Global Information Assurance Certification Paper

Tarpits for an Imperfect World

Christopher Shove Cook

July 9th, 2002

Abstract

Life isn't perfect. While many security plans are predicated on people making rational choices based on defined risks, the real environments I've seen have been less deliberate. Sometimes, security means mitigating security holes you can't close. One exciting new approach – **tarpitting** or slowing down unwanted traffic – has been used in email and Web environments for years. In 2001, Tom Liston wrote the "LaBrea" program (http://www.hackbusters.net/LaBrea) that creates a tarpit for unwanted TCP/IP connections. This can be a whole new layer of protection administrators haven't had before, and it's an idea I'm almost forced to describe as *seductive*. Taking unused IP addresses, LaBrea creates ghost systems that mimic open, unprotected computers. Where tarpits go beyond similar ideas like honeypots and honeynets is that a tarpit is *really* sticky – it not only accepts TCP connections, after completing a connection, it doesn't let go, keeping the connection alive until the originating system deliberately breaks it. The tarpit **traps** attackers.

I implemented a LaBrea Tarpit as a prototype to see how what was involved, how reliable it was, how much work it was to configure and maintain, and how much protection it could offer. The software was relative easy to install, although the configuration had some surprises. After the first few days, the tarpit had created a thousand ghost systems and was fielding over one million transactions a day, trapping tens of thousands of attackers, yet it did all this while consuming little network bandwidth and almost no computer loading. After the first week of running a tarpit, the number of malevolent connections dropped by 60 percent and stayed low during the time the tarpit was run. As a side benefit, the tarpit was effective as a new kind of network-based intrusion detection system. In the end, although the scans and connections haven't stopped, I believe the real systems on my networks were either less exposed to scanning, or at least better hidden among the decoys.

Background

Anyone connected to the Internet lives with the networking equivalent of thugs roaming the streets, trying each door to see what was left open. My company was forced to build and harden new systems offline: unhardened systems connected to the Internet have an uncorrupted life measured in seconds. In that same way, users who set up ISS BlackICE[1] (which I recommend as a cheap, host-based Intrusion Detection System[2]), ZoneAlarm[3] or any of the other personal firewalls, quickly see so many warnings about blocked connection attempts that most users I talked to had turned off routine notifications in frustration. The traffic in malevolent connections is that bad.

If this is the threat, what's the vulnerability? However attractive it might be for security professionals to design fortified systems with carefully planned and layered defenses, security in the real world remains less… tidy. Many organizations resist setting rules that are seen as limiting or intrusive, and security policies and practices are often weakened by political and idiosyncratic exceptions. Some organizations, by their structure and charter, have to remain open.

Forget security design limitations for a second. Even when security is tightly built and layered, the work usually stops at the walls of your network. Microsoft's recommendations for IIS and the Code Red worm ("A Very Real and Present Threat to the Internet" [4]) covered detection, removal and patching a system. The recommendations from Security Focus, McAfee, Symantec, and BugTraq may differ in details, but the goals were the same: find the problem, clear it from your systems, and harden them against re-infection. That's perfectly good as far as it goes, too, but when did we stop noticing that we've given up on the root cause? As with the personal firewall notifications, most IT security people I've talked to have given up on tracing attacks back to their source and getting them off the air. I know that, for me, it was a difficult and relatively unsuccessful chore that took me away from more productive work. Eventually I gave up trying to fix the source, and we fell back to blocking the worst offenders at the borders of our networks. I doubt I'm alone in that progression.

If the IT community's cure for Internet-wide infections is to patch and protect as many systems as possible, can anyone reasonably say it's worked? By the evidence I see on any open Internet connection, abstinence does not seem to have held back this tide.

One corollary to the Law of Unintended Consequences says that no good deed goes unpunished. Locked down networks are easy to detect – most network protections are *designed* to be obvious – but one result is that scanners and hackers can test and skip thousands of protected addresses in minutes, making it easier to focus their attention on the less protected systems. Going back to my analogy, suppose the thugs knew at a glance which houses on a block were strongly locked? Although it's an unpleasant thought, obvious protection acts as an unintentional multiplier against the less protected.

What makes tarpits promising is that they change the ground rules. If running a tarpit can actually slow down or trap automated searches, and if they can salt the scanner's results with bogus and possibly damaging data, we might be able to protect the strong without accidentally endangering the weak. This promise of delaying or stopping the thugs while sprinkling the landscape with attractive decoys is what makes a tarpit such an exciting idea.

Risk Evaluation

I'm going to evaluate the security in my company as an example. The details are specific to my case, but the results – dangerous gaps in security – are common.

I see the security in my environment as abysmal, although in fairness, it's probably average for an organization involved in government-funded research and development. The easiest place to start is with this question: Have any of our systems been hacked? You bet. Over the last year half a dozen systems were hacked. I don't have to resort to theoretical arguments or presentations with shocking counts of unsuccessful hacking attempts. Or at least so you'd think. Open SMTP relays are especially popular right now (yes, that was us sending you Spam) with FTP abuse second (temporary storage for hacked DVDs) and forgotten Linux systems cracked down to the r00t following a close third. For some reason, we seem to be especially popular with French and Brazilian hackers, but forget that part. We usually find the hacked systems reasonably quickly from the network volume, unusual traffic or other symptoms, and then kill them quickly (and possibly their owners). Still, we've been hacked many times, and I can expect, if the basic conditions continue as is, we'll continue to have our systems cracked.

Why would this level of insecurity be tolerated? It's a complicated answer. My company is really a small section of a much larger, somewhat disjointed corporation. That corporation designs and implements their policies based on the greatest good for the greatest number, but sometimes we're not part of that number. Working out the differences often looks more like a frozen tug of war than a reasoned discussion.

