Notes on Linux/Xor.DDoS

Archived: 2026-04-05 17:44:08 UTC

In this post we'll be focusing on a certain kind of malware: **Linux/Xor.DDoS** (also known as DDoS.XOR or Xorddos). As usual, we'll break the post down in several points:

Background
Diagnosis
Analysis
Disinfection
Prevention
Conclusion

The variant discussed in this blog post is an older variant, so certain infection mechanisms may have changed, as well as C&C's. The point of this post is to familiarize yourself with ELF malware in a better way - how to diagnose, analyse, remove and finally prevent malware from infecting your Linux machines. A lot of malware is going around and it's not (all) exclusively for Windows machines.

Background
You may have heard about Linux/Xor.DDoS already, a Linux Trojan with rootkit capabilities (belonging to the category of 'ELF malware'). What exactly is an ELF file? According to Wikipedia:

> *In computing, the Executable and Linkable Format (ELF, formerly called Extensible Linking Format) is a common standard file format for executables, object code, shared libraries, and core dumps.*
> Source

In other words: ELF is to Linux as PE (.exe, .com, .scr, ...) is to Windows and Mach-O to OS X.

There's a nice mini poster available by Corkami as well:

More information about the ELF format can also be found at the **Resources** section.

If you haven't heard about Linux/Xor.DDoS itself already, be sure to read the initial post by MalwareMustDie uncovering this malware:
Fuzzy reversing a new China ELF "Linux/XOR.DDoS"

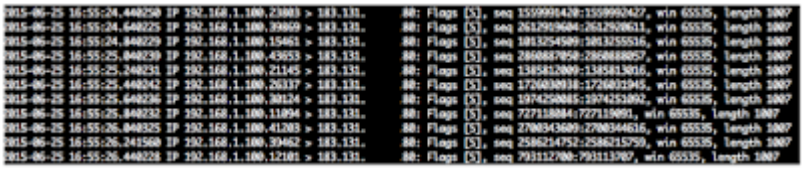
In short: Xor.DDoS is a multi-platform, polymorphic malware for Linux OS and its ultimate goal is to DDoS other machines. The name Xor.DDoS stems from the heavy usage of XOR encryption in both malware and network communication to the C&Cs (command and control servers).

There have been other write-ups about this malware as well, which will be mentioned throughout this article or referenced in the Resources section.

Diagnosis

How do you know you're infected with Xor.DDoS?

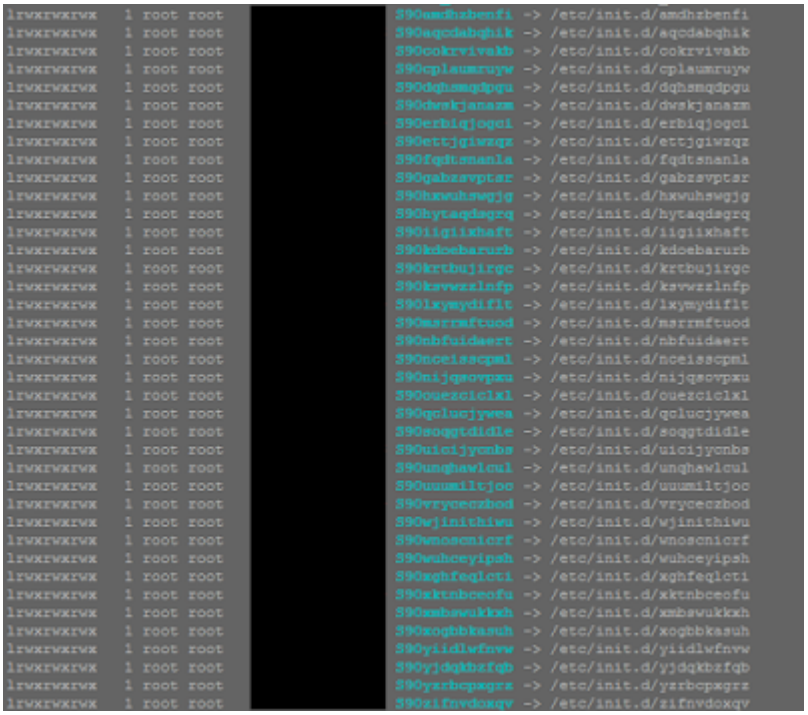
First and foremost (and obviously), you'll be conducting DDoS attacks from your machine(s) to targets chosen by the malware authors.



