Trapping and Tracking Hackers:
Collective Security for Survival in the Internet Age

Douglas B. Moran
Vice President, Research & Development
Recourse Technologies, Inc
2450 El Camino Real, Suite 100, Palo Alto, CA 94306
http://www.recourse.com/

Introduction

Why are Internet attacks so difficult to deal with? A significant factor is that the *coverage* of the current set of computer security tools is a poor match to the threat, analogous to a fence with multiple gaps. Most current tools fall into two broad categories:

> Prevention: configuration checking, identification-and-authentication, other encryption-based approaches, ...

> Limited response to individual instances of known vulnerabilities and attacks: firewalls, intrusion detection systems, ...

Problems arise because these tools are too often expected to provide broader coverage than their designs permit. The limitations of "prevention" tools are that (1) there is typically a large gap in time between the identification of the vulnerability and the availability of the solution, and (2) partial and faulty deployment resulting in gaps in coverage. Tools for response to individual attacks typically have significant false negative rates (*e.g.,* tunneling through firewalls) or high false positive rates (intrusion detection systems). With these tools, there is also a gap between the onset of new types of attack and when signatures/rules become available to handle them.

"I don't need to outrun the bear, I just need to outrun you." This punchline reflects the philosophy inherent in many sites' security measures. Increasing the difficulty of breaking into your site can be effective in persuading *some* attackers to shift their efforts to some other site. Notice that this is most effective against attackers who would constitute a mere annoyance, for example, script kiddies who are looking for sites to break into, without much regard for which ones. This approach is least effective against the more serious threats: attackers specifically targeted on your site and insiders.

This defensive approach blocks a wide range of known attacks, but gives you little leverage for preventing future attacks. Military doctrine has long recognized the futility of a purely defensive strategy—the aggressor, given the twin advantages of initiative and time, will inevitably find an exploitable weak spot. However, the military analogy's remedy of offensive actions must be adapted for the Internet because simple offensive actions are typically impractical, immoral and/or illegal [1]. Instead, the remedy is to deprive the attacker of the initiative and the time to stage

further attackers. To achieve this, computer security tools must better leverage information about attacks and attackers to better protect the larger cluster, the enterprise, and the larger Internet.

Handling new attacks is a major critical gap. Our approach to this is based on the philosophy of *time-based security* [2] and it has two components:

> *Intelligence gathering:* collecting information about the attacker, his intentions, and his capabilities
>
> *Delaying action:* This has multiple purposes:

- Slowing down the attack can limit the damage done. Many attacks indicate an awareness by the perpetrator that risk of detection rises disproportionately with time: they are heavily automated and increasingly segmented into separate sessions. Causing the attacker to waste his time on low-value targets reduces or eliminates the time available for attacking high-value targets.

- Delaying the onset of the attack against the high-value targets allows the defender to use that time to strengthen the defenses of those systems and to use the intelligence gathered during the early stages of the attack to prioritize the measures taken.

- Allowing time to perform additional intelligence gathering, especially tracking the attacker back through a chain of relay/cutout hosts, enhances the breadth and depth of the response.

A secondary benefit of this approach is that it not only helps fill the gap in the metaphorical fence, but makes the gap narrower: the improved intelligence gathering allows other security tools (*e.g.,* configuration checkers, IDSes) to be upgraded to handle new attacks earlier.

Currently, collecting information about attacks has multiple impediments:

> The rate of detection of attacks is low [3].
>
> The rate of reporting detected attacks is low [4].
>
> The quality of the data collected is typically poor [5]. The combination of automated attacks, delayed detection and delayed response almost always means that the attacker has plenty of opportunity to cleanup after himself. Additional evidence is lost to normal operations of the computer and its legitimate users.
>
> The quality of the reports of the data collected tends to be poor [6]. Even when expertise is available, the cost of collecting the data and assembling it into a report exceeds the available resources of the invariably overloaded system administrators.
>
> Substantial delays between the reporting of an attack and the receipt of countermeasures destroys any sense of urgency or importance.

The cost and delays to sanitize the data in a report to be submitted to an outside agency is often prohibitive.

Reporting that you have been successfully hacked can be not only embarrassing, but may involve penalties (*e.g.,*loss of investor confidence).

Deception Servers and Deception Hosts

There are some existing technologies to help fill this gap, but they are little used:

- Deception servers [7, 8, 9, 10, 11, 12]

- Deception hosts (a subset of the ill-defined term *"honeypots"*) [13, 14, 15, 16]

Deception servers emulate a range of network servers (*e.g.,* Telnet, FTP, SMTP), often for a range of platforms, but do not provide actual access to the underlying host. Most allow the operator to provide a dummy version of *some* likely targets (*e.g.,* the password file). This is very useful for discovering details of new attacks in which the opening gambit is the key move, but because they cut off access immediately after that, they give negligible information about the attacker's intentions. And with the increasingly rapid dissemination of new exploits from elite hackers to the script kiddies, these tools often fail to provide information about the skill level of attackers (important when you want to focus your efforts on the more dangerous ones).