What kind of security structure does my company have? It's best to start from the top down. My company has no IT Security policy for our Enterprise, for individual departments or for specific issues. Our parent corporation has their IT Security policy they follow, but that policy conflicts with our business requirements. For complicated political reasons we've haven't adopted our parent corporation's policy, adapted it, or written our own policy. While this isn't sensible, it certainly appears stable. This doesn't mean we've abandoned the idea of a security policy, but the evidence is that we have to work without one, at least in the short term.

We have several facilities, some of which we share with partners and part-time competitors. Welcome to the new competitive paradigm. Some of these facilities have firewalls while some only do basic filtering at the border routers. Servers and workstations have been hacked in both situations, and oddly, the systems protected by firewalls have fared slightly worse. Discounting firewall logs, we don't run any formal Intrusion Detection Systems (IDS), either network-based or host-based. Some people (me among them) have set up informal detection systems of varying capabilities, and some systems have those capabilities built in. Every day, the systems that pay attention report unsolicited connection attempts, which we dutifully and ineffectually report. As with the IT Security Policy question, we're working on improving the network-level detection and security, but we aren't depending on, or expecting, rapid change.

In case you were getting depressed, this is where things get better. Our facility servers have been hardened using procedures adapted from the NSA Security Guidelines (http://www.nsa.gov/)[5]. We pay attention to announced vulnerabilities and keep our

systems patched and updated. With some notable exceptions (which I won't note here), our facility servers have been secure and dependable.

The more limited project-based servers, legacy systems, prototypes and "temporary solutions" are a different, mixed bag. Some very smart and talented people created some of these systems, while others are half-thought, cobbled-together accidents waiting to happen. Maintenance ranges from active and current though minimally documented legacy systems. We haven't implemented a systemic "scan, detect and repair" procedure (yet) for the facilities, although we do ad hoc network scanning for specific vulnerabilities.

Almost all the desktop systems have anti-virus software, automatically updated from central sites. The central sites are updated promptly and consistently, and the clients are (finally) getting and installing the updates routinely. This has been very effective in stopping email worms from getting a foothold, and, where they do get in, it limits their spread.

Interestingly, our users have been one of the most effective layers of defense. The IT groups spend considerable time and effort communicating with our users about what's happening, what the current threats look like and what they can do to protect themselves. We host routine users' group meetings where we try for two-way communication, telling them what we know and listening to what they need. The results have been reassuring. Time and again, viruses and worms that crippled other major businesses passed us by. Since our institutional-level protection is so thin, my conclusion is that we've been saved by our users' careful practices. I cannot overstate the benefits we've seen from quickly informing our users, and listening to what they need.

So we have an organization that works well at the user and desktop levels, but with little depth of defense, and institutional barriers to many enterprise-level options. The challenge becomes: how to thicken this defense without getting tangled in existing conflicts?

Tarpits

I won't explain Tarpits in great depth – that's been done already by much smarter people than me – but here are the basics. Devin Carraway (http://www.devin.com/peachpit/), who markets several Tarpit tools, includes this description on his web site:

> "**Tarpitting**," briefly, is the tactic of slowing down a TCP connection when a hostile party's autonomous agent is on the other end -- an approach originally conceived to deal with spammers, though in principle it should work well with most any known-hostile web spider, to some degree.[6]

The idea of a network tarpit seems to have started with the first bulk commercial email (Spam). During less cynical times ("when everybody knew everybody"), email SMTP hosts were built to move email without asking a lot of questions. They were relatively simple to run and nobody worried about misuse much because email traffic was small enough that many people saw running an SMTP host as a kind of low-key public service. Spam changed that when commercial users started dumping huge quantities of email on unsuspecting SMTP servers. If they congested or even brought down an email server, it didn't really matter to them, since there were literally thousands of other open SMTP hosts available. That abuse was hard to stop for two reasons. First, SMTP hosts were less sophisticated back then, and many couldn't block email relaying. Secondly, many networks had deployed email with one of the requirements being open SMTP hosts for sending out email. Changing from that initial assumption required redesign and redeployments that couldn't be done overnight. (The third reason was system administrators who chose to allow Spam through their systems, an indifference that still engenders boycotts, but that's a different problem.)

Tarpitting was a technique developed as one way of slowing down outsize relays without harming legitimate relay traffic. I can't quote when it started, but a later example John Buckman wrote an article called "Stop Spam Now!" in 1998 (http://john.redmood.com/stopspam.html)[7] that described different characteristics of Spam vs. legitimate email, and included a section describing how to tarpit Spammers. Since his description included tarpitting as a given concept, I assume the idea was already well established. The general idea behind email tarpitting was simple: identify a characteristic of Spam (for example, huge distribution lists) and rather than simply deny access, create programs that watched for that condition. When a Spam threshold was crossed (for example, after the 50th name on a distribution list), new code kicked in that kept the connection open but started introducing deliberate, escalating delays into the SMTP service. To the system sending the email, the SMTP host simply appears temporarily busy. Because the connection wasn't broken, and because the sending system continues to receive "Still busy" replies from the SMTP host, many automated Spam programs wait very patiently for the imaginary traffic to die down.

The next stage started with autonomous programs called "Spiders". Here's how the web site http://spiders.must.die.net/ describes them:

> **Spider?**
>
> Crawling around the Web now are a wide variety of robots known as "spiders". Their goal is to recursively scan every document available and make a condensed form of the data they collected available to whoever controls them.
>
> Many of these are Search Engines, which allow the general public to search the spider's index for specific items for "free". (Without the right tools you usually have to view their advertising, though.) On the whole, Search Engines are beneficial to have around.

