

Snort IDS for the Aspiring Hacker, Part 1 (Installing Snort)

By [occupytheweb](#) 03/29/2016 9:47 pm

Welcome back, my neophyte hackers!

In the world of information security, the most common intrusion detection system (IDS) you will ever encounter is [Snort](#). As you probably already know, an IDS works similarly to antivirus (AV) software on your desktop; It attempts to identify malicious software on your network and warns you of its presence.

Snort, conceived by Martin Roesch in 1999, had become so popular that the networking giant Cisco purchased it in 2014, so you will likely see it built into nearly all Cisco devices in the near future. And since Cisco is the world's most widely used network device manufacturer, you are likely to encounter Snort everywhere.

Even if your organization never uses Cisco products (unlikely) or Snort, you will likely benefit from understanding how this IDS works since most other intrusion detection systems work similarly.

I have written a [few articles about Snort](#) in the past, but I thought it was time I put together [a comprehensive series on Snort](#). In this series, we will address how to use Snort from start to finish, including installation, configuration, managing output, rule writing, and alert monitoring.

Let's start Snorting!

Method 1 Installing Snort from the Repositories

Installing Snort is simple if you have Snort in your repository. Unfortunately, Snort is no longer in the Kali repository, so our first step is to add a repository that *does* have Snort. In this case, we will add some Ubuntu repositories.

First, we need to open the `/etc/sources.list` file. We can do this with any text editor (here, I will use Leafpad).

kali > leafpad /etc/apt/sources.list

```
File Edit Search Options Help
#
# deb cdrom:[Debian GNU/Linux 7.0 _Kali_ - Official Snapshot amd64 LIVE/IN
#deb cdrom:[Debian GNU/Linux 7.0 _Kali_ - Official Snapshot amd64 LIVE/INS
## Security updates
# Line commented out by installer because it failed to verify:
deb http://security.kali.org/kali-security kali/updates main contrib non-f
deb http://ch.archive.ubuntu.com/ubuntu/ saucy main restricted
deb-src http://ch.archive.ubuntu.com/ubuntu/ saucy main restricted
deb http://httpredir.debian.org/debian jessie main
deb-src http://httpredir.debian.org/debian jessie main
```

As you can see in the above screenshot, I added several Ubuntu repositories, which are also listed below. As Ubuntu is a fork from Debian (the base Linux distribution that Kali is built on), most Ubuntu packages will work on Kali.

```
deb http://ch.archive.ubuntu.com/ubuntu/ saucy main restricted
deb-src http://ch.archive.ubuntu.com/ubuntu/ saucy main restricted
```

```
deb http://httpredir.debian.org/debian jessie main
deb-src http://httpredir.debian.org/debian jessie main
```

After saving the file to update our repositories list, we need to next update the packages list. We can do this by typing:

```
kali > apt-get update
```

Once our packages have been updated, we can now install the Snort package from the repository with:

```
kali > apt-get install snort
```

And that's all there is to it. Snort is installed and ready to go! To test it, simply type:

```
kali > snort -V
```



```
kali > wget https://www.snort.org/downloads/snort/daq-2.0.6.tar.gz
```

```
kali > tar -xvzf daq-2.0.6.tar.gz
```

Next move to the daq directory.

```
kali > cd daq-2.0.6
```

Last, configure and make DAQ.

```
kali > ./configure
```

```
kali > make
```

```
kali > install
```

To download the Snort source code directly from Cisco/Snort, we can use the wget **command as follows (new version links can be found [here](#))**.

```
kali > wget https://snort.org/snort/snort-2.9.8.0.tar.gz
```

```
root@kali:~# mkdir snort_source
root@kali:~# cd snort_source
root@kali:~/snort_source# wget https://snort.org/downloads/snort/snort-2.9.8.0.tar.gz
--2016-03-16 09:24:53-- https://snort.org/downloads/snort/snort-2.9.8.0.tar.gz
Resolving snort.org (snort.org)... 104.16.62.75, 104.16.63.75, 104.16.65.75, ...
Connecting to snort.org (snort.org)|104.16.62.75|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://s3.amazonaws.com/snort-org-site/production/release_files/files/000/002/868/original/snort-2.9.8.0.tar.gz?AWSAccessKeyId=AKIAIXACIED2SPMSC7GA&Expires=1458145498&Signature=lZT2PUwS2qRyCFHlgxkj6hMkas%3D [following]
--2016-03-16 09:24:56-- https://s3.amazonaws.com/snort-org-site/production/rele
```

Once it has been downloaded, we need to un-tar it and decompress it. (For more information on the **tar** command, [check out my Linux Basics article](#).)

```
kali > tar -xvzf snort-2.9.8.0.tar.gz
```

```
root@kali:~# tar -xvzf snort-2.9.8.0.tar.gz
```

Next, we need to change directories to where the new Snort files are located.

```
kali > cd /snort-2.9.8.0
```

Then, we need to configure it.

kali > ./configure --enable-sourcefire

```
root@kali:/# ./configure --enable-sourcefire
```

Afterward, we need to use the **make** command, which determines which of the components of the source code needs to be recompiled and then issues the commands to do so.

kali > make

Finally, we **make install**. This takes our recompiled program components and places them in the proper directories.

