating system will stop the attack, or at least a portion thereof. By exploiting known vulnerabilities causing kernel panics in the remote host, such as the Ping of Death[11] or teardrop,[12] the attack would stop, temporarily at least. By gaining system privileges on the remote host, one can modify, clean, or spoil the system for the attacker for extended periods of time.

The second option would be to attack the immediate surroundings of the attacker. If for some reason the attacking hosts are impenetrable, disabling the nodes providing connectivity to the attacking hosts would make the attack stop. This can be achieved by crashing routers or causing the local routing infrastructure to collapse.

A third option would be to develop code that leverages strategic knowledge of the attacker's intentions and techniques to thwart the attack. Again using the Code Red incident as an example, the re-addressing of the whitehouse.gov servers which allowed

[7] Herbert HexXer, Code Green: a copy is available at *http://archives.neohapsis.com/archives/vuln-dev/2001-q3/0575.html.*

[8] Eeye Digital Security. ".ida 'Code Red' Worm": *http://eeye.com/html/Research/Advisories/AL20010717.html.*

[9] Rik Farrow, "Routing Instability on the Internet," Network Magazine, March 2002: *http://www.networkmagazine.com/article/NMG20020304S0007/2.*

[10] Nathan Buchheit, Anthony Ruocco, and Donald Welch, "Strike Back: Offensive Actions in Information Warfare," New Security Paradigms Workshop, 1999.

[11] CERT, Advisory CA 1996-26: *http://www.cert.org/advisories/CA-1996-26.html.*

[12] Microsoft Corporation, "Stop 0a in tcpip.sys When Receiving Out-of-Band (OOB) Data": *http://support.microsoft.com/support/kb/articles/Q143/4/78.asp.*

the network defenders to sidestep Code Red's date-triggered DDoS attack and the development of Code Green, were made possible by the analysis and scrutiny of Code Red's behavior produced by many cooperating analysts and reverse engineers. The recognition of the attacker's mission enabled response options otherwise not considered.

Dynamic Infrastructure Modification and Traceback

The aftermath of both the University of Minnesota (1999) and the February 2000 attacks generated a series of DDoS traceback and mitigation schemes. The following is a list of a few representative examples based on their predominant role, end host vs. infrastructure.

Schemes Within the End Hosts

Both Bellovin[13] and Savage et al.[14] show how information fed back to the attacked hosts facilitates attack path reconstruction. Since the audit messages are sent probabilistically (typically one every 20,000 packets), the victim, having accumulated enough of these, can trace the real path back to the attacker, who is attempting to evade tracking by spoofing its source address. Song and Perrig[15] improve on Savage's work, providing more efficiency and scalability. Both schemes incorporate messages into unused fields of the IP packets, effectively marking them. All of the schemes involving the marking of small numbers of packets are problematic for a very widespread attack using small numbers of packets from a very large number of machines.

Snoeren et al.[16] propose a passive monitoring scheme for assisting with the traceback of a single packet. Packet digests are kept for a limited amount of time and permit the traceback across the *traceable* infrastructure up to the edge of this tracing infrastructure. The scheme provides a low false positive rate, which diminishes as the packet moves closer to the source of the offending packet. The impact is minimal, since the mechanism attaches to the network passively, but it can be implemented on the router. Note that the schemes of Bellovin and Snoeren require that additional traffic be delivered to the host under attack in order to determine the attacking location. This is problematic under conditions where links or routers are saturated, but it is conceivable to perform this via an out-of-band mechanism.