Deception hosts have a long history of use in computer intrusions, but most have been *ad hoc*. They typically have been created and deployed in the midst of a serious incident. Long-term use of a deception host presents challenges not seen during an ongoing attack. ManTrap^{TM} is a system for creating and managing a collection of deception hosts that reside permanently on a network.

ManTrap^{TM} Key Features

Monitoring: The typical attacker bolts at the slightest sign that he is being monitored, so the monitoring and management of the deception host must be virtually undetectable. *Ad hoc* deception hosts are typically monitored with packet sniffers, which work well if the attack is conducted primarily with interactive commands. However, for automated attacks, a packet sniffer does not show what is happening on the system—the operator is forced to analyze the attack scripts and binaries, or, more dangerously, run them on a comparable system and monitor the effects. For many networks, packet sniffing is an unacceptable long-term monitoring tool because data captured presents both storage and confidentiality problems.

With ManTrap, the attacker is actually running inside a *cage* that contains a virtual environment that he can explore and change. The monitoring processes run outside the cage, both hiding and protecting them from the attacker. Multiple monitoring processes are running, providing the operator with multiple views of the attackers activities (*e.g.,* packet logging, process invocation).

These logs facilitate creating reports on an attack. Additional tools for summarizing the log contents are planned. ManTrap has minimal operator costs: it has a negligible false positive rate, and the operator sets a threshold for being notified.

Containment: The fundamental concept of a deception host is that you *want* the attacker to have apparent full access to the host so that you can study him. However, this brings with it the danger of him using the deception host as a base to attack other hosts, both inside and outside your network. During an ongoing attack, monitoring the deception host can be handled manually—it represents a small addition to the monitoring that typically is already being done. And since you are actively monitoring events, you are able to respond quickly. With ManTrap, containment is provided by a combination of the cage and its placement in the network, with multiple alternatives available. Different choices provide trade-offs between setup costs, the realism of the network environment, and how quickly the operator needs to begin active monitoring of an attack.

Convincing content: Not only must a deception host not scare off the attacker, it must induce him to stay connected *and* continue to perform actions that provide useful information to you. *Ad hoc* deception hosts typically use real (live) systems that have already been compromised, allowing some additional controlled compromises for the opportunity to prevent uncontrolled compromises.

This is not acceptable for most deployments of permanent deception hosts. ManTrap includes a Content Generation Module that probabilistically selects and instantiates templates using a combination of generic and locally-specified values. This generated data is not intended to withstand careful examination–the attacker is unlikely to expend valuable connect time on this. Instead, this data is intended to provide a convincing volume of false hits for the types of searches that attackers often perform. This fabricated content is typically supplemented with non-confidential real content, for example, the Web pages from the public Web server.

Because the contents of a ManTrap has been "pre-sanitized," this removes one of the substantial impediment to reporting (above). Also, since an attack captured by a ManTrap represents a success of your security measures, reporting an attack has lost the stigma of failure, thereby eliminating another of the impediments.

Variability of content: As ManTrap becomes more widely deployed, attackers will attempt to detect whether they have connected to one. To make detection more difficult, ManTrap probabilistically introduces variability, both in the configuration of the platform and in the text generated from the templates. Variations in the text include modifying line breaks and introducing common typos and misspelling.

Faithful representation of the platform: ManTrap does not *emulate* a platform, but instead implements the cage with a small set of modifications to the kernel. With this approach, the attacker sees the expected capabilities and idiosyncrasies of the target platform. The approach works well for an operating system that has unified APIs for the kernel (*e.g.,* Solaris), but for OSes that have multiple APIs to the same kernel functionality (*e.g.,* Linux), the work required to provide a *consistent* deception is greatly increased.

Resetting the trap: It is common for a serious hacker to *"own"* hundreds of hosts at one time, and thus it is not uncommon for significant time to elapse between the initial compromise of a host and when it is used to stage attacks on other hosts. An effective deception host needs to leave the attacker's downloads on the system for use in later visits. However, if the deception host is being used only for uninteresting activity, such as a "Warez" server (for pirated software), the operator will usually want to erase all changes (by resetting the deception host) and wait for a more interesting (threatening) attack. With ManTrap, the contents of the cage is generated from data stored outside the cage, and thus is trivially regenerated.

Deployment Schemes

Minefield: Currently, the most common scheme for deploying ManTrap systems is to place them among likely high-value targets. One site is deploying one ManTrap for every three of its Web servers. Legitimate requests are distributed among only the real Web servers, but someone attempting to gain access to the Web servers via non-*httpd* means has a 25% chance of selecting a ManTrap system as his *first* target, and since the typical attacker will attempt to "own" all the Web servers in the cluster, he will quickly wind up in a ManTrap, revealing both his presence and his actions.

