A Note on Distributed Coordinated Attacks

by Dr. Frederick B. Cohen ‡

Abstract

In this paper, we describe a new class of highly distributed coordinated attacks and methods used for tracking down their sources.

Search terms: Information Protection, Auditing, Trojan Horses, Distributed Computation, Coordinated Attacks, Distributed Coordinated Attacks

‡ This research was funded by Management Analytics, PO Box 1480, Hudson, OH 44236, USA

1 Background

Late in 1995, an article relating to Internet [1] [2] World Wide Web security [3] claimed that a Web server could contain code that directed a browser reading that code to attack other sites and that this sort of attack would bypass current firewall technologies. [4] In late February of 1995, an on-line report from the University of California at Berkeley [5] showed that a Universal Resource Locator (URL) [6] [7] could be used to launch such an attack. [8] This attack is particularly interesting because it is essentially a Trojan horse forced on the innocent Web user whose computer automatically runs the attacking program (e.g., the program *gopher* [9] in this example attacks the SMTP [10] port) with inputs provided by the attacker.

On or about 00:45 EST (GMT+5) on March 13, 1996, someone at a site in California chose to combine this attack mechanism with the automatic loading of background patterns provided by many modern Web browsers to cause users at thousands of sites across the Internet to automatically, without the users' knowledge or consent, and in the background, attempt to *telnet* [11] into our site (*all.net*). As each user viewed the malicious Web page, their computer was forced to attempt a telnet into our site. The net effect was that about 1500 telnet attempts were originated from sites all over the Internet in a few hours. This would have continued indefinitely if we didn't track down the original source of the attack.

It is also important to note that the particular Web site that launched this attack only had a few thousand visitors per day. There are Web sites on the Internet which fill more than 100,000 service requests per day, and the volume of traffic is increasing rapidly. If a high-volume site were used for an attack of this sort, the sheer volume of traffic generated

[1] The Internet is defined by a set of Requests For Comments (RFCs) that are available on-line through the site internic.com and elsewhere.

[2] The Internet Protocol (IP) is defined by RFC1011 J. Postel, J. Reynolds, "Official Internet protocols", 05/01/1987.

[3] F. Cohen, "50 Ways to Attack Your World Wide Web Systems", Computer Security Institute Conference - fall, 1995, also in Network Security - Dec 1995 and Jan 1996 issues.

[4] See for example, Cheswick and Bellovin, *Firewalls and Internet Security*, Addison Welsley, 1994

[5] Ian Goldberg at Berkeley first published this on the Internet on or about March 1st from UC Berkeley

[6] RFC1738 - T. Berners-Lee, L. Masinter, M. McCahill, "Uniform Resource Locators (URL)", 12/20/1994.

[7] RFC1630 T. Berners-Lee, "Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web", 06/09/1994.

[8] e.g., gopher://all.net:25/0[code for sendmail attack] can be used to cause the browser to attack the SMTP port (port 25).

[9] RFC1436 F. Anklesaria, M. McCahill, P. Lindner, D. Johnson, D. John, D. Torrey, B. Alberti, "The Internet Gopher Protocol (a distributed document search and retrieval protocol)", 03/18/1993.

[10] RFC821 Jonathan B. Postel, "Simple Mail Transfer Protocol", August 1982

[11] RFC854 S J. Postel, J. Reynolds, "Telnet Protocol specification", 05/01/1983..

would be enough to disable many Internet sites. Slight enhancements of this sort of attack might be used to support a wide range of other threats.

In order to better understand the issues underlying this sort of attack, we introduce and discuss a new class of attacks that we call *Distributed Coordinated Attacks* (DCAs). We begin with an information definition and a range of examples that show how DCAs can be used to the attackers' advantage. We then discuss limitations of current defenses against DCAs and characteristics of DCAs, and present a formal structure for considering DCAs. Next we describe the *all.net* incident involving a DCA and show audit trails indicative of several different sorts of DCAs. Finally, we summarize results, draw conclusions, and describe further work.

2 DCA's - A Class of Attacks

In the early days of computing, attacks against computers consisted primarily of individuals exceeding their authority. Audit trails were generally used to track down the sources of these attacks. As defenses and attacks got more sophisticated, the Trojan horse became a popular tool of attack because the attacker could automate elements of the attack, plant those automated elements inside another program, and remain less exposed to the risks of being detected or tracked down. Audit trails could still be used to track down these attackers in many cases, but the task was harder because of the indirection and delays involved.

With the increase in sharing caused by the widespread use of personal computers and floppy disks, computer viruses became viable. Viruses had the advantage of a Trojan horse plus the advantage of unlimited indirection between the source of an attack and the target. Of the many thousands of identified viruses, only a handful of virus writers have been tracked down. The majority of widespread viruses tend to be randomly targeted, which means that they are not coordinated against a particular victim. More recently, the increase in networking and in the use of the Internet, combined with the ability to send documents containing complex programmed macros has led to electronic mail and the Internet becoming a dominant vector for the spread of computer viruses.

In 1994 and 1995, the World Wide Web arose as a prime source of on-line interaction between information services and their clients worldwide and as a new and rapidly growing advertising media. Along with any such technology development, comes new security risks. In the case of the World Wide Web in specific, and the Internet in general, the popularity of Web browsers led to dramatic changes in the way people used computers. In a modern Web browser, individuals who often cannot be reliably traced, load information from service

providers who often cannot be reliably traced, and interpret that information, sometimes in a general purpose fashion.

As we have known for some time, in an environment with sharing, programming, and transitive information flow, computer viruses cannot be completely prevented. [12] This makes the evolving Internet an almost ideal viral computing environment [13] wherein service providers can use the network's computing power to perform highly parallel and widely distributed computations using the facilities of Web browsers all over the world. For example, a Web server could load Java [14] scripts into every visiting browser containing a Java interpreter to perform a brute force attack on a cryptographic key, to perform global searches of databases, or to perform a highly distributed weather forecast. As the reader can imagine, with the great power of highly distributed MIMD computation comes great potential for abuse. When such abuse involves distributed resources coordinated against a particular set of targets, we call it a DCA.

A DCA is informally defined as any attack that uses distributed resources in a coordinated fashion to achieve its goals. As a non-technical example, planting numerous news stories in different publications about slipping delivery dates in a competitors soon-to-be-released product would be a non-automated information-based DCA against a competitor. Since the stories come from distributed sources, they appear to be more credible, and it may be hard for the victim to track the source to an individual, even though one person might ultimately be responsible.

2.1 A DCA Password Guessing Attack

As a more automated example of a DCA, let's suppose that we know of a site *victim.org* that allows remote logins from over the Internet. If we tried to repeatedly guess passwords of users at victim.org from our site, we would almost certainly be tracked down very quickly by the normal audit trails kept by Unix-based Internet sites. Instead, we might use a programmed DCA in a Web page to launch a series of entry attempts where each attempt in sequence guesses a different *(user, password)* pair using a browser as the vector for the attempt.

[12] F. Cohen, *Computer Viruses - Theory and Experiments*, IFIP-TC11 Conference, Toronto, 1984.
[13] F. Cohen, *It's Alive!!!*, Wiley and Sons, 1994.
[14] Java is a language designed by Sun Microsystems to run in Web browsers.

```
DCA password guessing attack:=
        {Display normal Web page;
        cause browser to guess next-password for next-user-ID
                and command victim.org to email (user-id, password)
                to a usenet group via an anonymous email service.
        }

Example: DCA password guessing attack
```

In this attack, instead of sending a copy of the same Web page to each browser, we send a slightly different result on each browser access, the difference being that a different user ID and password pair are tried. If the attempted entry fails, the command to send email is not successful. If the entry succeeds, then the victim sends email via an anonymous remailer to a usenet news group. The whole newsgroup, perhaps 100,000 recipients, gets a message containing something like *(jjones ajs78sj)*. The message could also be marked and encrypted so as to obscure its meaning and make for easy processing by an automatic listener. As soon as the attack succeeds, further attempts against victim.org cease.

Now let's consider what this looks like from victim.org. Assuming that good audit trails and intrusion detection are in place, victim.org will see thousands of failed attempts at login over a fairly short time period. They may come from hundreds of different sites and come in no particular pattern. If victim.org cuts off user accounts after a fixed number of failed login attempts, many or all of the normal users will be unable to use the system, causing widespread and repeated denial of services, and ultimately leading to an alternative security plan. Otherwise, the attacker will eventually find an entry into the system, plant a Trojan horse to permit subsequent reentry, and thus win.

An extension to this attack that plants a more sophisticated Trojan horse on first entry defeats even many one-time authentication systems. Given enough tries, the attacker will eventually succeed in guessing a one-time password, plant the Trojan horse, and use the Trojan horse to originate further malicious actions.