> However a new species of spider seems to have made its way onto the Web lately that is less beneficial to the public. These particular bugs have a more sinister purpose in mind: They collect e-mail addresses so that their owners can send everyone unsolicited advertising. **No, thanks.**

One method of building a Spider tarpit is to add a special branch to a web site (marked with big warnings to stop real users from falling in). That branch is stocked with attractive phrases, provocative words and lots of "@" characters. As added bonuses, some tarpits are recursive, leading Spiders on infinite loops, and some are set to transfer pages in the tarpit very slowly. Once a Spider trips into that branch, it falls into long loops filled with what looks, to a Spider, like really good eats. In fact, the addresses and information are slow-acting poison: plausible but fake email addresses, legitimate addresses of senators and congresspeople, nonsense text filled with shocking and controversial key words, and so on. Some examples I ran into recently (in no special order) are:

- http://spiders.must.die.net/ (part of the die.net site)
- http://www.confusticate.com/ (choose the link, "tarpit, anyone?") and
- http://www.deepthought.org, which even drops a poisoned cookie on your system so that once you've visited the tarpit section, any attempt to return to the site drops you back in the tarpit.

I think the sites are entertainingly designed, but that may just be me.

The CodeRed and Nimda worms inspired Tom Liston to extend the idea of a tarpit to TCP/IP connections. These worms were flooding the Internet, and one key to their success was the speed with which they could scan for new systems. Tom Liston wrote CodeRedNeck as his first response[8], which, through some twists and turns, evolved into LaBrea (version 2.4 Beta 3 as of this writing).

Like other tarpits, the idea was to slow down unwelcome connection attempts without harming legitimate traffic. Mr. Liston approached this from an interesting direction. Here's a quote from the LaBrea.README file, taken from his post to the Intrusions list at www.incidents.org[9] (as quoted on the (http://www.threenorth.com/LaBrea[10] website):

> I'm pretty sure that most of you are using your allotted ip address space somewhat like I am. At any given time I'm using only about 20-30% of the ip addresses that I have available. What if I could put something on those other 80% of my ip addresses that would give "Code Red" something to play with that would slow it down to a crawl? A sort of "DoS" back at the worm.
>
> Since those "extra" ip addresses aren't actually expecting any inbound traffic, anything fired at them can safely be assumed to be "bad" traffic. If I

> wrote a little piece of software that sat on those ip's and listened on port 80, anything that it heard could safely be "played" with.[11]

The logic chain is simple. Systems that don't exist shouldn't be receiving stuff. Any traffic to nonexistent systems is, by definition, unwanted. Unwanted traffic is bad traffic. Administrators are free to fool the computers that send bad traffic.

The first step is for LaBrea to find unused Internet (IP) addresses on the local network. Instead of making the administrator provide static unused addresses, LaBrea finds unused addresses dynamically by watching unanswered Address Resolution Protocol (ARP) requests. Bear with me for a little bit and I'll explain. If a computer needs to find another computer on the same network, it broadcasts an ARP request. ARP requests are packets sent to all computers on the same segment that ask, "Who is using IP address <x.x.x.x>?" If a computer is using that address, it answers; if no computer is using that address, there's no answer, and it's possible the IP address may be unused. Modern switched networks complicate things a bit, so LaBrea includes the "-s" (safe) option, which makes LaBrea send out its own ARP broadcast for that same address just to make sure nobody's using it, (Practical tip: **always** use the "safe" (–s) option.) After LaBrea finds and verifies an unused IP address, it starts answering ARP requests saying, "Sure! Send packets for IP address <x.x.x.x> over here!" This is the first part of creating a ghost system. It does the same thing for every unanswered ARP request (except for the addresses and ports you tell it to ignore), building a list of available, unused IP addresses. None of these addresses should be receiving traffic.[12]

Once LaBrea has captured one or more addresses to use, the goal is to put convincing ghosts answering each address. Now, real computers react in certain ways to requests, so a ghost computer will have to make at least some of those same responses. Start with the basic TCP protocol for establishing a connection between computers: the *three-way handshake.* Here's a simplified description of the process:

1. The source computer that wants to establish a connection sends a TCP packet to the destination computer's address. That packet has the synchronization (SYN) bit in the packet header set to 1, and is called, amazingly, a "SYN packet."

2. The destination computer, if it's listening, hears the request (the SYN packet) and sends back a reply packet with both the synchronization (SYN) and acknowledgment (ACK) bits set. This is called a SYN/ACK packet.

3. The source computer receives the SYN/ACK packet and replies with an ACK packet, acknowledging the connection.

The whole handshake is measured in milliseconds, after which the two computers have established their connection and can begin the real conversation by exchanging data.

There's one more important detail: the port number[13]. Along with the Internet address, the first SYN packet will include the port number it wants to connect to on the destination computer. Different services run on different ports, and computers only start conversations on ports they're listening to – that is, when they offer a service on that port. Scanners work by sending packets to bunches of addresses (and sometimes bunches of ports), waiting for the few systems that answer.

LaBrea *mimics* the TCP handshake. If it sees a SYN packet, it sends back an appropriate SYN/ACK packet (and also specifies that subsequent packets should be very small). The originating computer then sends the last packet in the handshake (an ACK packet that LaBrea dutifully checks and ignores). At that point, the requesting computer has committed many other resources to the connection – computer time, memory for counters, buffers, etc., and it starts a countdown clock, waiting for the destination (tarpit) computer to start talking. For its side of the conversation, the LaBrea computer has committed… well, nothing, really. LaBrea doesn't keep track of the connection. It has rules for handling each packet as it comes in, and it just follows those rules. It doesn't remember anything, and it doesn't save anything for the next time – no commitment, no resources, and no strain.