Shield: Attacks against high-value targets are actively redirected into the ManTrap by a firewall or router. For example, an attempt to connect to the public Web server through an un-permitted service (*e.g.,* SMTP) is automatically redirected (via Network Address Translation) to a ManTrap whose contents mirror the Web server but that sits in a separate DMZ (for containment).

Zoo: This is an all-deception-host network, with the deception hosts configured with different applications just as you would expect for the normal mix of servers and workstations, and with different mixes of vulnerabilities. The purpose of such networks is to encourage the attacker to exercise more of his "bag of tricks" and to reveal his skill level and intentions, thereby allowing efforts to be focused on the more serious threats. Although such networks are under consideration, we are currently not aware of any operational deployments.

ManHunt™

A common practice of attackers is to use a series of intervening hosts (relays/cutouts) to hide their true location. Current trackback methods, especially across administrative boundaries (such as ISPs and enterprises), are too slow: the trail too often has gone cold after only one or two hops. ManHunt uses the existing infrastructure to provide rapid trackback across network segments and administrative boundaries [18].

Interlocking Fields of Fire

Current security tools are designed as independent *bastions* and this philosophy leads to them being deployed as multiple lines of defense. Incorporating the concept of *interlocking fields of fire* into such tools can provide a substantial boost in the effectiveness of the overall collection. Although this concept provides for redundant coverage, its most important attribute is that an attacker comes under fire from several different angles: an effective attack on one defender can leave him highly exposed to others.

For example, it is inevitable that attackers will discover ways to identify a ManTrap system. This potential problem can be turned into an advantage by including a suitable IDS. The anticipated tests for ManTrap all produce strong signatures, creating a dilemma for the attacker: If he tests for ManTrap but is on a normal system, he has announced his presence to any IDS aware of those signatures. Thus the mere presence of ManTrap, or even suspected presence, can provide enhanced protection for other hosts on the network. However, since ManTrap is just beginning to appear in operational environments, it has not yet become a factor in attackers' plans. It will be interesting to see how attackers react, and to see if individual sites can influence (to their advantage) the attacker's decision of whether to test for ManTrap.

Notes and References

[1] Example: Attackers rarely stage a Denial-of-Service (DoS) attack from their own site, instead launching it from another compromised site. Thus, a site that is DoS'ing your site is likely an unwitting victim of the real perpetrator.

[2] Time Based Security, Winn Schwartau. Interpact Press, 1999.

[3] In the DISA Vulnerability Analysis and Assessment Program (VAAP), the detection rate was roughly 5% [2-pg iii, 17].

[4] In the VAAP, the reporting rate during the first half was about 5% (or 0.25% of successful attacks) [2-pg iii], but later rose to 27% [17], almost certainly as a consequence of actions taken to remedy the low reporting rate. Hence, the 5% rate of reporting detected attacks is probably the better estimate for generic organizations.

[5] Personal communications from staff of various incident response teams and from law enforcement officials.

[6] The representative of a major computer security company (ISS) classified their customers thusly: 60% had expertise in neither their OS nor computer security, 20% had expertise in both, and the remaining 20% had expertise in OSs but not computer security. Comment made during a panel session at Recent Advances in Intrusion Detection (RAID) Workshop, Purdue U., 1999.

[7] "There Be Dragons," Steven Bellovin. In Proceedings of the Third USENIX UNIX Security Symposium, Baltimore, MD, September 1992. ftp://ftp.cerias.purdue.edu/pub/doc/true_stories/Steven_Bellovin_dragon.ps.Z

[8] "An Evening with Berferd In Which a Cracker is Lured, Endured, and Studied," Bill Cheswick, 1991. ftp://ftp.cerias.purdue.edu/pub/doc/true_stories/Bill_Cheswick_berferd.ps.Z

[9] Deception ToolKit (DTK) from Fred Cohen & Associates. http://all.net/dtk/dtk.html

[10] CyberCop Sting from Network Associates. http://www.pgp.com/asp_set/products/tns/ccsting_intro.asp

[11] SPECTER Intrusion Detection System. http://www.specter.ch/

[12] NetFacade from GTE/BBN. http://www.itsecure.bbn.com/NetFacade.htm

[13] "Stalking the Wily Hacker," Clifford Stoll. In Communications of the ACM, 31(5):484, May 1988.

[14] The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage, Clifford Stoll. Doubleday, 1989.

[15] "To Build a Honeypot," Lance Spitzer, 2000 June 7. http://www.enteract.com/~lspitz/honeypot.html

[16] "Know Your Enemy: Motives: The Motives and Psychology of the Black-hat Community," The Honeynet Project, 2000 June 27. http://www.enteract.com/~lspitz/motives/

[17] Information Security: Computer Attacks at Department of Defense Pose Increasing Risks, General Accounting Office, U.S. Congress, Chapter Report, 05/22/1996, GAO/AIMD-96-84, section 2:1 Number of Attacks Increasing. http://www.gao.gov/AIndexFY96/abstracts/ai96084.htm

[18] ManHunt is scheduled to be available in 4Q'2000.