Interested in learning more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Cheese Worm: Pros and Cons of a Friendly Worm

System administrators, and to some degree, the users of a computer system should be educated on malware risks. Administrators need to develop a heightened sense of security and a constant vigilance over the threats and available patches released to mitigate those threats. The system administrator becomes the gatekeeper, watching for any vulnerabilities of the system. It becomes a never-ending quest to stay one step ahead of the bad guys.

Copyright SANS Institute
Author Retains Full Rights

AD

Bryan Barber
GSEC v1.2e

Cheese Worm
Pros and Cons of a "Friendly" Worm

Malware is the current name for all of those ruthless pieces of code that are currently infecting various machines around the world. Malware consists of two different types of infections. The first is the virus. According to Symantec Anti Virus Research Center[1] (SARC), a virus is a computer program that is designed to spread itself from one file to another on a single computer. It does not try to spread itself to other computers. Viruses are typically spread by users trading files or sending emails to each other. The second type of malware is what is known as a worm. A worm, unlike a virus, tries to spread to other computers without human intervention. Once its creator sets it loose in the wild it will scan various computers on the network looking for a particular vulnerability that will allow it to break into that computer. When it discovers the vulnerability and penetrates the computer, it executes its code and perpetuates the process again. Therefore, worms can propagate much more quickly than a virus, causing destruction when left unchecked.

Recently, there was a worm written for the Linux operating system that was released on the Internet. It was called the Lion (or 1i0n) worm. The Lion worm looked for Linux machines running BIND version 8.2. This particular BIND vulnerability was first published by the Computer Emergency Response Team (CERT) in January 2001. Once it penetrated the computer, it installed the t0rn rootkit, which is a handful of trojaned programs to hide the existence of the Lion worm. It also set up a backdoor root shell on TCP port 10008. This would allow people to gain root access to the machine at a later date through that port.

Shortly after the Lion worm was released there was a report of a new worm that was released to get rid of Lion. The theory behind this new Cheese worm was that it would be a "friendly" worm that would scan machines looking for ones that were infected with the Lion worm. According to the

[1] Symantec

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

incident report posted at CERT[2], the worm looks for root shells installed on TCP port 10008 and infiltrates the machine through that port. Once the machine is infiltrated, the worm installs itself into the /tmp/.cheese directory. The cheese script is then run silently from that directory. The first thing that is does is change its process name to httpd. This will disguise the worm as a normal web server process. It then reads /etc/inetd and deletes any lines that have /bin/sh, which is the Bourne shell that the Lion worm setup to listen on 10008. Inetd is then restarted twice, once with /usr/bin/killall and once using /bin/killall. The effect of this is that all /bin/sh processes are killed, not just the one listening on 10008. The cheese process will continue running until it is killed somehow. During this time, it is scanning semi-random Class B networks, where the first octet of the address is between 193 and 218 and the second octet is between 1 and 254, with the psm program looking for other machines to "fix". When it finds a host listening on port 10008, it sends the cheese.uue file to the machine. It then uudecodes the file, which decodes the file from a text file into a tape archive file (tar). It then untars it to the /tmp/.cheese directory, which extracts the files from the archive. At this point the worm process starts all over again.

While the "friendly" worm is a relatively new concept, the anitvirus virus or "friendly" virus has been around for about a decade. According to Aled Miles[3], the first "friendly" virus was discovered in 1988 and named Den_Zuko. The intention behind it was to detect and remove the first PC virus, Pakistani Brain. There have also been other "friendly" worms. According to Michael Delio[4], one of those worms was written by Max Butler. The purpose was to eradicate the ADM worm by downloading and patching the security hole in the BIND software. Unfortunately, it also installed back doors on all of the systems that it "fixed". Butler, who was working as an FBI informant at the time, released it to the military and defense computer networks in 1998. Once the authorities discovered who had released the worm, he was arrested and is now serving a 12 to 18 month prison sentence.