Sending of large SYN packets (Source)

You may use netstat to print any current network/internet connections. Use tcpdump to get a more detailed analysis of which packets you are sending out.

Secondly, another indication is seeing processes running with random names and sudden new executable files created in **/etc/init.d/** or **/usr/bin/** (see example below). New entries will be/are added to your crontab as well (/etc/crontab).



Malware running and its related files

You may use any command based on top or on ps to check for running malicious processes. We will see more in the Disinfection part of this blog post.

Thirdly, if you are running the standard OpenSSH server you may see an unauthorised but successful login and immediate logout afterwards.

These symptoms should be very clear, even more so if you've already implemented several measures to protect yourself from potential intruders. If not, then it'll be harder to track the infection origin as well. (but more often than not the SSH credentials of the root users are brute forced.)

To ensure your machines will not get pwned, be sure to read the Prevention part of this blog post.

Analysis
First off, we have to identify

how

the malware entered the system. Usually, a weak root password is used (like *admin* or *123456*, see here for a list of tried passwords. *Note: huge .txt file!*) or the attackers are brute forcing their way in. (brute forcing the SSH credentials of the root user) Another, but less common possibility, is exploiting a vulnerable service that you have running (Apache for example).

This figure is an excellent visual representation on how it all happens:

This variant copies itself over to **/lib/libgcc.so**, then creates a copy in /etc/init.d and a symbolic link to /usr/bin. Afterwards a new cron script is created and added to the crontab.

We will now take a look at one of the samples created - named ***bmtsfnlgxu***.
(SHA1: **b34b6f0ec42a0153c043b0665ec47bf6e5aac894**)

Easiest way on Linux is to just use the "file" command:

```
remnux@remnux:~/samples$ file bmtsfnlgxu
bmtsfnlgxu: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, for
 GNU/Linux 2.6.9, not stripped
```

We can see it's an ELF 32-bit executable for i386 - and it's

not

stripped.

Why is that last part important? strip allows you to remove symbols and sections from choosen files, which in turn makes it harder to reverse engineer (disassemble) as well. In this case, the file doesn't seem to be stripped, great! For example, we can see the source files and get an idea of what this malware does:
(this will also be shown later on in the video below, using IDA)

```
; Source File : 'crtstuff.c'
; Source File : 'autorun.c'
; Source File : 'crc32.c'
; Source File : 'encrypt.c'
; Source File : 'execpacket.c'
; Source File : 'buildnet.c'
; Source File : 'hide.c'
; Source File : 'http.c'
; Source File : 'kill.c'
; Source File : 'main.c'
; Source File : 'proc.c'
; Source File : 'socket.c'
; Source File : 'tcp.c'
; Source File : 'thread.c'
; Source File : 'findip.c'
; Source File : 'dns.c'
```

Moving on, we will start by using readelf for some further investigation of the file. We know, thanks to the **file** command, it's an ELF 32-bit executable for i386. Using readelf and parameter **-h** we will be able to gather more information:

```
remnux@remnux:~/samples$ readelf -h bmtsfnlgxu
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x8048110
  Start of program headers:          52 (bytes into file)
  Start of section headers:          545660 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         5
  Size of section headers:           40 (bytes)
  Number of section headers:         28
  Section header string table index: 25
```

This gives us more information already, for example; the magic (**7F 45 4C 46** for ELF files, 4D 5A for MZ files) 2's complement, little endian, the exact type of the file (an executable; other types for ELF files may be a relocatable file, a shared object, a core file or processor specific) but most importantly here being the *Entry point address*, or the start of the program.

In regards to readelf, using parameter **-a** we can dump a ton of information, you can find the output of this command on our malware on Pastebin: Xor.DDoS - "readelf -a" output

Note that VirusTotal has added (since November 2014) detailed ELF information in reports as well, which is more or less similar to readelf's output.

To disassemble the file, we can use objdump which allows us to disassemble only those sections which are expected to contain instructions (**-d** parameter) or to disassemble the contents of all sections (**-D** parameter).

However, to dive a bit deeper into the malware code, we will be using IDA, a multi-processor disassembler and debugger and Radare, a well-known (portable) reversing framework. Note that it will still be a quick glance, as

MalwareMustDie has already reported extensively about it as well [1][2][3][4]. Note also that it's always a good idea to analyse malware in a virtual environment (VM).