What services does a LaBrea ghost reply to? All of them. When a request comes in to any port number, if that port number hasn't been specifically excluded, the tarpit always answers, "Yes." They're shameless that way.

For a basic tarpit, that's the end of things. It doesn't do any more. This is the IT equivalent of getting a telephone call from a telemarketer, saying, "Hello, could you wait for a second?" and putting the phone down. The requesting computer will wait a decent interval, send a few "Anyone there?" packets spaced out over time (which the tarpit ignores), and eventually the requesting computer drops the connection. While the timeout values vary between computers, most are measured in minutes, compared to the usual milliseconds for a handshake. Any scanner fishing for live systems will get very, very slow answers.

LaBrea can do even better, too. It has a marvelous option called "persistent connections." Think of it as "extra-sticky." After the handshake, the source computer waits for the tarpit to call back, but of course it never does. After a certain amount of time has passed, the source computer sends its "Anyone there?" query. If LaBrea is set for persistent connections, it promptly replies, "I'm still here – please wait," every time it receives a query packet. The source computer thinks the connection is good, it resets its timeout clock and starts waiting again. Every time the source computer queries the tarpit computer, it gets back a valid, if off-putting, answer. TCP isn't that smart a protocol – as long as it's receiving the proper responses within the times it expects, the connection is never broken. The tarpit can trap connections this way for days, weeks, or, according to the LaBrea site, even months.[14]

> **Side Note**: Because the Linux TCP/IP protocol stack handles packets more stringently than Windows. LaBrea has a special section just for

Linux. The technique is quite fancy, embedding information in the sequence number, but that's why some log entries specify "Linux" packets.

Although these "persist" connections are very small, there's a possibility that enough of them could cause network congestion. The Persistent option makes you specify a maximum data rate, above which it starts dropping packets. Our practical experience saw that maximum reached once, and then just barely, but it's a reassuring option.

LaBrea also mimics responses for SYN/ACK packets. On the surface, sending a computer a SYN/ACK packet doesn't make sense. The trick is that, if the receiving computer is listening on the port that receives the SYN/ACK packet, it responds with a Reset (RST) packet. This is a less conspicuous technique for finding active ports. LaBrea ghost systems answer a SYN/ACK packet with a Reset (RST) packet just like a real system.

Can packet forging be dangerous? Easily. In theory, LaBrea is mimicking non-existent (or unavailable) systems, so it's hard to argue that the requests to those addresses are legitimate. LaBrea also has a fairly flexible address and port exclusion mechanism, so it's possible to exclude whichever systems you choose. Good enough so far. Suppose, though, that LaBrea accidentally captured a critical system's address and began tarpitting all the local users who wanted that service? A runaway tarpit could easily bring your whole network down. That's why I was cautious about introducing the tarpit, and why I wanted to know how well it worked in my real, less unpredictable environment.

Implementation

LaBrea Program

Installing LaBrea was relatively easy. I chose to download the source code (http://www.hackbusters.net/Beta.html) and build it myself, although RPM files are also available. Call me old-fashioned. I built it on a testbed Red Hat[15] Linux 7.3 system I use for all manner of scans, detections and whatnot. The LaBrea documentation mentions that it needs the "libpcap"[16] and "libnet"[17] packages installed first, and after a false step or two, I got all the software compiled and built in the right order. Note that, unlike libpcap and libnet, LaBrea doesn't install itself, so you have to copy the single executable "LaBrea" file where you want it. I put it in /usr/local/bin.

Unfortunately, I didn't have a test environment capable of testing this kind of software. Gritting my teeth, I waited until late one evening when I knew nothing critical was happening and brought the software up on a live network. I set the logging to verbose (-v) and set the program to log to the screen (standard output) and not to detach (the –O option). That way I could watch what it was doing as it did it, and I could easily kill it at

the first sign of anything going wrong. I didn't attempt to do any reporting. The test system also has a fairly robust iptables firewall configuration, which I left enabled.

LaBrea started captured its first address almost immediately and started answering traffic. It looked good, but I didn't want to run it long without capturing the data, so I stopped the system and restarted it, this time redirecting the output to a log file. After another couple of minutes I killed the process and checked the logs.

I noticed some odd Server Message Block traffic (SMB is the protocol Windows uses). A system inside my network was being treated tarpitted. I also noticed that one of my router's IP addresses had been captured as an unused address, which shouldn't have happened.

My first goal was to create a stable, working prototype with known behavior that didn't risk killing my local networks. Once I had that, I knew I could slowly expand its behavior, but without that, I had nothing useful. With low risk in mind, I excluded the NetBEUI ports (137-139) and all local addresses from being tarpitted by adding these lines to the /etc/LaBreaConfig file:

```
137-139 portignore
<x.x.x.x>-<y.y.y.y> ipignore
```

[Note: The <x.x.x.x> and <y.y.y.y> were the starting and ending addresses of my local subnet. This was a wider exclusion than was necessary, but again, it limited the risk.]

For the next test I wanted to run the tarpit for about an hour, using syslog logging. I monitored the log using "`tail -f` <system log file>". The tarpit seemed itself looked benign, but the system log traffic was alarming. It started slowly enough, but as the tarpit began to harvest addresses and trap connection attempts, the number of packets it was handling (and subsequent logging) increased geometrically. After a few minutes I stopped the program and changed back to logging to a file because the volume was killing my system logs.

The LaBrea logging grows by megabytes – far too large to examine without filtering. Unfortunately, turning off verbose logging stops almost all the logging, and I wasn't willing to run a tarpit blindly, either, so I settled for big logs. Redirecting the output to a file is a good practice (I recommend something in the "/var/log" tree), but don't be caught unawares by how quickly that file can grow. When the tarpit started running continuously, it was handling a million transactions and creating 100 megabytes of log entries every day. Earlier I said the tarpit looked benign, and I know this doesn't *sound* benign, but according to my network monitor, all this activity only consumed around 1500 bytes per second (yes, *bytes*) of bandwidth and 2% of the system's CPU capacity. Not counting the huge logs, the tarpit system – and my network – was loafing.

Next, I started comparing the logs to what I was seeing on the IDS-equipped hosts, and that's when I noticed that the tarpit was missing some of the scans the IDS was reporting, MS SQL and FTP scans in particular. The tarpit system was still running the original iptables firewall that had been set to drop quite a number of "unwanted" packets, including MS SQL and FTP, so this wasn't a big surprise. I knew I'd either have to redesign the firewall or bypass it. Unfortunately, the test system I'd put LaBrea on was also running many other services to simply open the firewalls.

I needed to build a real Tarpit system, and just at the right time, an old desktop system fell into my lap. Such is fate. Knowing it might have to survive without firewalls, I wanted the system to be as hardened as I could easily get it. I chose to use Red Hat Linux again with the Bastille scripts (http://www.bastille-linux.org)[18]. I recommend their scripts, but if you're going to use Bastille, first check the most current operating system versions they support. Although other Linux dialects are god (OpenBSD coming to mind), I wanted to use Red Hat again because it's a dialect I already know, and I have the materials. Even though Red Hat 7.3 was available and stable, I chose 7.2 because at the time, the Bastille scripts hadn't been updated for Red Hat 7.3. I installed it with a basic configuration. I also included X and Gnome because I like them, although they add complexity and aren't necessary. I updated anything that needed it, made one last pass through the service list (I had missed one or two), and then ran the Bastille 1.3.0 scripts. I followed most of the defaults throughout the Bastille installation. In addition to the hardening, Bastille created its own iptables firewall script, but I had to turn it off, much though it pained me, because it interfered with detection. I'm still working on either moving the LaBrea program so that it receives packets before the firewall blocks them, or designing a set of firewall rules that protect the tarpit system while passing through the tarpit traffic. Until one of those is complete, though, the local firewall and LaBrea don't play well together.

As before, I installed libpcap, libnet and then LaBrea, setting it up as it was previously, including the same configuration file. I ran LaBrea and it acted like the previous installation, but this time, without the firewall, I was seeing many more connection attempts, including everything the host-based IDS systems had been reporting. After closely monitoring this system long enough to be sure it wasn't doing something untoward, I realized I needed better reporting tools to sift through the quickly growing logs.

Reports: LaBrea Reporter

LaBrea Reporter was next on my list. I had lots of raw logs and useful tools (if you don't use the UNIX grep command, now is the time to learn), but I needed better summaries of what was happening. I started with LaBrea Reporter (http://people.opera.com/sverrest/LaBrea/)[19], a script Sverre Stoltenberg wrote for LaBrea written in the Python[20] open-source scripting language. Python is new to me (as so many things are), but it isn't difficult to learn.

Easy was good. The LaBrea Reporter script didn't immediately work with LaBrea 2.4 beta 3. The log entry formats in this version weren't exactly what Reporter expected. The *offsets* – where the Reporter script looks in each log entry – weren't correct. I adjusted the offsets to fit the new formats and I'm very happy I did. The Reporter gave me summaries of the packet activity, the source hosts in order, and the destination hosts in order, along with summaries. These were critical reports, especially in the beginning. Here's part of one report (including my extensions):

```
LaBrea rapport:
===============

Start date:    Wed Jun 26 00:01:28
End date:      Wed Jun 26 23:55:05

Total transactions: 380634

New source hosts tarpitted this period: 105
               Number of target hosts: 1145
 New tarpitted connections this period: 12695
      Answered SYN/ACK & FIN/ACK scans: 32969       ...

Tarpit Target ports
===================
   21:     2
   22:   996
   25:     6
   80:  2676    ...

Source address
==============
   996 129.81.42.202   Wed Jun 26 01:07:08 - Wed Jun 26 01:08:32 EST
  1134 213.10.150.199  [ipd50a96c7.speed.planet.nl] Wed Jun 26 12:23:27 - Wed Jun 26 12:25:13
EST ...

Target address   Total    Port: Count   Port: Count   Port: Count
==============   =====   ============  ============  ============
 xxx.xxx.xxx.10:    13 |    22:     1     80:     2   1433:     7
xxx.xxx.xxx.101:    26 |    22:     2     80:    17   1433:     7   ...


This report is created with LaBrea-file.py, a variation
of LaBrea-stats.py. The latest version can be found at
http://people.opera.com/sverrest/LaBrea/

Information abount what LaBrea is can be found at
http://www.threenorth.com/LaBrea/
```

I've chopped this example report dramatically to get it to fit. The full report listed all the source and target addresses, which is important information.

It could be very bad if the tarpit captured a local address, particularly if it had captured the address of a critical system. The LaBrea Reporter gave me a summary of all the source addresses (the theoretical "bad guys") in address order. I could easily search for local addresses. If there are a substantial number of local systems being tarpitted, there's a real chance that your tarpit has captured an address it shouldn't. In my first run, I found a critical address (one of the routers) and legitimate local addresses in the report as "source addresses". I added exclusions for the critical addresses and used

search tools (primarily “grep”) to find out why local addresses were in the tarpit. This is how I found out that people are ignoring “This system is no longer available” messages for really extended periods of time.

One useful option would be a less verbose reporting option. The full logs are very large, and while that level reporting is good at first, another option would be helpful. “Persist” traffic, for example, fills over 90% of the logs and only shows that the tarpit has trapped connections. I’ve looked at modifying the code, but it’s still on my To-Do list.

Reports: LaBrea::Tarpit

My next goal will be to implement the LaBrea::Tarpit program (http://scans.bizsystems.net/)[21]. These Perl modules look very promising. They even start the LaBrea program from inside their own wrappers, and that seems to solve many of the reporting problems I was having. The logging is much more concise, and, in theory, it uses report modules to create various reports, including a report formatted for the Dshield.org (http://dshield.org/)[22] central reporting site, plus it saves LaBrea running information in cache files, which saves a lot of re-discovery when LaBrea restarts during log rotation. Unfortunately, I haven’t yet gotten the reporting side of LaBrea::Tarpit to work, but this remains high in my “To Do” list.

Conflicts, Problems and Changes

Conflict: Firewall

I’ve already mentioned in the Implementation section above that most firewall rules limit what LaBrea is allowed to mimic. The options are to run without a host-based firewall or to design firewalls that allow tarpit activity while protecting the tarpit system itself.

Conflict: DHCP

During the first sustained test, another engineer noticed that DHCP clients started getting “No address available” messages from the DHCP server. We ran a few more tests and got that same answer for each new system. A moment’s thought gave the answer.

Like many networks, our clients are a mix of static, reserved and dynamic IP addresses. To avoid IP address conflicts, we set the DHCP server to PING addresses before allocating them to avoid our users getting embarrassing address conflict messages. Suddenly, though, no matter which unused IP address the DHCP server chose, when it PINGed that address, it always got an answer.

I blushed, briefly, and added the DHCP server to the ipignore list.

Conflict: System Monitoring

We run system-monitoring software – WhatsUp™ from Ipswitch (http://ipswitch.com/)[23], although this applies to most system monitoring programs. These programs usually work by periodically checking a list of systems using a combination of PINGs, TCP packets and SNMP requests. When the tarpit started running, I noticed that the monitoring software started saying that systems I *knew* were really, indubitably and sincerely dead, had suddenly and inexplicably come back to life.

I eventually realized that the tarpit had swallowed the IP addresses of these demised systems and put up ghosts in their place, and the monitoring software simply believed the ghost answers.

Like the DHCP software, we excluded the monitoring system using "ipignore" and, so far as the monitoring system was concerned, the ghost systems evaporated. Similar problems would probably corrupt the results of any internal monitoring software.

Conflict: Port Locking

Many Enterprise-level switches and hubs allow "port locking". When a port is locked, it only accepts a connection from a specific, registered computer. Every network card uses a unique address burned into it at the factory called a Media-Access Control (MAC) address, and local networks really use those MAC addresses to that move packets. A port is locked when it's set to only allow traffic from one or more specific MAC addresses. This prevents people from plugging a foreign computer into your network successfully, and as a side benefit, it means people have to tell someone when they move a computer.

While a network card usually uses its own MAC address, nothing limits it to that. LaBrea creates ghost MAC addresses to match its ghost IP addresses, and sets the network card to listen to all traffic ("promiscuous mode"). This is important, since if someone scanning a network saw the same MAC addresses in replies from multiple IP addresses, it would be obvious they had hit a tarpit or a honeynet.

If port locking is used, that port will block all the tarpit traffic, since it's coming from unregistered MAC addresses, and hopefully report it as foreign. Make sure to have port locking turned off for the tarpit system.

Problem: Accidental System Capture

This isn't a problem we saw, but a possibility that rose in our minds during the system-monitoring problem I mentioned above. Suppose an important system – say, your authoritative Domain Name Server (DNS), if you run one – crashed for an hour. Most TCP/IP stacks have secondary DNS servers for just such an event, and the clients will switch to the secondary DNS when the primary doesn't answer promptly. Everything

keeps running automatically. Suppose, though, that after a minute or two, requests to the primary DNS server suddenly started getting valid – but really slow – acknowledgments? Every client DNS request would (eventually) time out, but this is a failure they'd be unlikely to handle gracefully.

This thought gave us chills. A LaBrea tarpit is *designed* to capture unused IP addresses, and if it isn't restricted, it captures every unavailable system's address. Any system that went down might be accidentally captured. This would never be a good thing for a critical system.

Populate the LaBrea configuration file with all the critical servers: name servers, authentication servers, routers, file and print servers, application servers, etc. Exclude them all. Don't leave a tarpit running unsupervised if you haven't excluded all of your critical addresses.

Change: FIN/ACK packets

Early in the tests, we noticed activity entries in the log that clearly showed someone scanning up through our addresses in order. When I turned on a packet-sniffer system (I used the Ethereal[24] package as it comes packaged with Red Hat), I found that the tarpit wasn't answering these scans because they were FIN/ACK packets (packets with both FIN and ACK bits set). This packet doesn't make official sense – there's no reason those bits would be set at the same time – but systems that hear them answer with a Reset (RST) packet, and the tarpit wasn't.

This wasn't a problem per se. LaBrea never said it mimicked every possible reaction. Still, when we started seeing thousands of FIN/ACK scans, I added a FIN/ACK section to the LaBrea source code, copying from the SYN/ACK code, adjusting it to answer FIN/ACK packets and adding appropriate messages. I freely admit that I did this somewhat blindly, not fully understanding the original code, and there were any number of other fixes also needed, but when I tested the changes using HPING2 (http://www.hping.org) to fashion my own FIN/ACK packets, the tarpit sent back appropriate RST packets.

Change: PING packet reporting

I knew the tarpit was answering PINGs – I'd deliberately run PING against tarpit addresses as a test – but when I checked the logs, there were no PING reports. I even added a PING section to the Reporter program, but the totals were zero in every report. I found a minor code defect. LaBrea answers ICMP PINGS correctly, but then it prematurely (if gracefully) bails out of the reporting section of the code. I've written the author to mention this problem and my less than graceful workaround.

Conclusions

Did I create a stable tarpit that's benign to my users? Yes.

Even though it's still Beta software, LaBrea has been stable, and the only changes I've made to that code were to extend its functionality and fix a non-critical reporting problem. With the proviso that it's important to do your homework about the addresses and ports to be ignored and/or excluded, the scheme LaBrea uses for capturing unused IP addresses appears to work as promised. As a side benefit, LaBrea and its reporting tools are very useful as a kind of network-based intrusion detection tool. I find I've been checking the reports daily.

Has our tarpit been successful at limiting scanners? Probably.

If scanning and hacking volume is any measure, the tarpit looks good. Activity started slowly, but after a few days of running, the tarpit was recording roughly one million transactions a day, including tarpitted connections, SYN, SYN/ACK and FIN/ACK scans, persist activity, additional activity and ICMP PING packets. By the start of the second week, the tarpit volume had dropped to under 400,000 transactions a day, and it's stayed below that total during the following two weeks. In fact, the volume may still be dropping, although the sample is too small to be significant yet. (No systems have been hacked in the last two weeks, either, for what that's worth.) Does this mean scanning is more limited? Not having access to the hackers and scanners, I can't say why the traffic changes with any certainty. Still, I suspect some people have chosen to stop scanning my addresses, which makes me think the tarpit has probably been at least somewhat effective.

Are internal users safer because of the tarpit? This doesn't have a clear answer.

I have no hard evidence so far (the test period is far too short for a valid sample), but I believe the ghost systems have at least provided some system protection through obscurity. Any vulnerable systems on my network are much harder to find in a field of convincing, slow ghosts. Unfortunately, because we don't have a body of intrusion detection data to compare against, I can't measure any improvements. I think the local users are realizing some level of protection, but I have no hard proof.

Is the tarpit protecting users outside my network? I believe so.

The tarpit seems to be successfully trapping – sometimes for long times – more HTTP connections than I'm comfortable thinking about. Although I'm shocked at the volume, the tarpit seems to be doing exactly what it set out to do: trapping unwanted connection attempts from dumb, automated programs. Does this make the Internet a better place? Yes, I think this goal is also successful to some extent, even if the exact contribution is hard to measure.

Will the hacker community adapt to tarpits? Probably.

> I expect to see scanning tools incorporate tarpit detection before too long, especially if tarpits become more popular. I also expect to see tarpits become more sophisticated in that same kind of arms-race development curve that virus and anti-virus software have been dancing for years. It's a new arena for an old pattern.

Finally, and perhaps the most important question, is a tarpit fun? Completely. They're small, undemanding, and everyone should build one.

References

"Bastille Linux" release 1.3.0. Bastille Linux development team. URL: http://www.bastille-linux.org/ (6/28/2002).

Beker, Jeremy. "confusticate.com". URL: http://www.confusticate.com (6/20/2002).

BlackICE version 3.5.cbq. Internet Security Systems, Inc. URL: http://www.iss.net (7/9/2002).

Carraway, Devin. "Peachpit -- a Tarpit for Censorware." URL: http://www.devin.com/peachpit/ (6/21/2002).

"Die.net" homepage. URL: http://www.die.net/ (6/20/2002).

"Dshield.org Distributed Intrusion Detection System" home page. Dshield is a trademark of Euclidian Consulting. URL: http://dshield.org/ (6/18/2002).

Ethereal network protocol analyzer version 0.9.3. Network Associates, Inc. URL: http://www.ethereal.com/ (6/22/2002).

"Global Information Assurance Certification." SANS Institute. http://www.giac.org (7/9/2002).

"HackBusters" homepage. URL: http://www.hackbusters.net (5/30/2002)

"Incidents.org" web site discussion list. SANS Institute. URL: http://incidents.org/intrusions/index.html (6/30/2002)

"LaBrea -- The Tarpit" web page. URL: http://www.hackbusters.net/LaBrea (5/30/2002) and mirror site URL: http://www.threenorth.com/LaBrea/ (5/30/2002).

Microsoft TechNet. Microsoft Corporation. URL: http://www.microsoft.com/technet/ (6/30/2002).

"McAfee AVERT Virus Information Library". Network Associates, Inc. URL: http://vil.nai.com/ (7/9/2002).

"NSA Security Recommendation Guides: Windows NT Guides." Security Recommendation Guides. National Security Agency. URL: http://www.nsa.gov/ (6/20/2002).

Python Scripting Language. Release 2.2. Python organization. URL: http://www.python.org/ (6/23/2002).

Redmood, John. "Stop Spam Now!" URL: http://john.redmood.com/stopspam.html (6/18/2002).

Robinton, Michael, and Bizsystems. "LaBrea::Tarpit" Perl module, release 1.03. URL: http://scans.bizsystems.net (6/23/2002).

SANS Institute. URL: http://www.sans.org (7/9/2002).

Schiffman, Mike D. "The PacketFactory" homepage 7/2/2002. URL: http://www.packetfactory.net/ (7/9/2002).

Stoltenberg, Sverre, "LaBrea Reporter script" release 1.10. 2001/09/25. URL: http://people.opera.com/sverrest/LaBrea/ (6/21/2002).

"tcpdump/libpcap" web page. 5/13/2002. URL: http://www.tcpdump.org/ (7/9/2002)

WhatsUp Gold. Ipswitch, Inc. URL: http://ipswitch.com/ (6/13/2002).

ZoneAlarm version 2.6.362.0. Zone Labs, Inc. URL: http://www.zonelabs.com (6/18/2002).

Endnotes:

[1] BlackICE is a trademark of Internet Security Systems, Inc., http://www.iss.net
[2] Intrusion Detection Systems are sometimes abbreviated as IDS, and subdivided into Host-based IDS and Network-based IDS (NIDS).
[3] ZoneAlarm is a trademark of Zone Labs, Inc., http://www.zonealarm.com
[4] "A Very Real and Present Threat to the Internet." Microsoft TechNet. Microsoft Corporation. URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/topics/codealrt.asp (6/30/2002).
[5] "Security Recommendation Guides: Windows NT Guides." Security Recommendation Guides. National Security Agency. URL: http://www.nsa.gov/ (6/20/2002).

[6] Carraway, Devin. "Peachpit -- a Tarpit for Censorware." URL: http://www.devin.com/peachpit/ (6/21/2002). Tarpitting is also sometimes called "teergrubing", after the German word for tarpit.
[7] Redmood, John. "Stop Spam Now!" URL: http://john.redmood.com/stopspam.html (6/18/2002).
[8] Liston, Tom. "LaBrea.README" file. LaBrea software. Release 2.4 Beta 3. 3/25/2002. Tom Liston's post to the Incidents.org list has long since expired, but it is extracted in the LaBrea 2.4 Beta 3 LaBrea.README file. Here's the quote from that file:

> The original concept for LaBrea started in response to the CodeRed worm. Our IP block was being scanned non-stop. I began to wonder:
>
> > "Is there anything that can be done to stop this worm?" Of course, many things came to mind, but most of them were illegal. Then, late one night, I thought, "Maybe I can't stop these machines from scanning, but I bet I can slow them down..."
>
> The following morning I posted to the INTRUSIONS list at incidents.org:
>
> > "I'm pretty sure that most of you are using your allotted ip address space somewhat like I am. At any given time I'm using only about 20-30% of the ip addresses that I have available. What if I could put something on those other 80% of my ip addresses that would give "Code Red" something to play with that would slow it down to a crawl? A sort of "DoS" back at the worm.
> >
> > Since those "extra" ip addresses aren't actually expecting any inbound traffic, anything fired at them can safely be assumed to be "bad" traffic. If I wrote a little piece of software that sat on those ip's and listened on port 80, anything that it heard could safely be "played" with.
> >
> > My hypothetical program should be pretty simple: you see an inbound packet at port 80 with SYN set, and you craft up a return packet with SYN/ACK set and perhaps an option to set the MSS to something small... say about 60 bytes. What does that do? Well, as far as the attacking worm is concerned, after replying with an ACK, it has completed a three-way handshake... it's connected. It's also been told to send information back to you in small chunks (to keep traffic to a minimum), which it dutifully does. The only problem is, my program just answers SYN packets and ignores everything else. So now the worm has to sit around while the whole TCP connection times out. I'm not sure what the timeout NT

is, but I think most stacks are pretty persistent about "good" connections, so it should hang it up for a good long time."

A few days later, Mihnea Stoenescu sent a message back to the list:

"Tom's concept works - I have a living proof.

For a few hours I've been teergrubing CodeReds via three-way Handshake on behalf of an entire C-block, by using only one host. At a rate of 6 hosts per minute hitting my block, I'm consuming circa 15 minutes of effective attack time every minute. A lot of hosts can be scanned in 15 minutes."

[9] Incidents.org web site discussion list. SANS Institute. URL: http://incidents.org/intrusions/index.html (7/9/2002)
[10] Liston, Tom. "Section 2: What Prompted You to Write It?" URL: http://www.threenorth.com/LaBrea/ (5/30/2002)
[11] Liston, Tom. LaBrea.README file. LaBrea software. Release 2.4 Beta 3. 3/25/2002.
[12] An ARP request is really asking for the Media-Access Control layer (MAC) address of a computer. IP addresses are used between networks and subnets, but inside a subnet, every packet is sent a MAC address. So that it's not obviously a tarpit, LaBrea creates a ghost MAC address for each ghost IP address.
[13] A "port number" is an additional address on top of the Internet Protocol (IP) address. Every Internet computer has 64K different "port" numbers it can use for different services. Although nothing is fixed – any service can be offered on any port – most services have default ports (HTTP on port 80, SMTP on port 25, etc.). If a connecting system wants a specific service from an unknown server, it connects to the computer using the default port for that service.
[14] "LaBrea::Tarpit HELD SINCE" web page. URL: http://www.hackbusters.net/cgi-bin/guests.cgi?captured.
[15] Red Hat Linux. URL: http://www.redhat.com (5/12/2002).
[16] Libnet beta version 1.1.0-b7, The Packetfactory. URL: http://www.packetfactory.net/
[17] "tcpdump/libpcap" web page. 5/13/2002. URL: http://www.tcpdump.org/ (7/9/2002)
[18] Bastille Linux. Release 1.3.0. URL: http://www.bastille-linux.org/ (6/28/2002).
[19] Stoltenberg, Sverre. LaBrea Reporter script. Release 1.10. 2001/09/25. URL: http://people.opera.com/sverrest/LaBrea/ (6/21/2002).
[20] Python Scripting Language. Release 2.2. URL: http://www.python.org/ (6/23/2002).
[21] Robinton, Michael, and Bizsystems. LaBrea::Tarpit. Release 1.03. URL: http://scans.bizsystems.net (6/23/2002).
[22] "Dshield.org Distributed Intrusion Detection System" home page. Euclidian Consulting. URL: http://dshield.org/ (6/18/2002).
[23] Ipswitch, Inc. WhatsUp Gold. URL: http://ipswitch.com/ (6/13/2002).
[24] Ethereal network protocol analyzer. URL: http://www.ethereal.com/ (6/22/2002).