---

[2] CERT
[3] Miles
[4] Delio

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

Pro Viewpoint

One of the viewpoints that is being advocated currently is that these "friendly" worms can be a good thing. According to Cyrus Peikari, a medical doctor and CTO of VirusMD Corp., "It's no longer sufficient to immunize PCs - basically we need someone to immunize the Internet as a whole"[5]. Peikari believes that a governing body needs to be in charge of releasing various worms to clean up the Internet after an outbreak. He likens this to the current method of distributing human vaccines and medicines. The thought is that if a governing body were to release the worm it would cut down on the paranoia concerning the fact that untested code could be part of the "friendly" worm. Peikari also points to the fact that most people do not update their virus software as frequently as they should, which leads to higher rates of infections.

Con Viewpoint

The other side of the coin is that there are so many different configurations, just in the Linux arena alone, that the whole idea seems to be impractical. For instance, some people may be running inetd to control their Internet services and some may be running xinetd, which is a more secure version with a different type of configuration file. Different worms will be needed for each configuration. Even those machines that are running the same daemons may have different configurations. Every system administrator tends to have their own way of setting up their machines.

One of the criticisms of the Cheese worm was that it was scanning all of the machines and generating large amounts of traffic on networks. If the cheese worm were to "infect" a couple of machines on a network and then start scanning that network for more machines to fix, it could possibly generate a Denial of Service (DoS) of that machine or even the whole network. One of the basic tenets in the networking world is to try and cut down on the amount of broadcast traffic. The reason for this is that every machine on the network must process the broadcast traffic, thus wasting CPU cycles. This also causes problems in Ethernet networks. Ethernet networks work off the Carrier Sensing Multiple Access/Collision Detection (CSMA/CD) algorithm. This means that a machine will test the network

[5] Costello

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

to see if any other machines are communicating at that particular time. If so, then they will wait before trying to send their packets. If there is a constant stream of broadcast traffic on that segment, the other machines cannot communicate with the network. If nothing else, the chance of sending a packet would be greatly reduced making it hard for the users to get their work done.

The original creator has no control over the worm once it is released into the wild. The most famous example of this was the Morris worm that was released to the Internet in 1988 by Robert T. Morris Jr., which shut down most of the Internet[6]. The problem with the Morris worm was that there was a minor programming bug that caused denials of service to most of the machines on the Internet at that time. Even once the problem was discovered there was no pragmatic way to shut down the worm. Viruses and worms tend to be unpredictable once released into the wild. What reassurance do people have that it will clean every machine on the Internet? If there were even one site blocking the "friendly" worm at their firewall and they had an infected machine, the "friendly" worm would not be able to penetrate and the cycle would start all over again. The infected machine at that site would continuously infect other unpatched machines, never able to get cleaned. It would be almost impossible to eradicate the virus completely. At this point the only fix would be for the system administrator to physically remove the worm from their machine and implement the patch for the insecure software.

One of the biggest criticisms of the "friendly" worm is that before a patch is implemented on a production machine, most administrators will want to test the patch on a test machine in case it interferes with any of the current production applications. Even if the worm is open source and many people have had the chance to review the code, there is still always the possibility that one little change that the worm fixes would be what interferes with a crucial production application. This could result in downtime and lost revenues. Historically, we have seen vendors release patches to the Internet that are supposed to fix security holes and the patch has either opened new holes or rendered the machine unusable. Most administrators would at least like the option to implement the patch or not.

[6] Boettger

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

In addition, most sites have some sort of security configuration document for quality assurance purposes. The main theory behind quality assurance is that if there were ever any problems with your machine, you could rebuild it to the exact state before the problem by following the procedures and applying the patches that are listed. If that site would have to rebuild the machine, there is no guarantee that those administrators would know to apply the proper patches that the "friendly" worm installed.

Even if all of these criticisms were addressed, the time that it would take to produce the "friendly" worm, have it reviewed by the open source community, and released by an international governing body would be an eternity to those sites infected by the bad worm. By the time that all of these steps are taken a worm could have infected a large number of machines. The number of infections happens at an exponential rate until there are no more machines to infect.