We will be using both tools on Windows, but you can just as easily run them on Linux or Mac.

I've made an instruction video on how to use IDA Pro Free to take a quick peek into the file discussed:

Download IDA Pro Free for Windows from here. If you're interested in working more with IDA, there's a handy list of IDA plugins available here.

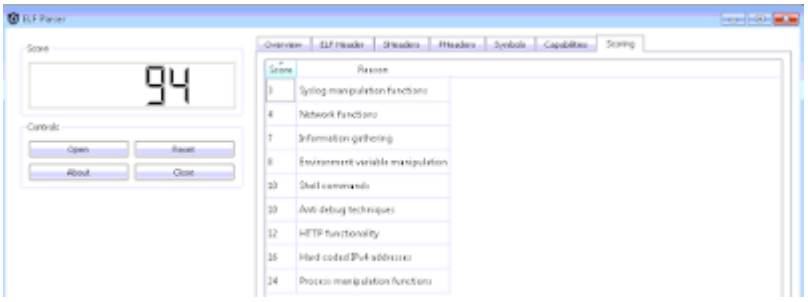
... And just the same for Radare, where we will discover a bit more - namely the C&C of the malware:

Download radare2 for Windows from here. More documentation about Radare can be found here. There's also a handy cheat sheet available here.

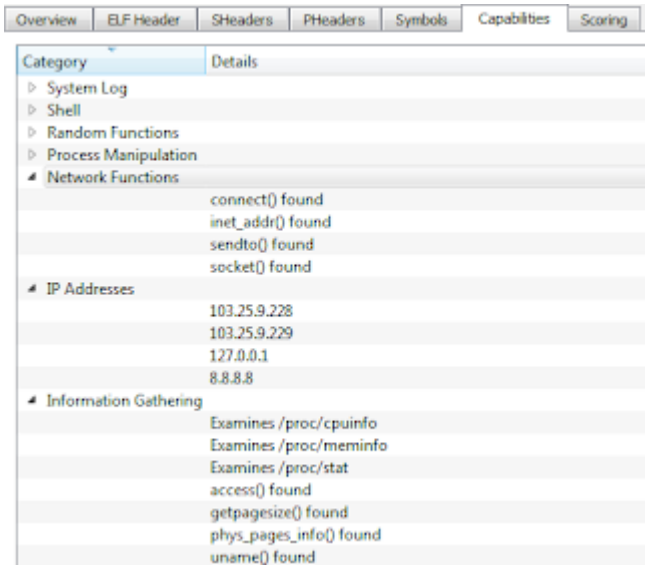
Note that the Xor.DDoS variant discussed in this blog uses 2 XOR keys for its (network) communication, they are the following:

- BB2FA36AAA9541F0
- ECB6D3479AC3823F

If you like GUIs, then I have another useful utility: ELFparser. It will perform a scoring based on several factors, such as shell commands, HTTP functionality and process manipulation. For example, for our file:



You can see it's scored pretty highly. I wonder what it has to say about the hardcoded IP addresses..:



You can also see 8.8.8.8, Google's DNS server and likely used to resolve the C&C domains

Great, it was able to extract our C&C servers:
103.25.9.228 - VirusTotal - IPvoid - DomainTools (whois)
103.25.9.229 - VirusTotal - IPvoid - DomainTools (whois)

Using ELFparser you can also look at the ELF header, sections, but also all of its capabilities like Information Gathering and Network Functions for example. It's a handy second-opinion tool.

Finally, one last tool which should not be missed when analysing ELF files: a sandbox. We will be using detux, a multiplatform Linux sandbox.

DNS Queries

Type	Query	Response
CN	www.wangzongfacai.com	wangzongfacai.com
A	wangzongfacai.com	174.139.106.51
A	gh.dsaj2a1.org	66.102.253.30
CN	www.wangzongfacai.com	wangzongfacai.com
A	wangzongfacai.com	174.139.106.51
A	gh.dsaj2a1.org	66.102.253.30

Connections to wangzongfacai.com and dsaj2a1.org

You have Network Analysis (IPs connected and DNS queries) and Static Analysis (Elf Info and Strings). In our example we have connections to *wangzongfacai.com*, not an unfamiliar domain. View the complete report made by Detux on our file here.

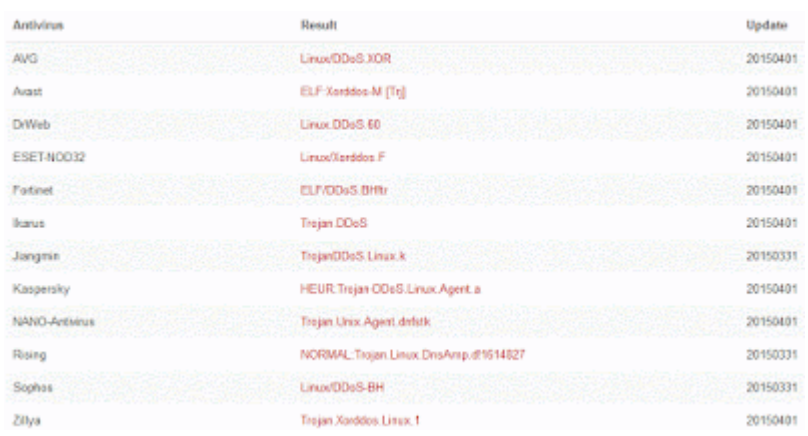
It's worth noting that several months ago, I already sent a file to Detux (and VirusTotal) which yielded similar results:

DNS Queries

Type	Query	Response
A	info.3000uc.com	23.234.60.140
A	ndns.dsaj2a.com	192.126.126.64
A	ndns.hcxiaoao.com	103.240.141.54

*3000uc.com, another familiar player - and again dsaj2a**X***

Detux report of that file here. When I sent the latter file to VirusTotal several months ago, it only had 12 detections, after re-submitting it had 19 detections. That's better but we're still not there.



Antivirus	Result	Update
AVG	Linux/DDoS.XOR	20150401
Avast	ELF:Xorddos-M [Trj]	20150401
DrWeb	Linux.DDoS.60	20150401
ESET-NOD32	Linux/Xorddos.F	20150401
Fortinet	ELF/DDoS.BH!tr	20150401
Ikarus	Trojan.DDoS	20150401
Jiangmin	TrojanDDoS.Linux.k	20150331
Kaspersky	HEUR:Trojan-DDoS.Linux.Agent.a	20150401
NANO-Antivirus	Trojan.Linux.Agent.dvfstk	20150401
Rising	NORMAL:Trojan.Linux.DnsAmp.df1514827	20150331
Sophos	Linux/DDoS-BH	20150331
Zillya	Trojan.Xorddos.Linux.1	20150401

Just a visual representation of detection difference. Read this for info.

You may find an overview of all gathered files as well as most common/recurring domains and their IPs they connect to/download from here, available via AlienVault's OTX.

That's it for our Analysis section, let's move on to Disinfection.

Disinfection

Most importantly, you'd of course like to remove/disinfect this malware completely. Some pointers:

- **Identify malicious processes**: run **ps ef** (ps stands for process status) to see which processes are running. Alternatively, you can use **top** or again ps with other parameters, for example **ps ej** or **ps aux** for a more complete, human readable table. Look for processes with random names; in our example it started with *S90* and random letters afterwards, linked to files with all random names, as is the case in our example malware named *bmtsfnlgxu*.

  Once you've identified the malicious process(es), you can use the following command to find related files as well: **for pid in $(ps -C -o pid=); do ls -la /proc/$pid/fd; done**
  Where is the name of the suspicious process. This command will display any open, related files. For example, for *bmtsfnlgxu* it would be:
  for pid in $(ps -C *bmtsfnlgxu* -o pid=); do ls -la /proc/$pid/fd; done

- **Identify malicious files**: look for newly created files in **/etc/init.d/**, **/boot/** and **/usr/bin/**. Again, look for files with random names. You may also use the command **ls -lat | head** to view recently changed files.

  Check your crontab (**/etc/crontab**). Delete the malicious cron jobs, more specifically the cron.hourly jobs and in the case of Xor.DDoS they will be the following:

  */3 * * * * root /etc/cron.hourly/cron.sh*
  */3 * * * * root /etc/cron.hourly/udev.sh*

  Delete these two lines from your crontab. Don't forget to save. Delete the related files, located in **/etc/cron.hourly**. In our case, their content was as follows:

  cron.sh

```
#!/bin/sh
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/usr/X11R6/bin
for i in `cat /proc/net/dev|grep :|awk -F: '{print $1}'`; do ifconfig $i up& done
cp /lib/libgcc.so /lib/libgcc.so.bak
/lib/libgcc.so.bak
```

  udev.sh

```
#!/bin/sh
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/usr/X11R6/bin
cp /lib/libgcc4.so /lib/libgcc4.4.so
/lib/libgcc4.4.so
```

As said earlier, delete these files manually, as well as the file(s) mentioned in the scripts. (in this case: **/lib/libgcc.so.bak**, **/lib/libgcc.so** and **/lib/libgcc4.4.so**.) Note that these files are

not

related to GCC's runtime library and thus can be safely deleted. It's just another way how the malware tries to hide itself.

Also double-check there are no malicious files or scripts in **/etc/rc.d**. If so, remove them as well.

- **Stop and kill malicious processes**: identify the parent process; usually it will be the one consuming the most CPU (which you can verify using any of the earlier commands, *top* being the easiest). Firstly, be sure to stop the parent process and wait for the child processes to die. Use the command: **kill -STOP $pid**

  When the child processes are dead, kill the parent by using: **kill -9 $pid**
  Note: in case you see any other malicious processes, go through the last 2 commands again.

- **Delete any leftover malicious files**: locations where the malware may reside have been indicated before, but to be complete:

  > / (*root* directory, in rare cases)
  > **/bin/**
  > **/boot/**
  > **/etc/init.d/**
  > **/etc/rc.d**
  > **/etc/rcX.d** (where X is a number)
  > **/lib/**
  > **/lib/udev/**
  > **/sbin/**
  > **/tmp/**
  > **/usr/bin/**

That's it. Some additional tips and tricks:

- Use **rm -rf** to permanently remove a file. Be careful with this command.
- Having troubles removing a file? Are you root? If not, try killing a process or deleting a file using root by prepending **sudo** before your command. For example: *sudo kill -STOP $pid*
- Malicious process keeps coming back? Go over the steps again, but this time note down where the malware resides. Make that directory and its files unmodifiable by making use of the chattr command. For example, malware is being recreated in */usr/bin/*. Use the command: **chattr -R +i /usr/bin/** Then, stop the parent, wait for the children to die and kill the parent. Remove the files. Don't forget to use chattr again after you cleaned the infection. (in our example: chattr -R -i /usr/bin/)

  It's also possible the malware is temporarily storing files into **/tmp/** while you are trying to kill its processes. When that happens, use the same chattr command on the /tmp/ directory and start over. If you are in doubt, use that chattr command on all aforementioned directories and start over.

  Very important: do not forget to use chattr -R -i on them afterwards!
- In rare cases, the attacker may still be connected to your box. If possible, cut the internet connection and go over the disinfection steps. If this is not possible, firstly stop SSH by entering the command:
  sudo /etc/init.d/ssh stop

Then, use iptables to drop any connection to the IPs the malware is connecting to (use netstat for example, see also Diagnosis) and to drop any connection from the attacker or cybercriminal. How to do this:

In our example, we learned that our C&C's were 103.25.9.228 and 103.25.9.229. Thus, type or copy/paste these 2 commands:
iptables -A OUTPUT -d 103.25.9.228 -j DROP
iptables -A OUTPUT -d 103.25.9.229 -j DROP

To block connection(s) from the attacker (you can find the attacker's IP using **netstat** for example):
iptables -A INPUT -s $attackerIP -j DROP

Don't forget to save your freshly created iptables rules by using the command
/etc/init.d/iptables save

Afterwards, change all passwords. (SSH, your user, root)

Best case scenario here is obviously:

- restoring from a backup
- if the machine is virtual, restore to a previous snapshot

When you have either of these available, don't forget to change all passwords afterwards to prevent re-infection - and patch your machine(s)!

Some Xor.DDoS variants may also incorporate a rootkit. In that case, hope you have a "best case scenario" available to you. Once a box is fully compromised, it may be hard to reinstate it back to normal or its original state.

For double-checking for rootkits and other malware, you may want to check out chkrootkit or alternatively, rkhunter. Additionally, you may download and install an antivirus, for example ClamAV.

If you perform manual clean-up as indicated above and have confirmed all is in order again, you can install ClamAV and perform an extra scan to be sure. Better be safe than sorry. Then, follow the prevention tips below to stay safe.

Prevention

- Use strong passwords for SSH or use keys instead of passwords for authentication. You can read how to do that here. In the unlikely event of you not needing SSH to a particular machine, disable it on that machine by:
  sudo apt-get remove openssh-server

  To disable it from starting up you can use:
  update-rc.d -f ssh remove

- Don't open the incoming SSH port (default 22) to ANY, but rather restrict it to trusted IP addresses.
- For more information about safely using SSH, see: SSH: Best practices

- Use a strong firewall. In Linux there are many options, iptables is a solid choice. A good basic iptables howto can be found here. In a network or if you need to protect several machines, you may want to consider a seperate hardware appliance as your firewall/UTM/... of choice.
- Iptables can do a very decent job once properly configured. In case you want to do less manual work, I advise to check out sshguard or artillery. Both can monitor and alert you when something funky happens. In the context of our blog post, it also looks for & protects against SSH bruteforce attempts. Another application to consider is fail2ban. An additional tool is snort. For more information about these tools, refer to their pages.
- Consider using SELinux. **S**ecurity-**E**nhanced Linux is a compulsory access control security mechanism provided in the kernel.
- Consider locking down cron jobs to only certain users. To deny all users from using cron you can use: **echo ALL >>/etc/cron.deny**
- Consider disabling remote root login. Read how to do that here.
- If you browse a lot, consider using NoScript as well.
- Keep your software and applications up-to-date, as on any system.
- Consider installing an antivirus as second opinion or at least as an additional layer. This is not a necessity but may come in handy. I recommend ClamAV.
- Don't forget to protect other appliances that may be running on *nix systems, for example your router (and nowadays, IoT devices). Upgrade the firmware as soon as possible and change the default root/admin password(s). Install updates/patches for your particular firewall/UTM/... as well.
- For even more (general) tips on hardening your Linux system (not against Xor.DDoS in particular): 20 Linux Server Hardening Security Tips

Conclusion

Don't be fooled: Linux malware very much exists and starts to become more prevalent. Other prevalent Linux malware nowadays is:

- Every ELF malware made by the ChinaZ actor or group (Linux/ChinaZ.DDoS, Linux/Kluh, ...)
- Linux/Aes.DDoS (Dofloo, MrBlack)
- Linux/Bash0day (Shellshock, Bashdoor)
- Linux/BillGates (Gates.B)
- Linux/Elknot (DnsAmp)
- Linux/GoARM (Ramgo, Goram)
- Linux/IptabLes and Linux/IptabLex

Note that this list is not complete and new ELF malware may pop up every day. (it's not a question of **if**, but **when** it will pop up) You can find a list of (interesting) Linux malware here.

Hopefully you have learned new things along the way of this blog post. For any specific questions, don't hesitate to leave a comment or contact me on Twitter: @bartblaze

To conclude this blog post, some acknowledgements and resources/references:

Acknowledgements

My colleague from Panda France, Julien Gourlaouen for informing me about this incident.

Everyone who helped, helps and will help in battling creators of ELF malware, in particular @MalwareMustDie for their excellent research and increasing awareness about these threats.

Last but not least, thank **you** for reading my blog post.

Resources

AlienVault - Xor.DDoS hashes, IPs and domains (see also related pulses)
Avast - Linux DDoS Trojan hiding itself with an embedded rootkit
Cisco - Threat Spotlight: SSHPsychos
FireEye - Anatomy of a Brute Force Campaign: The Story of Hee Thai Limited
KernelMode - Linux/Xor.DDoS (samples)
KernelMode - List of Linux Malware
MalwareMustDie - Fuzzy reversing a new China ELF "Linux/XOR.DDoS"
MalwareMustDie - Linux/XorDDoS infection incident report (CNC: HOSTASA.ORG)
MalwareMustDie - A bad Shellshock & Linux/XOR.DDoS CNC "under the hood"
MalwareMustDie - Polymorphic in ELF malware: Linux/Xor.DDOS
Yale - ELF Format (PDF)

Source: https://bartblaze.blogspot.com/2015/09/notes-on-linuxxorddos.html