2.2 A DCA Sendmail Exploit Against a Company

Another example of a DCA is an attempt to exploit known security flaws in network services behind a firewall. In this example, a particular company is targeted using widely known historical sendmail vulnerabilities. Even with a modern firewall in place, this attack will succeed against internal machines because all of the behaviors that would be blocked by

the firewall is performed by computers within the firewall.

```
DCA Sendmail Exploit Against a Company:=
        {Display normal Web page;
        If the-user is from victim-company
                then
                cause browser to try next-sendmail-attack
                against next-company-IP-address
        }

Example: DCA Sendmail Exploit Against a Company
```

In this example, only browsers within victim-company are exploited and the attack systematically tries every sendmail hole against every IP address within the victim's domain. The exploit code could then notify the attacker of success by a means similar to that in the password guessing attack, after which the browser-based attack would cease.

If detected, from inside the victim's domain, it would appear as if each of several company computers were trying to break into other inside computers. With good enough audit trails, and assuming the defender knows what to look for, a common thread might be found and evidence gathered.

2.3 A One-per-site Variation

In a widespread DCA attack, one way to conceal the nature of the attack it to eliminate repeatability. To do this, the attacker might create a database of the IP addresses that have been used in the attack and make certain that they aren't used for more than a finite number of attempts per unit time. For example:

```
A One-per-site Variation:=
        {Display normal Web page;
        If this-browser was-not-used in allowable-time-frame
                then
                cause browser to guess next-password for next-user-ID
                etc.
        }
```

Example: A One-per-site Variation

In this case, the attacker can force limits on how many times each browser is used to launch an attack. This makes it far harder to track down the source of the attack since fewer systems administrators at remote sites are likely to cooperate in the investigation if only a single attempt was made from their site. Furthermore, any audit detection scheme with a non-zero tolerance on attacks based on their source, will fail to detect this sort of coordinated attack.

2.4 Spams as DCAs

In the Internet, a spam is loosely defined as the flooding of a system or, as in the case of a mailing list, systems with unwanted information. Spams in the form of unwanted and off-topic advertisements are regularly posted to newsgroups and mailing lists, but there are many other forms of spamming available in the Internet, and Internet sites rarely have adequate anti-spamming defenses.

A recent series of spams were perpetrated by a party or parties unknown who decided to subscribe the Whitehouse, a Time magazine editor, a New York Times reporter, two 'hacker' publications, MTV, and others to about 1,700 Internet mailing lists. [15] In this case, almost 5,000 new email messages per day were sent into the victim's mail boxes, flooding their systems till they ran out of disk space, and causing a lot of inconvenience. Unless you have an automated *unsubscribe* program or some other defense, it takes about a week to get unsubscribed. With an automated subscribe program, an attacker can easily subscribe several hundred people per day to each of these lists from a PC at a public library. [16]

[15] Phillip Elmer-DeWitt, *I've Been Spammed!*, Time Magazine, March 18, 1996, page 77.

[16] To provide improved spam removal, we created a free Internet-based mail spam removal program wherein the victim specifies the site where the spam is originating (e.g., netcom.com) and we provide a command script that will automatically remove the user from all of the mailing lists originating from that site.

A different form of spam is the mailing of massive volumes of useless information to a recipient from a single source. For example, one person threatened to email us the complete sources to the GNU Unix system, an activity that would tie up our Internet link for hours and probably run us out of disk space if we didn't have a defense.

A spam of substantial magnitude is a DCA in that it is distributed, perhaps among almost two thousand widely known mailing lists, and coordinated in the sense that it is launched against a single individual - or perhaps another mailing list. A circular spam, where two mailing lists are subscribed to each other, creates an environment in which any mail to either list acts as a computer virus. [17]

There are some defenses against email-based spams using mailing lists as the vector:

- Email spams can be eliminated by refusing large-volume email from unknown senders. Whenever email arrives from a new sender, the user is told about it, and further email from the same sender is refused until and unless the recipient signals the system to allow ongoing email from that source. In this way, the user is notified of new senders and has the chance to refuse them or listen to them. The user should also be able to refuse further email from a recipient at a later date by changing the setting associated with that user. Since most legitimate first-time connections involve only a single piece of mail, the normal sender would never even notice this mechanism.

- Mailing lists could eliminate their use in signup spams rather easily. Instead of the single signon protocol they use now, they could send a confirmation email containing a unique identifying string to the proposed list member. The new member would have to reply before being added to this list. This would be a simple and effective method of limiting email spams to one notification per list per recipient.

- For the technically inclined among us, there is always the extensive use of digital signatures. If every email were signed, we could authenticate sources before allowing them to post to lists and trace them back to their sources.

- Some types of email spams can also be prevented by sending back a *User-ID* to the sender of email which must be used in subsequent communications. This is unlikely to be effective because of the social environment of the Internet.

- Then there is the moderated list. Moderated lists can only be spammed if they use outside mail servers for automated distribution.

[17] For examples of this and for deeper understanding, review "A Short Course on Computer Viruses", F. Cohen, Wiley and Sons, 1995.

- All lists should include details on how to sign off the list in each mailing. This reduces the workload of the victim dramatically.
- An optional *X-Mailing-List* header for mail would allow lists to be easily differentiated and identified. Today, most mail from mailing lists has a *From* address containing the characters *owner-* or -owner. It is relatively easy to set up a filter that differentiates desired and undesired lists from other mail and refuses mail from undesired lists. This defense requires participation by list owners and works to a reasonable extent today.

At the core of this example and this issue is the concept of workload. Just as modern cryptography is oriented toward gaining a substantial and quantifiable workload advantage over an enemy cryptanalyst, much, if not all, of modern operating system and network protection is oriented toward finding ways to make the workload for defense smaller and driving up the workload for attack, or in other words, making protection more efficient.

2.5 An Example of a Non-DCA

Using the same techniques, it is also possible to launch non-coordinated attacks. For example, to increase the overall noise level of the Internet, the attacker could randomly attack sites or send useless email.

```
A One-per-site Variation:=
        {Display normal Web page;
        send email to postmaster@localhost
        }

Example: A non-coordinated distributed attack
```

In this case, the attacker causes each browser to send mail to the host it is being run on. The net effect is one piece of mail sent to each user. Depending on how gullable the user is and how cleaver the mail is, this might be used for advertising, to fool users into doing something inappropriate, or for other similar purposes.

The key difference between this attack and a DCA is that the attacker in this case is not coordinating the activities toward a specific goal, and in this sense, this attack is similar to typical computer viruses. Just as typical viruses are random in their targets, an uncoordinated distributed attack is random, but just as distributed attacks can be coordinated, viruses can attack specific targets.

2.6 An Alternative Vector for Attack

We have used Web browsers in several examples because of their widespread distribution and the way they automatically run host programs with untrusted arguments, but there are other methods for implementing a DCA. For example, if a widely used service such as the *file transfer protocol* (ftp) has a known defect, [18] that defect can be exploited to use intermediate sites as springboards for a DCA.

```
ftp Defect DCA:=
        {for each springboard in list-of-springboards
        cause springboard to attack victim.org}
```

```
Example: A Defect-based DCA
```

In this case, as a prelude to DCA, a large number of sites are chosen for testing, each is tested for the presence of the defect that permits its use in the DCA. Once a list of such sites is known, each is exploited only once against each target system. By combining this attack with one of the Web attacks above, a DCA can be constructed to cause numerous Web browsers to each exploit one of a large number of defective systems as springboards. In this case, it would take the cooperation of administrators of at least three sites to track down the attacks, and each might have only a very small number of audit records reflecting the overall attack.

2.7 Some Other DCA Examples

DCAs don't have to be launched by a single individual. Several examples of multiparty DCAs have occurred wherein a number of individuals combine forces to increase the impact of the attack. In the extreme, a large number of individuals can attack a site or a group of sites in order to achieve a collective goal. By acting in concert, they may be able to achieve far greater success than any of them could individually achieve, and they may make it harder to track the attack because the apparent multiplicity of sources masks the combined nature of their efforts.

Selective masking DCAs are another interesting approach. For example, a DCA could first check to see if the system accessing a Web page has a sendmail capability, and only

[18] FTP can be caused to use arbitrary ports and send network-specified information to those ports just as the gopher daemon can.

attack through systems without sendmail ports. The chances of minimal administration are far greater when no mail is accepted by the vector machine and the workload on the defender is far higher if the machine acting as the vector refuses automated attempts at response. As a side benefit, when a typical multiuser machine attempts to access the Web page to see if the attack is present, the attack masks itself.

Probabilistic DCAs can be created to only attack under circumstances that are weighted to favor increased difficulty in tracking them down. For example, a DCA could have a low probability of transmitting attack code to vector machines in the attacking country and very high probabilities for machines in The Netherlands. [19] This makes most attempts come through other countries, creating larger delays in administrative communications, making it harder for the defender to find someone willing to look for the attack (it won't appear to be from within their own country), introducing language barriers for cooperating defenders, and exploiting weaknesses in international law and law enforcement.

If an attacker has the ability to access a Domain Name Server (DNS), [20] the DNS can be used to further confuse tracking by creating fictions about the sources of attacks. A similar attack can be carried out by using IP address forgery to give the appearance of a DCA when the attack actually comes from a single source.

2.8 DCAs in IW

A final note is in order on the implication of DCAs on *Information Warfare*. (IW) Information Warfare has been previously defined [21] as *Conflict where [information or information technology] is the weapon, the target, the objective, or the method*. Clearly, by this definition, DCAs are a form of IW. In a low-intensity DCA such as a simple mail spam, defense without substantial retaliation is feasible. As the intensity increases, it may become more and more difficult to defend without retaliation. For example, in the medium-intensity incident detailed later in this paper, there was repeated contacting of systems administrators at sites where attackers were using high-intensity techniques. In the case of the response to c2.org, an escalated response was used involving sending details of each vector and each attempt to the administrators at c2.org. It could be argued, although that was not the intent, that this was a form of IW in which retaliation was used to cause the attackers to back down.

Consider now, a situation in which two military groups of substantial size attack each

[19] There is no reliable and automatic method today for determining where a particular computer in the Internet is actually located, however, most computers can be tracked to a country based on their host name or IP address.

[20] RFC1101 P. Mockapetris, "DNS encoding of network names and other types", 04/01/1989.

[21] F. Cohen, The Information Warfare Mailing List, iw@all.net, 13 Dec 1995

others' information infrastructure in force in the guise of DCAs. This might be an all-out war scenario, involving civilian vectors to attack military and industrial targets. It might involve thousands of Web sites spread all over the world, including sites preconfigured by spies to launch based on predefined messages. It might not only impact the victims, but also bring down substantial portions of the information infrastructure. Unlike computer viruses, DCAs may be for more controllable, and their effects may be far easier to measure. They can be made hard to trace, but it is also easy to identify a particular source in order to demonstrate the capability to the enemy. In these senses and others, DCAs are nearly ideal IW weapons.

3 Characteristics of DCAs

We have found a few common characteristics of DCAs that are noteworthy and may be useful in understanding them better.

- The first and perhaps most important characteristic is that the source of the attack is only indirectly related to the site that appears to be launching the attack. This means that it may be impossible to track down the real source of the attack without the cooperation of people at two or more sites. When combined with the one-per-site variation described above, this means that such attacks may only be reliably tracked to their source by cooperation between two or more sites where a single case of potentially illicit behavior occurs in the vector sites.
- Another important characteristic of a DCA is that, unlike other concerted attacks, a DCA will likely involve a high overall rate of attacks even though the contribution to this rate by each vector site may be low or even singular. Although cleaver DCA attackers may keep rates below the detection threshold of the victim, it is likely that most perpetrators will select higher rates of attack believing that they are adequately shielded by indirection.
- Vectors in DCAs are likely to be completely unaware that their system is being exploited. It may be difficult to explain this to systems administrators who have innocent users that are unwittingly made to participate in an attack which doesn't lead to any outward indication on their system.
- To date, most DCAs have been open loop. This is primarily because the systems they exploit as intermediaries provide only limited function. With the introduction of Java and similar loadable scripts, this is likely to change.

These properties of DCAs appear to make prevention, detection, and response to DCAs quite difficult, but they also lead to a better understanding of defenses.

4 Defenses Against DCAs

Current defense against DCAs consists primarily of three components; prevention, detection, and response.

4.1 Prevention

Preventing DCAs is highly desirable, but it is unlikely to be fully effective in the near future because the nature of modern network services.

- A primary problem is that there is a strong trend in the computing world toward retrieval of software from remote untrusted systems for interpretation on the user's personal computer. This makes the planting of Trojan horses simple and provides a wide venue for their rapid distribution. Nowhere is this taken more to extremes than in the Java language, which is a general purpose programming language designed to operate within a Web browser.
- A secondary problem is that there are so many insecure systems that can be attacked and used as a springboard. In a target rich environment such as this, a sophisticated attacker can break into a popular Web site and plant their code. Even if the attack is traced back, the site containing this code might have been a victim themselves, and if they have inadequate protection, it may be impossible to trace the attack back from there.

One effective defense for sites that don't allow remote access from anywhere is to refuse service to sites that aren't authorized for that particular service. This does not necessarily prevent a DCA, and in fact may help to mask it.

- If services are denied at the router and audit trails are not generated in the process, even though the attack on the particular host may not be noticed, it may be effective in reducing available bandwidth.

- Router or firewall-based defenses are inadequate to prevent DCAs from exploiting a Web browser to attack internal systems. In order for access-controls to be effective they must be set on each system within an organization, not just on a firewall, and they must be set against access from all internal machines as well as external machines. Setting up consistent access controls in a substantial organization may be cost prohibitive.

- There are imperfections in some implementations that may fail to prevent attack in sufficiently high volume and attacks against defensive perimiters that may open inner systems up to DCA attacks. For example, if a DCA attack is being stopped by TCP wrappers, [22] enough volume may cause the build up of so many processes that denial of legitimate services results.

- A wide area DCA attack may involve networks with authorized access to your site, thus making access controls ineffective against that portion of the attack.

Another approach is to prevent services that are known to participate in DCA attacks. Unfortunately, this currently includes FTP servers and Web browsers as well as any service offering URL-based descriptions of services. Since this represents the majority of current Internet traffic, it is unlikely to meet with widespread acceptance, however, there is at least one case where this prevention technique has proven highly effective. That is the case of spams from mailing lists.

In the case of a mailing list spam, the attacker uses a mailing list as the vector for attack. By signing the victim up to a mailing list, the attacker obscures the relationship to the attack and causes high volumes of undesired mail to be sent to the victim. Fortunately, there are only a finite number of mailing lists on the Internet, the bulk of the mailing lists are managed by only a few hundred sites, and mail from most mailing lists comes from an address that is relatively easy to identify. As a result, it is possible to prevent mailings from more than 10,000 known mailing lists by blocking inbound mail that matches any of a list of only a few patterns. [23] Such a defense is in use at some Internet sites and is supported by some of the more advanced mail handling programs.

A third approach to prevention would be to enforce limitations on the software that implements these services. Unfortunately, many attempts to do this by companies including Sun and Netscape have failed repeatedly despite substantial time and effort. As these companies become more aware of the complexity of protection, they are improving, but to date,

[22] Wietse Venema (wietse@wzv.win.tue.nl), *TCP/IP daemon wrapper package*, Department of Mathematics and Computing Science, Eindhoven University of Technology, The Netherlands.

[23] For example in the ManAlMail package, a highly successful filter consists primarily of scanning for the patterns *owner-* and *-owner* in the From address and rejecting all such mail before the content is transmitted. Accepted mailing lists are explicitly listed before this rejection filter.

no Web browser is free of these problems, and almost all ftp servers are still vulnerable. There are likely many other examples of such vulnerabilities that are yet to be discovered.

In essence, today's environment has deeply embedded design flaws that make DCAs possible. The current trends in computing make it highly likely that these flaws will be extended and enhanced, making prevention even more difficult and less likely.

4.2 Detection and Response

If prevention is unlikely in the current and coming networking environment, the next logical approach would probably be detection and response. It turns out that detection of high volume DCAs is very easy, but that the implication of DCA detection and response creates a social challenge that must be met.

One simple detection method that picks up high-volume DCA attacks is a threshold scheme. By setting a threshold on the total number of anomalous behaviors in a given period of time, such attacks become clear immediately. Unfortunately, all threshold schemes suffer from the weaknesses of false positives and false negatives. The false positives come when non-DCA events cause a detection. The false negatives come when a DCA attacker throttles back the attack to remain below the detection threshold.

A common example of a threshold scheme is the detection of three failed login attempts from any given site in a period of a few weeks. Unfortunately, the Internet has millions of sites that can be used to launch DCAs and it is a simple matter to remain below this threshold while launching a very serious attack.

As far as we can tell, the only way to reliably track down a DCA is by coordinating audit trails between a set of sites that form a complete path between the attacker and the target. If the number of attempts per site is small, this means that administrators at other sites must be willing to track down the full details of as little as a single message being sent from their site to another site, and they must also be able to track down the site that their user was using at the time of the incident. If a DCA is created with even a few minutes of delay between being loaded and exploited, the audit tracking problem may be multiplied dramatically for each intermediate location because there may be a large number of possible sources over the time window.

This form of community response creates very serious social challenges. In the Internet, there are a substantial number of sites where administrators either don't care about what their users do to other sites or where a single attempted entry is considered unworthy of investigation. The challenge is to find a way to convince the other administrators to give assistance without having to personally contact each of them one-at-a-time as the attack

rages. When we were under attack, our automated response system sent mail to each site.

Several sites complained vigorously about being contacted. We became quite concerned, but as we investigated, we found evidence in the audit trails that most of the sites making complaints had read about the attack on our site and then attempted to enter in order to generate the warning message just so they could complain. As we looked further, we found evidence that some of these administrators were collaborating with others in a joint DCA.

Another interesting defense against simple Web-based DCAs was proposed by an administrator in The Netherlands. He had the idea of exploiting Internet-based *search engines* to track down attackers. It turns out there are a number of search engines that use automated programs to search large portions of the publicly accessible areas of the Web. They collect millions of Web pages each day and provide remote capabilities for searching the entire Web for the information required. The results are presented as URLs that point to the Web pages that match the search criteria. The suggestion in our case was to search using the *Altavista.Digital.Com* Web-based search engine for:

link:all.net and not url:all.net

This gives you a list of about 400 locations in the Web that refer to our site (all.net), and leaves out all links on our own site. Unfortunately, there is an average lag of about 3 months between the introduction of such pointers and their inclusion in this search process.

4.3 Close The Loop

We have discussed the fact that most current DCAs are open loop attacks, and this leads to a defensive strategy with some promise. Instead of trying to prevent entry, some sites allow limited entry and try to exploit this in their defense. For example, by simulating an entry, capturing the details used to send the notification back to the attacker, and waiting for the attacker to reenter, you may be able to track down all but the most sophisticated attacks intended to gain reentry.

Unfortunately, not all attacks require reentry, nor is it always easy to track the source even when you are observing a perpetrator in action. A common attack in the Internet today is designed to deny services. In this sort of attack, the attacker never closes the loop. Another technique is to use anonymous postings to send commands to the planted Trojan horse as well as to get results back. Nevertheless, this technique may help in tracking down the sources of some attacks.

4.4 Returning Fire

It has often been said that the best defense is a good offense. If the source of a DCA can be identified, and if all other attempts at stopping that attack fail, the only effective response to a DCA may be another DCA. We do not advocate such action except in extreme cases, however, it is important to note that most DCAs seen in the real world have been traceable to a source and that in cases where no assistance is available from the attackers side, there may be no other viable response option.

5 A Mathematical Characterization of DCAs

Mathematically, DCAs can be characterized by the following structure:

DCA:=$(A, V, I, P : (A, I^*, V)$ where

A:=$\{a_1, ..., a_n\}$	a set of attackers
V:=$\{v_1, ..., v_m\}$	a set of victims
I:=$\{i_1, ..., i_h\}$	a set of intermediaries
$P : A \times I^* \rightarrow V$	a set of paths from As to Vs

A few simple results appear to be immediately obvious. The first is that in order to track down the elements of A from V through audit trails of the attack, a set of paths from elements of V to each element of A have to be tracked. In symbols:

$\forall a \in A, \exists (i_{s_1}, ..., i_{s_x}) \in I^*\ (a, (i_{s_1}, ..., i_{s_x}), v) \in P$
where x is a finite integer, $v \in V$, and $i_{s_?} \in I$

In practice today, the maximum size of I is about 20 million. A typical size of I for a Web site ranges from about 100 to 2,000 per day, and many of the elements of this population repeat over time because visitors to Web sites tend to include people who have been there before.

The size of V is most often small (i.e., 1 or perhaps a class C network which has 255 systems), the size of A usually ranges from 1-10, and the number of intermediary systems ranges from 1 to 3.

If these figures are right, this makes the number of possible Ps range from 100 in the minimal case to about $2,000^3 \times 255 \times 10$, or about 180 billion per day. This is quite a range!

Some statistics from the incident at all.net may also be enlightening. In this case, out of about 250 sites emailed in real-time about the incident, only 2 responded within 8 hours

with the information required to track down the attack. This means that only about 1 in 125 sites in I provided the information required to trace the attack. For the purposes of analysis, an I^1 attack was used against all.net, and the mean time for tracking the incident was 4 hours.

Realizing that this analysis is based on a sample size of 1, its accuracy is highly dubious. However, based on this information, if this were an I^2 attack, and if ever administrator that reported audit records also had a zero-tolerance response to the causes of these activities, only one in 125^2 paths would produced a sufficient P to track down the attacker.

At the rate of about 500 sites per day involved in the first step of the attack, and assuming that there were 20,000 preconfigured sites used in sequence for the second step of the attack (e.g., the ftp attack discussed earlier), the time till the source is identified is on the order of 15,000 attempts, or about 30 days. In the limit, each added intermediary degrades the ability to track down the source of a DCA exponentially.

An attacker with serious intentions could be expected to break into one Web site per month to create such an attack. In this case, the source of the attack would be different from the beneficiary. The effort in tracing the attack would likely result only in its location and removal, not in eliminating the attacker or the attack.

Now consider the implications of a Web site that signs up victims to mailing lists through intermediaries. Unless the mailing lists or victims have a mechanism for automatically defending against such attacks. Taking our previous numbers, 500 sites per day might sign up each of 3 victims to each of 3 mailing lists. The effective scale of the attack is 4,500 mailing lists per day. Assuming each mailing list sends an average of 10 mailings per day to its members, this comes to 45,000 more pieces of electronic mail per day created by a single individual using a DCA on a single Web page. Assuming it takes a week to undo all the subscriptions from a typical user, the average persistence of an unwanted subscription is about 3 days. Over the 30 day period before being tracked down, one individual could create $\frac{45{,}000 \times 15 \times 30 \times 3}{30}$ or just over 2 million pieces of electronic mail.

6 Summary, Conclusions, and Further Work

A new class of attacks against information systems resulting from the increasingly networked nature of those systems has been identified, and we have started to explore the issues of attack and defense. Based on initial results, it appears that DCAs offer significant challenges for defenders, that tracking a DCA to its source is, in general, impossible without community-based defenses, and that community-based defenses against DCAs have

substantial limitations.

On the good side, the DCA incident at all.net was quickly identified and successfully tracked to its source. Although this attack was only a very simple one, the fact that it was tracked down quickly and publicly will hopefully act as a deterrent to further attempts. The fact that it happened in the real-world should also act to energize the computing community toward improved defenses.

The statistical details are still not fully understood, but it appears that the DCA described in the appendices was launched by a total of no more than ten people, and that it was probably more like five. Of these, we have identified four individuals. These 5-10 people caused more than 2,000 attempted entries involving more than 800 computers from all over the world. Of the 800 sites we automatically tried to contact in the process of tracking the attack, only six people provided correlating information necessary to track down the sources of these attacks, and of those, only one was a site involving three or more entry attempts. Clearly, contacting all of the sites from which even a single entry was attempted was critical to tracking down the sources of these attacks.

Our initial mathematical characterization and very preliminary results indicate that the magnitude of the problem could be substantial and that defending against multi-hop DCAs is significantly harder and takes significantly longer than against single-hop DCAs, even in a mode where automated response with zero tolerance is universally used. This implies that a dramatic change in the way we handle incident response would be required in order to meet the challenge of DCAs if they become dominant modes of attack.

Based on our results to date, we believe strongly that the most effective DCA defense today is an automated zero-tolerance approach to reporting detected anomalies, and that such a defense will require community tolerance and vigilance. We also believe that DCAs in today's Internet environment provide a very rich environment for attack, and a very challenging environment for defense.

A Incident at All.Net

This paper was an indirect result of a real-world attack on our Internet site (all.net) and some long-term speculation about how vulnerabilities in Internet systems could be exploited. To get a better sense of what a DCA incident looks like, we have included detailed log files of this incident. These also include examples of most of the other phenomena seen in current Internet attacks.

A.1 Incident Background

We historically encountered about one unauthorized attempted telnet into our site per day. We block it before it gets to the login prompt and send email to the administrator (postmaster) at the site from which the attempted telnet took place.

Some people have commented that this message was accusatory and that it indicated that an attack has taken place when one had not. When we talked to other security administrators they told us that people see what they want to see. Some people will call any message abusive. Of the people who called this abusive, at least one of them was also performing port scans of our site (commonly used as a prelude to break-ins) and another was using forged email to convey the message. The audit trails presented later will give examples of this.

After we tracked down one person who attempted some telnets, we got the following response in email:

```
Subject: Who the Hell are You?

Status: RO

I don't care if you coined "computer virus".  I can telnet into whatever
I want.  Don't be writing me back here again.  I WILL get into your
system.  Feel free to write me back for any other complaints you have to
give to me.  Bee-ach!!!!!
```

The systems administrator at that site took this seriously and the individual apologized, but within a day of that incident, we started to see an increase in telnets into our site. Before the DCA, unauthorized attempts to use services (e.g., in.ftpd is the ftp service and in.telnetd is the telnet services) were not very common:

```
Date     Time  Site   Service[PID]    Action-Taken           Remote-Site
------------------------------------------------------------------------------
Feb 27 11:24:23 all in.ftpd[18268]: refused connect from pfizergate.pfizer.com
Mar  4 03:38:07 all in.telnetd[16226]: refused connect from ebola@terra.igcom.net
Mar  4 14:32:57 all in.telnetd[22958]: refused connect from cveley@gunnison.com
Mar  4 19:26:37 all in.ftpd[9914]: refused connect from very.friend.ly.net
Mar  5 22:12:36 all in.telnetd[7960]: refused connect from wfarge@gunnison.com
Mar  5 22:13:22 all in.telnetd[8010]: refused connect from wfarge@gunnison.com
Mar  6 13:17:32 all in.ftpd[26482]: refused connect from edmund.cs.andrews.edu
Mar  7 11:37:10 all in.ftpd[10231]: refused connect from noc.tor.hookup.net
Mar  7 15:24:12 all in.ftpd[21409]: refused connect from 143.211.156.105
Mar  7 16:46:32 all in.ftpd[26084]: refused connect from asdn.on.ca
Mar  8 09:52:26 all in.ftpd[22413]: refused connect from marlowe.physcip.uni-stuttgart.de
Mar  8 20:17:50 all in.telnetd[1057]: refused connect from maxx@osh1.datasync.com
Mar  9 04:48:26 all in.telnetd[25289]: refused connect from dhp.com
Mar  9 18:52:41 all in.telnetd[5561]: refused connect from raven.psc.edu
Mar 10 19:12:08 all in.telnetd[7456]: refused connect from VP24A97S@ubvmsa.cc.buffalo.edu
```

But after the threat was made, activity increased substantially:

```
Mar 11 08:57:36 all in.telnetd[13321]: refused connect from gryphon.psych.ox.ac.uk
Mar 11 12:02:29 all in.ftpd[24237]: refused connect from mail.healthgate.com
Mar 11 13:25:50 all in.telnetd[27915]: refused connect from dunster-lab4.student.harvard.edu
Mar 11 14:21:49 all in.telnetd[781]: refused connect from pm075-00.dialip.mich.net
Mar 11 15:16:27 all in.telnetd[3244]: refused connect from dunster-lab1.student.harvard.edu
Mar 11 15:30:48 all in.telnetd[4020]: refused connect from gh@HELP011.UTCC.UTK.EDU
Mar 11 16:14:52 all in.telnetd[6075]: refused connect from slc118.xmission.com
Mar 11 17:36:53 all in.telnetd[10125]: refused connect from shell.aros.net
Mar 11 17:55:34 all in.telnetd[10899]: refused connect from bifrost.seastrom.com
Mar 11 18:18:11 all in.telnetd[11893]: refused connect from symptom-7.digex.net
Mar 11 18:18:20 all in.telnetd[11913]: refused connect from symptom-7.digex.net
Mar 11 20:55:29 all in.telnetd[20074]: refused connect from bobmacd@netcom21.netcom.com
Mar 11 21:53:22 all in.telnetd[22514]: refused connect from 204.69.200.39
Mar 11 22:53:20 all in.telnetd[25034]: refused connect from remarque.Berkeley.EDU
Mar 11 22:53:51 all in.telnetd[25064]: refused connect from halon.sybase.com
Mar 12 16:30:10 all in.telnetd[11864]: refused connect from dracula.cis.ohio-state.edu
Mar 12 17:56:46 all in.telnetd[15722]: refused connect from ALBATROSS.BBN.COM
Mar 12 18:04:29 all in.telnetd[16053]: refused connect from SCULPEY.BBN.COM
Mar 12 18:04:31 all in.telnetd[16059]: refused connect from nunic.nu.edu
Mar 12 18:11:59 all in.telnetd[16368]: refused connect from mgate.catapent.com
Mar 12 18:19:06 all in.telnetd[16682]: refused connect from emily.la.asu.edu
Mar 12 19:22:38 all in.telnetd[19265]: refused connect from spectre.netseer.com
Mar 12 19:44:07 all in.telnetd[20176]: refused connect from toddw@gol1.gol.com
Mar 12 19:45:11 all in.telnetd[20243]: refused connect from IUS4.IUS.CS.CMU.EDU
Mar 12 19:54:22 all in.telnetd[20700]: refused connect from killian@fusion.leba.net
Mar 12 19:55:35 all in.telnetd[20772]: refused connect from clihost.cli.creaf.com
Mar 12 20:05:42 all in.telnetd[21838]: refused connect from tide02.microsoft.com
Mar 12 20:08:30 all in.telnetd[22380]: refused connect from firewall-user@tide02.microsoft.com
Mar 12 20:27:48 all in.ftpd[23676]: refused connect from pm2-20.ppp.satelnet.org
Mar 12 20:28:45 all in.telnetd[23718]: refused connect from dist@pm2-20.ppp.satelnet.org
Mar 12 21:15:32 all in.telnetd[25586]: refused connect from chelsea.ios.com
Mar 12 22:08:22 all in.telnetd[27772]: refused connect from woodland.digex.net
Mar 12 22:22:32 all in.telnetd[28338]: refused connect from 131.107.2.23
Mar 12 23:42:19 all in.telnetd[1560]: refused connect from UNIX18.ANDREW.CMU.EDU
Mar 12 23:42:45 all in.telnetd[1584]: refused connect from declan@well.com
Mar 13 00:11:42 all in.telnetd[2719]: refused connect from jekyll.piermont.com
Mar 13 00:38:20 all in.telnetd[3766]: refused connect from 206.152.14.3
```

Then, all of the sudden, activity increased dramatically.

```
...
Mar 13 00:45:33 all in.telnetd[4058]: refused connect from sameer@infinity.c2.org
Mar 13 00:46:05 all in.telnetd[4096]: refused connect from sameer@infinity.c2.org
Mar 13 00:46:54 all in.telnetd[4135]: refused connect from windoze.c2.org
Mar 13 00:48:19 all in.telnetd[4211]: refused connect from thad.got.net
Mar 13 00:48:33 all in.telnetd[4230]: refused connect from thad.got.net
Mar 13 00:49:40 all in.telnetd[4282]: refused connect from 7299@halsey.CS.Berkeley.EDU
Mar 13 00:49:57 all in.telnetd[4296]: refused connect from alhazen.CS.Berkeley.EDU
Mar 13 00:51:24 all in.telnetd[4378]: refused connect from windoze.c2.org
Mar 13 00:51:39 all in.telnetd[4398]: refused connect from max2.13.maxnet.westworld.com
Mar 13 00:51:56 all in.telnetd[4432]: refused connect from NUBS65.ccs.itd.umich.edu
Mar 13 00:51:58 all in.telnetd[4419]: refused connect from caribe1-96.caribe.net
Mar 13 00:52:16 all in.telnetd[4454]: refused connect from 164.124.200.11
Mar 13 00:52:39 all in.telnetd[4487]: refused connect from danh@godzilla.EECS.Berkeley.EDU
Mar 13 00:53:07 all in.telnetd[4518]: refused connect from 0.0.0.0
Mar 13 00:53:53 all in.telnetd[4575]: refused connect from 129.71.29.135
Mar 13 00:55:39 all in.telnetd[4654]: refused connect from misf225.cern.ch
Mar 13 00:56:06 all in.telnetd[4679]: refused connect from misf225.cern.ch
...
```

Some event occured on Mar 13 at 00:45:33 that, all of the sudden, pumped up the telnet rate to more than one per minute. It started with what was probably two test attempts by sameer@infinity.c2.org.

We mention this because c2.org was the site eventually tracked down as the source of the attack. If the attacker would have tested from another site, even this audit trail would not appear. It is also important to note that the *3 attempts* threshold common on the Internet would not have detected this activity and that even with this information, there was no obvious way to determine that their involvement involved a Web page with malicious code.

Then, almost exactly an hour after the first part of the DCA was in effect, another incident within the incident happened.

```
...
Mar 13 01:45:38 all in.telnetd[7816]: refused connect from utdppp216.utdallas.edu
Mar 13 01:45:42 all in.telnetd[7824]: refused connect from iww32.mb.Uni-Magdeburg.DE
Mar 13 01:47:06 all in.telnetd[7899]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:47:29 all in.telnetd[7936]: refused connect from iww32.mb.Uni-Magdeburg.DE
Mar 13 01:47:37 all in.ftpd[7948]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:47:39 all in.telnetd[7949]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:47:39 all in.thttpd2[7955]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:47:44 all in.rshd[7963]: refused connect from 150.135.28.168
Mar 13 01:47:46 all in.rlogind[7961]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:47:55 all in.telnetd[8015]: refused connect from iww32.mb.Uni-Magdeburg.DE
Mar 13 01:48:07 all in.telnetd[8036]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:48:17 all in.telnetd[8057]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:48:19 all in.telnetd[8061]: refused connect from lafn.ORG
Mar 13 01:48:54 all in.telnetd[8112]: refused connect from ppp101.jetlink.net
Mar 13 01:50:55 all in.telnetd[8204]: refused connect from ppp101.jetlink.net
Mar 13 01:51:29 all in.telnetd[8257]: refused connect from ppp101.jetlink.net
Mar 13 01:52:03 all in.telnetd[8279]: refused connect from ppp101.jetlink.net
Mar 13 01:54:10 all in.telnetd[8375]: refused connect from magicall2.dacom.co.kr
Mar 13 01:54:23 all in.telnetd[8390]: refused connect from arctic-14.vf.pond.com
Mar 13 01:54:33 all in.telnetd[8386]: refused connect from colt10.qad.com
Mar 13 01:55:17 all in.telnetd[8444]: refused connect from colt10.qad.com
Mar 13 01:55:20 all in.telnetd[8463]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:55:27 all in.telnetd[8481]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:55:35 all in.telnetd[8496]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:55:43 all in.telnetd[8524]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:55:50 all in.telnetd[8552]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:55:58 all in.telnetd[8568]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:55:59 all in.telnetd[8545]: refused connect from magicall2.dacom.co.kr
Mar 13 01:56:06 all in.telnetd[8594]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:56:08 all in.telnetd[8569]: refused connect from magicall2.dacom.co.kr
Mar 13 01:56:14 all in.telnetd[8616]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:56:22 all in.telnetd[8641]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:56:29 all in.telnetd[8657]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:56:33 all in.telnetd[8642]: refused connect from magicall2.dacom.co.kr
Mar 13 01:56:37 all in.telnetd[8685]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 01:56:45 all in.telnetd[8712]: refused connect from yuma13.ResComp.Arizona.EDU
...
Mar 13 02:11:59 all in.telnetd[11541]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 02:12:03 all in.telnetd[11558]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 02:12:07 all in.telnetd[11569]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 02:12:12 all in.telnetd[11579]: refused connect from yuma13.ResComp.Arizona.EDU
Mar 13 02:14:25 all in.telnetd[11723]: refused connect from oliwall.olivetti.dk
Mar 13 02:14:29 all in.telnetd[11717]: refused connect from wsketola.ntc.nokia.com
Mar 13 02:14:38 all in.telnetd[11745]: refused connect from ip93-70.tor.interlog.com
Mar 13 02:14:58 all in.telnetd[11774]: refused connect from ip93-70.tor.interlog.com
Mar 13 02:15:14 all in.telnetd[11805]: refused connect from oliwall.olivetti.dk
Mar 13 02:15:46 all in.telnetd[11840]: refused connect from sir.univ-rennes1.fr
...
```

Note the attempts from Arizona. In this case, a total of more than 200 attempted entries came from one site. The timing of this second event is too close to the first event to be a coincidence. In fact, the 1 hour time difference (to within 5 seconds) between the start of the two attacks may indicate that they were intended to start together but that the person in one state incorrectly compensated for a non-existent time difference. This was a multiple

source DCA. We also had a port scan from Arizona as part of their attack - presumably an attempt to find weaknesses followed by an attempt to overwhealm our defenses.

```
Date    Time       Type  Our-Site/port     Their-site/port      Refused and logged
-------------------------------------------------------------------------------
3/13-01:47:30-17613 tcp 204.7.229.1/3 <- 150.135.28.168/1502 44 syn !pass(9)
3/13-01:47:31-17613 tcp 204.7.229.1/5 <- 150.135.28.168/1503 44 syn !pass(9)
3/13-01:47:32-17613 tcp 204.7.229.1/sunrpc <- 150.135.28.168/1607 44 syn !pass(11)
3/13-01:47:33-17613 tcp 204.7.229.1/3 <- 150.135.28.168/1502 44 syn !pass(9)
3/13-01:47:33-17613 tcp 204.7.229.1/5 <- 150.135.28.168/1503 44 syn !pass(9)
3/13-01:47:35-17613 tcp 204.7.229.1/sunrpc <- 150.135.28.168/1607 44 syn !pass(11)
3/13-01:47:39-17613 tcp 204.7.229.1/3 <- 150.135.28.168/1502 44 syn !pass(9)
3/13-01:47:39-17613 tcp 204.7.229.1/5 <- 150.135.28.168/1503 44 syn !pass(9)
3/13-01:47:41-17613 tcp 204.7.229.1/sunrpc <- 150.135.28.168/1607 44 syn !pass(11)
3/13-01:47:53-17613 tcp 204.7.229.1/sunrpc <- 150.135.28.168/1607 44 syn !pass(11)
```

The person at Arizona was also identified by the combination of our audit trails being sent to their administrator and their audit trails showing the individual responsible.

A.2 Good Morning

Between midnight and 6AM on the next day, we got over 800 attempted telnets from sites all over the world. When the first person in the office that morning came in at 6AM, it took about 30 minutes or so to locate a message in answer to one of our automated responses indicating the following:

```
I had 4 automated messages this morning about a user here attempting to
telnet to your site.

I've investigated this as far as I'm able and don't fully understand
what happened.  I'll give you as much information as I can, you may be
able to make sense of it.

The connections were made from [...] which is a server for ftp, web
pages etc.  No users ever log into it directly.  It runs the W3C web
server in proxy mode for internal clients to access the web.

The connections to your site were from the web proxy attempting to
access the URL http://all.net:23/ - as the target port is 23 this
explains why it looked like a telnet attempt.  I've looked through the
proxy logs and can see the 4 connection attempts on behalf of a user at
one of our remote sites[...].  I checked all the pages he looked at
around that time and can't see any links to the above URL.  I also spoke
to him and we're both at a loss to explain where this URL came from.
For what it's worth I'm satisfied he's not deliberately making these
attempts - it's hard to see what good an HTTP connection to your telnetd
would be even if it was successful.

...
I'd be most interested if you can make sense of any of this.  I hope
this has been helpful, and please contact me if you need any other
information.
...

---  proxy.log, filtered for all connections made by...
...
[13/Mar/1996:07:57:08 +0000] "GET http://www.c2.org/hacknetscape/n.gif
[13/Mar/1996:07:57:09 +0000] "GET http://www.c2.org/blosser_cy/gifs/...
[13/Mar/1996:07:56:56 +0000] "GET http://www.c2.org/hacknetscape/
...
```

A visit to the listed Web pages found that in the c2.org home page, there was code that automatically, and without the knowledge or consent of the user, caused Web browsers to telnet into our site. A copy was secured for evidentiary purposes. Just to make this clear, and in case you missed it:

> Innocent user visits Web page.
> Browser automatically telnets into a third site.

This is interesting because it means that (and is the first major real-world incident where) a Web page can contain code that causes users' browsers to automatically launch attacks

against third parties. The user doesn't have to click a special button to do this. It is a Trojan horse embedded within the normal loading of another Web page or image.

Normally, this would be very hard to track down because the attacks come from hundreds of different locations even though they are all initiated from one site. In this case, it took only 20 minutes because two audit trails were combined together.

It was the combination of our automated response with the other administrator's audit trails made at almost the same time that gave the clues needed to track this down. Without both, the attack could have raged for a long time before anyone would have been able to track it. We believe that the attacker was counting on this.

As soon as the nature of the incident was clear, we changed our automated response message to indicate the source and nature of the attack and tried to contact the site with the malicious code. We sent email, called and left a message, and put a message on their beeper. Next, we remailed all of the systems administrators who had gotten mail overnight (250 of them) identifying the nature of this attack and asking them to determine whether or not this was the case in their part of the overall incident. We also asked them to contact the offending site (c2.org) and register their distaste with this situation. All of this took about 20 more minutes.

Next we ran our internal audit tool to verify system integrity. It took about 10 minutes to determine that no logins had been successful and no system files were altered, to extract details on the sequence of events in a form useful for follow-up, and to detect that during the larger incident, there were several sub-incidents, including an automated attempt to find remotely accessible services on our system via a port scan similar to that done by *Satan*. [24]

While the audit analysis was underway, we followed up on the port scan with an additional email to the site administrator, found from the audit trails that the same site had made scores of attempted telnets in a short time period while the attack was at its peak, and reported a summary of the attack to their systems administrator.

After a significant amount of time passed without response from the c2.org site originating the attack, we decided to forward details of each attempt to that site as they ocurred along with the information required for them to prevent its continuation. Since we believed that a systems administrator at that site might be involved, we wanted to make certain that the message got through, so we had copies of these responses sent to the *root* account, the *postmaster* account, and the person listed as the site administrator. Within an few hours, the site responsible for the attack removed the malicious code and the rate of telnets slowed.

[24] SATAN was written by Dan Farmer and Weitse Vienema as a testing tool to help identify known Internet vulnerabilities.

A.3 Examples of Other Intentional Abuses

Time sequenced selective extraction of log entries such as those shown above are useful in tracking overall activities, but they are very difficult to analyze from a standpoint of tracking attacks and establishing what happened on a case-by-case basis. To get this sort of information, it is necessary to combine different audit sources.

Here's a series of 14 attempts, each from a different computer at the same Internet Service Provider (ISP). The 15th was the administrator doing a test. In these listings, a star indicates an unauthorized attempt and a period indicates another event from the same network or host. In this case, we contacted the site within a few minutes by phone, but they claimed they could find no common thread. We are still encouraging them to investigate more deeply. A cross-host analysis is vital to detecting these sorts of DCAs.

```
*/. Site        User     Date    Time     Service                      Details
------------------------------------------------------------------------------
*** Network sfo4.as.crl.com has exceeded detection threshold: net total = 15
...
*crl.com unknown 1996/03/14 17:44:37 in.telnetd 28248 all twist crl.com to (/bin/cat /etc/telmessage)&
*crl2.crl.com unknown 1996/03/14 17:44:37 in.telnetd 28249 all twist crl2.crl.com to (/bin/cat /etc/telmessage)&
*crl3.crl.com unknown 1996/03/14 17:44:39 in.telnetd 28250 all twist crl3.crl.com to (/bin/cat /etc/telmessage)&
*crl4.crl.com unknown 1996/03/14 17:44:40 in.telnetd 28259 all twist crl4.crl.com to (/bin/cat /etc/telmessage)&
*crl5.crl.com unknown 1996/03/14 17:44:41 in.telnetd 28269 all twist crl5.crl.com to (/bin/cat /etc/telmessage)&
*crl6.crl.com unknown 1996/03/14 17:44:41 in.telnetd 28271 all twist crl6.crl.com to (/bin/cat /etc/telmessage)&
*crl7.crl.com unknown 1996/03/14 17:44:43 in.telnetd 28275 all twist crl7.crl.com to (/bin/cat /etc/telmessage)&
*crl8.crl.com unknown 1996/03/14 17:44:44 in.telnetd 28285 all twist crl8.crl.com to (/bin/cat /etc/telmessage)&
*crl9.crl.com unknown 1996/03/14 17:44:45 in.telnetd 28291 all twist crl9.crl.com to (/bin/cat /etc/telmessage)&
*crl11.crl.com unknown 1996/03/14 17:44:46 in.telnetd 28304 all twist crl11.crl.com to (/bin/cat /etc/telmessage)&
*crl10.crl.com unknown 1996/03/14 17:44:46 in.telnetd 28298 all twist crl10.crl.com to (/bin/cat /etc/telmessage)&
*crl12.crl.com unknown 1996/03/14 17:44:49 in.telnetd 28310 all twist crl12.crl.com to (/bin/cat /etc/telmessage)&
*crl13.crl.com unknown 1996/03/14 17:44:50 in.telnetd 28321 all twist crl13.crl.com to (/bin/cat /etc/telmessage)&
*crl14.crl.com unknown 1996/03/14 17:44:52 in.telnetd 28322 all twist crl14.crl.com to (/bin/cat /etc/telmessage)&
.mail.crl.com unknown 1996/03/14 20:22:06 sendmail 10052 all connect from mail.crl.com
.mail.crl.com unknown 1996/03/14 22:14:32 sendmail 17133 all connect from mail.crl.com
.mail.crl.com unknown 1996/03/14 23:19:54 sendmail 20935 all connect from mail.crl.com
.mail.crl.com unknown 1996/03/15 00:24:53 sendmail 24818 all connect from mail.crl.com
*crl11.crl.com unknown 1996/03/16 17:36:20 in.telnetd 10025 all twist crl11.crl.com to (/bin/cat /etc/telmess)&
.crl11.crl.com unknown 1996/03/16 17:36:55 in.thttpd 10118 all twist crl11.crl.com to /usr/etc/in.thttpd crl11.crl.com unknown
...
```

The next example is part of a series where the systems administrator told us to stop sending *all this crap* about attempted telnets to them. They also did a port scan.

```
*** Host soda.csua.berkeley.edu has exceeded detection threshold: host total = 27
*soda.csua.berkeley.edu unknown 1996/03/13 15:12:01 in.telnetd 24216 all refused connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 15:12:39 in.telnetd 24326 all refused connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 15:13:10 in.telnetd 24370 all refused connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 15:13:51 in.telnetd 24461 all refused connect from soda.CSUA.Berkeley.EDU
.soda.csua.berkeley.edu unknown 1996/03/13 15:14:12 sendmail 24511 all connect from soda.CSUA.Berkeley.EDU
.soda.csua.berkeley.edu unknown 1996/03/13 15:14:37 sendmail 24550 all connect from soda.CSUA.Berkeley.EDU
.soda.csua.berkeley.edu unknown 1996/03/13 15:15:05 sendmail 24607 all connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 15:20:03 in.telnetd 25282 all refused connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 15:38:50 in.telnetd 28188 all refused connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 16:03:54 in.telnetd 1669 all refused connect from soda.CSUA.Berkeley.EDU
```

Next, we see the service in.fingerd which explicitly states that their are no user accounts

on the all.net system. But this has no effect.

```
.soda.csua.berkeley.edu unknown 1996/03/13 16:04:26 in.fingerd 1735 all connect from soda.CSUA.Berkeley.EDU
.soda.csua.berkeley.edu unknown 1996/03/13 16:05:01 in.fingerd 1819 all connect from soda.CSUA.Berkeley.EDU
.soda.csua.berkeley.edu unknown 1996/03/13 16:05:11 in.fingerd 1846 all connect from soda.CSUA.Berkeley.EDU
.soda.csua.berkeley.edu unknown 1996/03/13 16:05:17 in.fingerd 1865 all connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 16:06:19 in.telnetd 2006 all refused connect from soda.CSUA.Berkeley.EDU
.soda.csua.berkeley.edu unknown 1996/03/13 16:28:49 in.fingerd 4364 all connect from soda.CSUA.Berkeley.EDU
.soda.csua.berkeley.edu unknown 1996/03/13 16:29:09 in.fingerd 4396 all connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 16:31:15 in.telnetd 4600 all refused connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 16:39:29 in.telnetd 5373 all refused connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 16:51:37 in.telnetd 6425 all refused connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 16:56:14 in.telnetd 6845 all refused connect from soda.CSUA.Berkeley.EDU
.soda.csua.berkeley.edu unknown 1996/03/13 16:57:02 sendmail 6915 all connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 17:07:13 in.telnetd 7796 all refused connect from soda.CSUA.Berkeley.EDU
.soda.csua.berkeley.edu unknown 1996/03/13 17:13:54 sendmail 8370 all connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 17:30:26 in.telnetd 9798 all refused connect from soda.CSUA.Berkeley.EDU
```

Then a port scan comes - note the sequence of attempts on in.rlogind, in.fingerd, in.ftpd, and in.telnetd. This is indicative of successive attempts on one service after the next. The timeframe for this port scan indicates that it was done at a far slower rate than most port scans, presumably so that it would be hidden within the other ongoing activity at the target site.

```
*soda.csua.berkeley.edu unknown 1996/03/13 17:30:46 in.rlogind 9824 all refused connect from soda.CSUA.Berkeley.EDU
.soda.csua.berkeley.edu unknown 1996/03/13 17:33:24 in.fingerd 10041 all connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 17:47:44 in.ftpd 11188 all refused connect from soda.CSUA.Berkeley.EDU
*soda.csua.berkeley.edu unknown 1996/03/13 18:20:54 in.telnetd 13912 all refused connect from soda.CSUA.Berkeley.EDU
...
```

While this was underway at *soda*, we also got attempts from *uclink.berkeley.edu*:

```
*** Host uclink.berkeley.edu has exceeded detection threshold: host total = 23
*uclink.berkeley.edu unknown 1996/03/13 16:31:47 in.telnetd 4640 all refused connect from uclink.Berkeley.EDU
*uclink.berkeley.edu unknown 1996/03/13 16:36:47 in.telnetd 5142 all refused connect from uclink.Berkeley.EDU
*uclink.berkeley.edu unknown 1996/03/13 17:21:08 in.telnetd 9003 all refused connect from uclink.Berkeley.EDU
*uclink.berkeley.edu unknown 1996/03/13 17:21:22 in.telnetd 9031 all refused connect from uclink.Berkeley.EDU
*uclink.berkeley.edu unknown 1996/03/13 17:33:00 in.telnetd 10006 all refused connect from uclink.Berkeley.EDU
*uclink.berkeley.edu unknown 1996/03/14 00:43:44 in.telnetd 17128 all refused connect from uclink.Berkeley.EDU
*uclink.berkeley.edu unknown 1996/03/14 01:34:49 in.telnetd 20679 all refused connect from uclink.Berkeley.EDU
*uclink.berkeley.edu unknown 1996/03/14 01:35:51 in.telnetd 20772 all refused connect from uclink.Berkeley.EDU
*uclink.berkeley.edu unknown 1996/03/14 01:35:55 in.telnetd 20799 all refused connect from uclink.Berkeley.EDU
*uclink.berkeley.edu unknown 1996/03/14 01:36:43 in.telnetd 20860 all refused connect from uclink.Berkeley.EDU
*uclink.berkeley.edu unknown 1996/03/14 01:37:16 in.telnetd 20921 all refused connect from uclink.Berkeley.EDU
...
*uclink.berkeley.edu unknown 1996/03/15 22:21:09 in.telnetd 22750 all twist uclink.Berkeley.EDU to (/bin/cat /etc/telmess)&
```

There were also attempts from other *Berkeley.EDU* computers. All told, there were 74 attempted entries from Berkeley. It is also noteworthy that the site *c2.org* and *Berkeley.EDU* are physically close to each other. One cannot help but believe that the participants were combining forces in this DCA.

In the next example, an attacker entered a computer located at a community college in Pennsylvania, took over the superuser account and caused the *cron* service to attempt one entry every 5 minutes. It took a few tries before the user decided to automate and it took

the administrator two hours to respond after we got him on the phone.

```
*** Host 198.133.170.253 has exceeded detection threshold: host total = 31
*198.133.170.253 unknown 1996/03/13 19:56:33 in.telnetd 24601 all refused connect from 198.133.170.253
*198.133.170.253 unknown 1996/03/13 20:01:45 in.telnetd 25009 all refused connect from 198.133.170.253
*198.133.170.253 unknown 1996/03/13 20:02:01 in.telnetd 25046 all refused connect from 198.133.170.253
*198.133.170.253 unknown 1996/03/13 20:03:02 in.telnetd 25138 all refused connect from 198.133.170.253
*198.133.170.253 unknown 1996/03/13 20:03:57 in.telnetd 25228 all refused connect from 198.133.170.253
*198.133.170.253 unknown 1996/03/13 20:08:58 in.telnetd 25630 all refused connect from 198.133.170.253
*198.133.170.253 unknown 1996/03/13 20:13:57 in.telnetd 26141 all refused connect from 198.133.170.253
*198.133.170.253 unknown 1996/03/13 20:18:57 in.telnetd 26542 all refused connect from 198.133.170.253
*198.133.170.253 unknown 1996/03/13 20:23:57 in.telnetd 26930 all refused connect from 198.133.170.253
*198.133.170.253 unknown 1996/03/13 20:28:57 in.telnetd 27317 all refused connect from 198.133.170.253
*198.133.170.253 unknown 1996/03/13 20:33:57 in.telnetd 27715 all refused connect from 198.133.170.253
*198.133.170.253 unknown 1996/03/13 20:38:55 in.telnetd 28108 all refused connect from 198.133.170.253
*198.133.170.253 unknown 1996/03/13 20:43:56 in.telnetd 28499 all refused connect from 198.133.170.253
*198.133.170.253 unknown 1996/03/13 20:48:56 in.telnetd 28890 all refused connect from 198.133.170.253
...
*198.133.170.253 unknown 1996/03/13 22:10:07 in.telnetd 6022 all refused connect from 198.133.170.253
```

Then we have this email from a systems administrator:

> "Sure, I would be more than happy to track down breakin attempts. If you get repeated attempts please contact me, if it is a single telnet without any attempt to enter a false username or truly break in I don't need to be bothered."

Here's the audit trail:

```
fully.organic.com unknown 1996/03/14 23:46:11 in.thttpd 22685 all twist fully.organic.com to /usr/etc/in.thttpd fully.organic.com unknown
fully.organic.com unknown 1996/03/14 23:46:11 thttpd 22685 all cat /index.html
fully.organic.com unknown 1996/03/14 23:46:19 in.thttpd 22694 all twist fully.organic.com to /usr/etc/in.thttpd fully.organic.com unknown
fully.organic.com unknown 1996/03/14 23:46:19 thttpd 22694 all cat /allnet.gif
fully.organic.com unknown 1996/03/14 23:46:46 in.thttpd 22716 all twist fully.organic.com to /usr/etc/in.thttpd fully.organic.com unknown
fully.organic.com unknown 1996/03/14 23:46:51 thttpd 22716 all cat /journal/netsec/9604.html
```

This last entry (cat /journal/netsec/9604.html) was the description of the ongoing incident up until that time. After getting this file through a World Wide Web browser and having about 6 minutes to read it, this user proceeded to try to telnet into our site:

```
fully.organic.com unknown 1996/03/14 23:53:00 in.telnetd 23097 all twist fully.organic.com to (/bin/cat /etc/telmess)&
fully.organic.com unknown 1996/03/14 23:58:12 in.thttpd 23456 all twist fully.organic.com to /usr/etc/in.thttpd fully.organic.com unknown
fully.organic.com unknown 1996/03/14 23:58:13 thttpd 23456 all cat /readonly.html
```

When we sent the combined log entry to this administrator via email, he knew that we knew that he had done this intentionally to create an excuse to complain.

Here's another adminstrator with a poor attitude. After being notified of a total of 21 different attempted telnets from his site, many of them after notice had been served to the individual users, he advised us:

> I don't care if one of my users tried to telnet into your site, live with it, and

> please do not send me any more of the BULLSHIT! Telnetting to someones site is not my idea of a serious offense, in fact I see nothing wrong with it. Maybe I will instruct all my users to telnet to your machine. What I do see as a problem is the fact that because of your anal security you feel you have the right to send me mail telling me about your so-called problems. I will have to start ignoring your mail if I see any more of it coming into my mailbox. Please leave me out of your problems.

The audit trails indicated that numerous attempted entries came from their domain (mindsping.com), beginning with the morning of the 13th when the DCA started, and continuing into the next week. The total over the next several days came to 34 attempted entries.

Then on March 18, we got the following helpful lead on this particular site:

```
Date: Mon, 18 Mar 96 17:32:04 EST
To: fc@all.net
Subject: breakin

Hi:
There is an http site being distributed publicly and being
spread around.  Yes some one is playing a joke on your site which is
kind of sad.  Here is the site you can take a look at and probably do
something about it.  I would be pissed about it too.
                     "http://www.shorty.com/[details witheld]"

This might help you solve a part of your problem.
```

It turns out that www.shorty.com resolves to a domain within mindspring that is leased by a client for their use.

```
% traceroute www.shorty.com
traceroute to widow.mindspring.com (204.180.128.20), 30 hops max, 40 byte packets
...
```

The site www.shorty.com prohibits Web access from our site, however, we asked another Internet site to get the data for us. This is an example of a slightly more advanced DCA component, designed to make it hard for the site under attack to track the details down.

The URL above has a link for *hackers* to *hack into* the *all.net* computer. When they press on the link, it causes their browser to try to connect to the telnet port at our site. The information contained in their Web page is also false and misleading.

A.4 Summary of the Incident

The major DCA components included:

- A breakin at a community college in Pennsylvania where the attacker attained root access and rigged the University computer to automatically telnet to our site every 5 minutes. The account was found and terminated and further investigation is underway.
- A port scan followed by a series of scores of attempts to telnet into our site for over an hour from a University site in Arizona. The perpetrator has now been found and is being subjected to administrative action at the school.
- Several IP spoofing attempts that we are tracing down to the specific dial-in accounts used to launch the attacks. To date, the Internet Service Providers (ISPs) involved in these incidents have not acted to stop their clients from launching further attacks.
- An intentional insider corruption of a Web page designed to turn innocent browsers into launchpads for their attack (c2.org). We tracked this person down and further investigation is underway.
- Several Web sites include information that is misleading people into telnetting into our site under the auspices of getting a letter from a self-proclaimed computer security expert or other similar claims. We have tracked several these people down, and some have altered their policies, but others continue with impunity.
- A systems administrator at a prominent university who did a port scan followed by numerous telnets. This person was identified and, after we publicly identified the user on our Web site, the activities from that site have stopped.
- Someone who apparently broke into another major university's system and launched port scanning attacks as root. The individual responsible was not tracked down, but the account was removed.

A.5 Comments

We encountered a wide range of responses from systems administrators, and we thought we would share the range with you:

- **How can we help?** - This was the dominant response by far, and these administrators were the ones that ended up tracking down attackers in their sites and helping to track down the high volume intruders.

- **Why don't you attack back?** - Because it's against the law and because two wrongs don't make a right. There is a certain level of frustration involved in not being able to fight back but, if you can't take frustration, you should probably not be a systems administrator or an information security professional.

- **Why do you bother me with this? Don't send me any more mail about this. (primarily from The Netherlands)** - We gave two choices. Either we could terminate all access from that site, or continue to send the messages. The unanimous response was to continue to get our automated responses.

- **Why don't you give in to their demands and shut down your response to attacks?** - The real reason not to do this is because this is what allows DCAs to succeed. Recall that less than 1 in 100 of the messages we sent out resulted in the information required to track down the source of the attack and that the reason that administrators at other sites were able to identify the items of interest was because of the time proximity of the events. Consider also that the incident involved people from all around the world. Delaying these messages might result in a 24 hour delay in communications with distant systems administrators, resulting in lower response rates, longer delays, and the unnecessary extension of the incident.

- **Why not ignore individual attempts and only respond to more persistent attempts?** - Because then attacks such as those in this incident would go completely undetected as a set of independent events. It's only by tracking each attempt that coordinated attacks such as these are detected and stopped.

- **You're not going after real attackers here, only casual perusers.** - This depends on what you call casual. We consider several thousand attempted entries involving misuse at several hundred sites, IP forgeries, breakins at several sites, and misleading advertisements to be a pretty serious attack.

In total, we detected more than 2,300 attempted entries, about 1,500 of them in the first day. More than 230 of the 1st day entry attempts were from the *Arizona* site. There were a total of about 800 different computers involved, and only about 200 of those were involved in more than 3 entry attempts. This means that more than 600 computers probably belonged to innocent third parties whose computers were fooled into attacking our site.