Conclusion

A "friendly" worm seems like a good idea to most uninformed people, but as I have shown the downsides far outweigh any good that it might do. It is definitely an impractical option. Not only would it require a central repository to distribute the worm and keep up with new infections, but it would also require the confidence of system administrators that would have to ultimately trust the integrity of the patches. Unfortunately, that group of people tends toward the paranoid and cautious side of things and would rather not have unknown code running on their systems. Any code that modifies system settings and the functioning of the operating system directly affects other programs. If even one program relies on what is patched then the worm has caused damage, which will result in lost time troubleshooting the problem and ultimately fixing it. The patch will have to either be removed and system restored or the program will have to be modified, which could take months. Most people seem to believe that it is impractical for an international governing body to be able to produce the fixes and release the "friendly" worm to the Internet in a timely manner. The answers lay in the education of the system administrators, and to some degree, the users of the system. Administrators would need to develop a

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

heightened sense of security and a constant vigilance over the threats and available patches released to mitigate those threats. The system administrator becomes the gatekeeper, watching for any vulnerabilities of the system. It becomes a never-ending quest to stay one step ahead of the bad guys.

© SANS Institute 2001, Author retains full rights

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

Bibliography

1. "Learn More About Worms and Viruses", 20 Oct 2000, URL: http://www.symantec.com/avcenter/reference/worm.vs.virus.pdf (23 June 2001)
2. Houle, Kevin. "CERT Incident Note IN-2001-05", 17 May 2001, URL: http://www.cert.org/incident_notes/IN-2001-05.html (25 July 2001)
3. Miles, Aled. "Bug Watch - Beware the Friendly Virus", 29 May 2001, URL: http://idg.net/ic_650989_1773_1-3921.html (1 July 2001)
4. Delio, Michael. "A 'White Hat' Goes to Jail", 22 May 2001, URL: http://www.wired.com/news/politics/0,1283,44007,00.html (26 July 2001)
5. Costello, Sam. "A Virus to Fight Viruses", 16 July 2001, URL: http://www.vnunet.com/News/1122084 (20 July 2001)
6. Boettger, Larry. "The Morris Worm: How it Affected Computer Security and the Lessons Learned by it", 24 December 2000, URL: http://www.sans.org/infosecFAQ/malicious/morris.htm (20 July 2001)

© SANS Institute 2001, Author retains full rights

Last Updated: May 21st, 2013

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Malaysia @ MCMC 2013	Cyberjaya, MY	Jun 03, 2013 - Jun 08, 2013	Live Event
SANS Pen Test Berlin 2013	Berlin, DE	Jun 03, 2013 - Jun 08, 2013	Live Event
Industial Control Systems Security Training - Houston	Houston, TXUS	Jun 10, 2013 - Jun 15, 2013	Live Event
Security Impact of IPv6 Summit 2013	Washington, DCUS	Jun 14, 2013 - Jun 16, 2013	Live Event
SANSFIRE 2013	Washington, DCUS	Jun 14, 2013 - Jun 22, 2013	Live Event
SANS Canberra 2013	Canberra, AU	Jul 01, 2013 - Jul 13, 2013	Live Event
Digital Forensics & Incident Response Summit 2013	Austin, TXUS	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS London Summer 2013	London, GB	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS Rocky Mountain 2013	Denver, COUS	Jul 14, 2013 - Jul 20, 2013	Live Event
SANS Mumbai 2013	Mumbai, IN	Jul 22, 2013 - Jul 27, 2013	Live Event
SEC528 SANS Training Program for the CompTIA® New Advanced Security Practitioner (CASP) Certification	Washington, DCUS	Jul 22, 2013 - Jul 26, 2013	Live Event
SANS San Francisco 2013	San Francisco, CAUS	Jul 29, 2013 - Aug 03, 2013	Live Event
SANS SEC 560: Network Penetration Testing @ Bangalore 2013	Bangalore, IN	Aug 05, 2013 - Aug 10, 2013	Live Event
SANS Boston 2013	Boston, MAUS	Aug 05, 2013 - Aug 10, 2013	Live Event
Critical Security Controls Summit	Washington, DCUS	Aug 12, 2013 - Aug 18, 2013	Live Event
Industrial Control Systems Security Training - DC	Washington, DCUS	Aug 12, 2013 - Aug 16, 2013	Live Event
SANS Thailand 2013	Bangkok, TH	Aug 19, 2013 - Aug 31, 2013	Live Event
SANS Virginia Beach 2013	Virginia Beach, VAUS	Aug 19, 2013 - Aug 30, 2013	Live Event
Mobile Device Security Summit 2013	OnlineCAUS	May 30, 2013 - Jun 06, 2013	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced