;login:

THE MAGAZINE OF USENIX & SAGE
November 2000 • volume 25 • number 7

THEME ISSUE: SECURITY

edited by Rik Farrow

inside:

SCALPEL, GAUZE, AND DECOMPILERS

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

scalpel, gauze, and decompilers

Dissecting Denial of Service (DDos)

by Sven Dietrich

Sven Dietrich is a senior security architect for Raytheon ITSS at the NASA Goddard Space Flight Center. His focus is computer security, intrusion detection, the building of a PKI for NASA, and the security of IP communications in space.

<spock@netsec.gsfc.nasa.gov>

You walk into your office on a Monday morning and find that your usual game of Quake is painfully slow, like molasses in deep winter, and that your email inbox is scarily full with tons of messages from stricken users complaining about the network. Phrases like "the network is slow," "DNS is not working," and "I can't get to my Web site" ring in your ears as you retrieve your voicemail. Chances are, you are under a DoS (Denial of Service) attack.

What should you do? The most important goal is to determine whether you are in a simple DoS attack or a more complex DDoS (Distributed Denial of Service) attack, a distributed variant. The latter attacks began occurring on a large scale during the summer of 1999. Groups of intruders using massively automated compromise tools began infiltrating a large number of computer systems worldwide in late July and early August of that year. For what purpose, you may ask. The first sign of malignant behavior I encountered was in the form of a script that first mentioned a name that would go around the world, literally: trin00 a.k.a. Trinoo.[1] The script itself was copying in a program called leaf and placing it in a typically unsuspicious location on a UNIX system: /usr/bin/rpc.listen. For the inexperienced, this may appear to be a legitimate process taking care of RPC (Remote Procedure Call) services.

Well equipped with rootkits, the intruders would even hide that process on occasion and most likely would have gone unnoticed, except for one thing: the load level. Due to a flaw in the program, a cron job was launched every minute to keep the program, which came to be known as a DDoS agent, alive. The name leaf seemed to imply a tree, a leaf node of a larger structure.

The Scalpel

So what was the big deal? We all have seen DoS programs, whether they are UDP flooders, ICMP flooders, spider programs, Smurfers, or even the original Morris worm of 1988. What made this one different became apparent when I took out my electronic scalpel and started dissecting this piece of code: references to a "master server," IP numbers, and traces of crypted strings and commands. The electronic scalpel, of course, is your favorite binary or "hex" editor, but the UNIX strings command will also do in a pinch. Only a few weeks later, the next attack wave unleashed a packet storm known as the "University of Minnesota incident," in which roughly 250 systems bombarded the networks of the University of Minnesota and brought them to a screeching halt for three days.

What was going on? The packets were coming in so fast, they must have come seemingly from everywhere, at very high rates. The packets were very short, only 40 bytes in most cases, but the sheer rate at which they were aggregating at the target, the University of Minnesota, was simply overwhelming. So what was it really? Digging further into the code and tracing the packets to their apparent source led to the discovery of the "master server," a.k.a. the DDoS handler. This was the program "coordinating" the attack, ordering the DDoS agents to flood the target with packets. The intruders, counting on being discovered sooner or later, had taken precautions: The list of all the DDoS agents was encrypted, and in some cases the encrypted file was unlinked. Extreme precaution was needed to recover the full list of agents. As agents were discovered and shut down, new ones were being added in order to maintain the constant flood of packets. It was indeed a large attack structure. In a better-covered set of events,

the incidents of February 2000 involved several well-known e-commerce and industry sites as targets of DDoS attacks.[2]

So we return to the original question: What should you do? First and foremost: Start recording packets. Network (and computer) forensics have been and still are the key instruments in tracing back to the attacker. A simple tcpdump[3] process at your site border(s) will do for lack of a better choice. Then, and only then, start talking to your upstream Internet service provider. With high probability, the source IPs of the flood packets are spoofed, making determination of the actual source a bit harder at first. A good relationship with that provider is critical to getting a fast response, as floods can last anywhere from a few minutes to hours or sometimes days. Let us keep in mind that several scenarios are possible:

1. You are the victim.
2. You are the host for one or more DDoS agents.
3. You are the host for one or more DDoS agents and a handler.

The Gauze

Of course, variants and combinations of the above are indeed possible. In the case of 1, most likely you would like to restore your connectivity as soon as possible. Talking to your upstream providers should help locate the source of the floods, as they may have a better view of the flows of the (spoofed) packets. Of course, it is beneficial to get each intermediate provider to start recording packets for later inspection. Once one source is located, case 2 applies, as follows.

Now is the time for a tricky decision. Two types of traffic are involved in a DDoS situation, the flood traffic – felt heavily at the target, not necessarily at the source – and the very light control traffic. The control traffic is the interesting one, as it will be the path to the DDoS handler if no references to it can be found in the DDoS agent code. So what should one do? Shut down the DDoS agent system cold and subsequently make a binary snapshot, e.g., using dd, of the hard disk by transporting it to a different physical system? Maybe. Or should you take a snapshot while it is running? By experience it has been a good idea to "freeze" the system by doing the former.

There is really no simple answer, as it may be necessary to preserve the contents of memory also. Using lsof[4] and forensics tools such as The Coroner's Toolkit[5] are definitely a good idea, as things may not be as they appear. Of course, you are still recording packets at this point and any attempt at contacting the agent will be recorded in the logs, hopefully. The important aspect is to seep up as much evidence as possible with our electronic gauze, for later scrutiny. In any case, seek the advice of a DDoS foe, a CERT or FIRST member.

In some cases, law enforcement may choose to have its own idea of what to do in that situation and can advise you. That is outside of my domain and I shall not comment. For general guidelines, see the original CERT report[6], compiled by a small group of experts.

In the case of 3, you win! You are the lucky host of a DDoS handler and possibly of a few DDoS agents. Now it is doubly important to proceed with extreme caution in order not to destroy any evidence that may lead to the discovery of the entire list of agents, the handler and/or the agent code, the possible source code, and last but not least, the path to the actual intruder.

What should you do? First and foremost: Start recording packets.

The number of vulnerable systems is increasing exponentially.

So, you think, why not just remove the handler, reformat the drive of the computer, reinstall a more secure version of the operating system, and forget that it ever happened? Well, it is complicated. The intruders have considered that possibility and taken precautions for one or more "backup handlers" to take over the existing agents and continue their flooding. Think of it as a bad weed: Unless you get to the root of it, it will come back to haunt you. Taking a closer look at your network will help identify other, potentially dormant, agents. For scanning your network for well-known agents or handlers, see [7, 1, 2,]. Similarly, one can scan the host for well-known agents or handlers. Unfortunately, these programs are not always "well-known," and host integrity checking is an important asset in determining what files, if any, have been added or modified.

Of course, I have experienced quite a variety of "mishaps" during my quest for DDoS tools – from system administrators not wanting to surrender their tapes, not performing the correct backups, not responding altogether, or not acknowledging that their systems were compromised, to just reinstalling the operating system on the compromised host without prior backup. The good times, however, are finding the actual code for the tools, mostly in binary form.

Once one has the actual tool, how should one proceed? While it is often possible to replicate the traffic and provide substantial guidance to intrusion-detection system configurators, research prototypes and commercial variants, it is far more exciting to find the real code, the source, in order to discover potential flaws in the algorithm. These flaws are the mechanisms for registering flood or control traffic that might otherwise go unnoticed or dismissed as "noise." In the Shaft case[7] analysis of the agent source code revealed a fixed TCP sequence number for flooding. The "echoes" of attacks have been felt elsewhere in the world and are now signs of an ongoing Shaft attack. Yet other flaws may reveal mechanisms for shutting down floods in progress, such as improper authentication of handler-to-agent communications.

The Decompiler

In a different setting, reverse engineering of the binary executable may be the last resort at getting a deeper understanding, short of reading straight COFF or ELF executable binaries. On to the next ace up our sleeve: decompilation. In the case of Mstream[8], hand decompilation was essential to obtaining the attack algorithm, as no source code was immediately available, and informing the appropriate audience of our findings. So far, the decompilation process has been very much human, as the decompilation problem is hard and remains more of an art than a science.

Conclusion

As we lie in the aftermath of recent advertised and not-so-advertised DDoS attacks around the globe, we find ourselves still faced with the threat of ever-evolving DDoS tools. [9, 10] We need rapid incident-response teams, cooperative ISPs, good network forensics, good host forensics, well-established policies, and competent and DDoS-aware experts, all wrapped into one. While there are some good starting points for limiting the impact of such attacks, or even using traceback for identifying the source of the packet floods, [9, 10] that is outside of the scope of this article; the reader is referred to [2, 7] as starting points.

Nevertheless, the number of vulnerable systems is increasing exponentially, providing more hunting grounds for what apparently started as an IRC-channel takeover war. The fact remains: DDoS attacks should be our concern, as they represent a powerful tool to disable or incapacitate government, e-commerce, industry, and educational sites alike,

in some cases even the underlying infrastructure.[9] Thus far, intruders launching those attacks have successfully evaded detection through their cleverly built DDoS networks and are continuing to taunt us. A coordinated response and ongoing research will hopefully put us one step, or at least half a step, ahead.

REFERENCES

1. David Dittrich. The Trinoo Distributed Denial of Service Attack Tool. 21 October 1999.

2. Sven Dietrich's DDoS page. *<http://netsec.gsfc.nasa.gov/~spock/ddos.html>*.

3. *<ftp://ftp.ee.lbl.gov/tcpdump.tar.Z>*

4. *<http://vic.cc.purdue.edu/>*

5. Dan Farmer and Wietse Venema. The Coroner's Toolkit. *<http://www.porcupine.org/forensics/tct.html>*

6. CERT. Results of the Distributed Systems Intruder Tools Workshop. December 1999. *<http://www.cert.org/reports/dsit_workshop.pdf>*.

7. Sven Dietrich, Neil Long, and David Dittrich. Analyzing Distributed Denial of Service Tools: The Shaft Case. In Proceedings of USENIX LISA 2000, to appear.

8. David Dittrich, George Weaver, Sven Dietrich, and Neil Long. The "mstream" Distributed Denial of Service Attack Tool. May 2000. *<http://staff.washington.edu/dittrich/misc/mstream.analysis.txt>*.

9. Sven Dietrich, Neil Long, and David Dittrich. The History and Future of Distributed Systems Attack Methods. 5-minute presentation at IEEE Symposium on Security and Privacy, Oakland, CA. 16 May 2000.

10. Sven Dietrich. Dietrich's Discourse on Shaft (DDoS). Work-in-Progress presentation at USENIX Security Symposium 2000, Denver, CO. 17 August 2000.