Schemes Within the Infrastructure

All the schemes in this category suffer from various degrees of the same shortcoming: a substantial latency involved in recognizing an attack and implementing a countermeasure. Short-lived, or one-packet, attacks cannot easily be handled with these techniques. In addition, it may be the case that attacks that occur in bursts with durations shorter than the countermeasures' recognition and reconfiguration period will largely evade the countermeasures. Similarly, it is not clear that the approaches are viable if an attack comes from a very large number of well-dispersed sources. Note that any of the traceback techniques of the previous section could be used to guide these countermeasures. Stone's CenterTrack[17] uses an overlay network of routers that allows for monitoring and rerouting of suspicious traffic. Bellovin and Ioannidis implement pushback,[18] a router-based mechanism that treats DDoS as a congestion-control problem and drops the traffic causing the congestion. Sterne et al. propose an active network approach in their autonomic response to DDoS attacks.[19] Malicious attacks are countered by sending mobile code upstream, which analyzes traffic flows on each router and duplicates itself at split points until the source of the offending stream is narrowed down or identified. Papadopoulos et al. investigate the coordinated

[13] Steve Bellovin, Marcus Leech, and Tom Taylor, "ICMP Traceback Messages," IETF work-in-progress, October 2001.

[14] Stefan Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," in ACM SIGCOMM 2000.

[15] Dawn Song and Adrian Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," IEEE Infocom 2001.

[16] Alex Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayer, "Single-Packet IP Traceback," *IEEE/ACM Transactions on Networking* 2002, forthcoming.

[17] Robert Stone, "CenterTrack: An IP Overlay Network for Tracking DoS Floods," 9th USENIX Security Symposium, 2000: *http://www.usenix.org/publications/library/proceedings/sec2000/stone.html*.

[18] J. Ioannidis and Steve Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks," Network and Distributed Systems Security Symposium, February 2002.

[19] Dan Sterne, Kelly Djahandari, Ravindra Balupari, William La Cholter, Bill Babson, Brett Wilson, Priya Narasimhan, Andrew Purtell, Dan Schnackenberg, and Scott Linden, "Active Network-Based DDoS Defense," 2002 DARPA Active Networks Conference and Exposition (DANCE 2002), May 29–31, 2002.

approach to dealing with DDoS in their COSSACK scheme.[20] The correlation between the attacking hosts, i.e., the simultaneous presence of similar packets, reveals the presence of an attack in this snort-based tool. D-WARD[21] looks at the validity of TCP connections, such as completed three-way handshakes, for allowing or disallowing packets through routers. By establishing traffic models, potentially offensive packets are kept at bay via throttling at the router level.

Most of this work assumes that the attacks and their sources have been correctly identified before performing what amounts to a denial of service on itself. The risk remains that due to a partial compromise of the system a denial of service is easily triggered, effectively finishing the task intended by the attacker.

On a more practical note, UUNET has developed an interesting technique for dealing with traceback of an attack flood, called "backscatter traceback."[22] In this technique, a clever combination of BGP configuration and triggers can quickly lead to the entry point into the infrastructure of spoofed IP addresses. During an attack, the offensive traffic is redirected to a null interface on the border routers. The resulting flurry of ICMP unreachable messages is sent back to both legitimate and spoofed sources, and a large portion of these messages destined for non-routable addresses (a large chunk, say 96.0.0.0/3) are redirected to a so-called sink hole network. Since the source address of these ICMP messages is one or more routers, which represent the entry points into the infrastructure, the source of the attack can easily be traced and quenched, often within one or two minutes.

A Comment in Closing

While some of the techniques in AND (the preferred term is now Computer Network Defense Response Actions or CND RA) remain controversial, it provides fertile ground for research as countermeasures are challenged and circumvented in a continuous cat-and-mouse game.

ACKNOWLEDGEMENTS

I would like to thank John McHugh, Howard Lipson, Eric Hayes, and other colleagues at CERT for their contributions and comments, and Tom Longstaff for making this article possible.

[20] Christos Papadopoulos, Ramesh Govindan, Bob Lindell, and John Mehringer, "Coordinated Suppression of Simultaneous Attacks (COSSACK)," December 2001: *http://www.isi.edu/cossack/*.

[21] Jelena Mirkovic, Peter Reiher, and Gregory Prier, "A Source Router Approach to DDoS Defense," Technical Report 010042, UCLA Computer Science Department, 2001.

[22] Brian W. Gemberling, Christopher L. Morrow, and Barry R. Greene, "ISP Security: Real-World Techniques," October 2001: *http://www.nanog.org/mth-0110/greene.html*.