kali > make install

Because we installed new library files with this installation, we will need to update the shared libraries. We can do this with by typing:

kali > ldconfig

```
root@kali:/# ldconfig
root@kali:/#
```

To enable us to start and run Snort from any directory, we can make a symbolic link from the binaries in */usr/local/bin/snort* and a new file in */usr/sbin* called **snort**. Since */usr/sbin* is in our PATH variable, we can then type Snort anywhere within our operating system to start our IDS.

kali > ln -s /usr/local/bin/snort /usr/sbin/snort

```
root@kali:/# ln -s /usr/local/bin/snort /usr/sbin/snort
```

Finally, let's test our installation of Snort by typing:

kali > snort

```
root@kali:~# snort
Running in packet dump mode

--== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "eth0".
Decoding Ethernet

--== Initialization Complete ==--

-*)> Snort! <*-
o"_)~ Version 2.9.2 IPv6 GRE (Build 78)
  ( )~ By Martin Roesch & The Snort Team: http://www.snort.org/snort-snort-team
eam

Copyright (C) 1998-2011 Sourcefire, Inc., et al.
Using libpcap version 1.3.0
Using PCRE version: 8.30 2012-02-04
Using ZLIB version: 1.2.7

Commencing packet processing (pid=3599)
```

As you can see, Snort has started and is running successfully in packet dump mode, aka sniffer mode.

Now that we have successfully installed Snort, we will progress to configuring it to detect malicious software. That will be in our next article in [this series](#), so keep coming back, my neophyte hackers!

Snort IDS for the Aspiring Hacker, Part 2 (Setting Up the Basic Configuration)

By [occupytheweb](#) 04/15/2016 4:33 am

Welcome back, my tenderfoot hackers!

As you should [know from before](#), Snort is the most widely deployed intrusion detection system (IDS) in the world, and every hacker and IT security professional should be familiar with it. Hackers need to understand it for evasion, and IT security professionals to prevent intrusions. So a basic understanding of this ubiquitous IDS is crucial.

In my [previous tutorial](#) in [this series](#), I showed you how to install Snort either by using the packages stored in the Ubuntu repository or from the source code directly from [Snort's website](#).

Now, I will show you the basic configuration so you can get started Snorting right away. In later tutorials, we will tweak the configuration to optimize its performance, send its output to a database, and analyze the output through a GUI interface.

Step 1 Get Snort Help

Before we configure Snort, let's take a look at its help file. Like with [most commands and applications in Linux](#), we can bring up the help file by typing the command or application's name followed by either **-?** or **--help**.

```
kali > snort --help
```



```
root@kali:/# snort --help

  __  __
 o"/  )~
  '    '

eam

-*> Snort! <*-
Version 2.9.2 IPv6 GRE (Build 78)
By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-t

Copyright (C) 1998-2011 Sourcefire, Inc., et al.
Using libpcap version 1.3.0
Using PCRE version: 8.30 2012-02-04
Using ZLIB version: 1.2.7

USAGE: snort [-options] <filter options>
Options:
  -A          Set alert mode: fast, full, console, test or none (alert fil
e alerts only)
  -b          "unsock" enables UNIX socket logging (experimental).
  -B <mask>   Log packets in tcpdump format (much faster!)
  -C          Obfuscated IP addresses in alerts and packet dumps using CIDR
mask
  -c <rules>  Use Rules File <rules>
  -C          Print out payloads with character data only (no hex)
  -d          Dump the Application Layer
  -D          Run Snort in background (daemon) mode
  -e          Display the second layer header info (are you able to hear?)
  -f          Turn off fflush() calls after binary log writes
  -F <bpf>    Read BPF filters from file <bpf>
  -g <gname>  Run snort gid as <gname> group (or gid) after initialization
  -G <Oxid>   Log Identifier (to uniquely id events for multiple snorts)
  -h <hn>    Set home network = <hn>
              (for use with -l or -B, does NOT change $HOME NET in IDS mode
```

As you can see in the screenshot above, I have circled several key switches.

- The first is **-c**, along with the location of the [Snort rules](#) file, tells Snort to use its rules. Rules in Snort are like virus signatures; they are designed to detect malicious packets and software.
- The second is **-d**, which tells Snort to show the application layer of data.
- The third is **-e**, which tells Snort to display the second layer information, or the [Data-Link Layer](#), which contains the MAC address.

If we scroll down a bit, we can see even more switches.


```

-H      Make hash tables deterministic.
-i <if> Listen on interface <if>
-I      Add Interface name to alert output
-k <mode> Checksum mode (all,noip,notcp,noudp,noicmp,none)
-K <mode> Logging mode (pcap[default],ascii,none)
-l <ld>  Log to directory <ld>
-L <file> Log to this tcpdump file
-M      Log messages to syslog (not alerts)
-m <umask> Set umask = <umask>
-n <cnt> Exit after receiving <cnt> packets
-N      Turn off logging (alerts still work)
-O      Obfuscate the logged IP addresses
-p      Disable promiscuous mode sniffing
-P <snap> Set explicit snaplen of packet (default: 1514)
-q      Quiet. Don't show banner and status report
-Q      Enable inline mode operation.
-r <tf>  Read and process tcpdump file <tf>
-R <id>  Include 'id' in snort_intf<id>.pid file name
-s      Log alert messages to syslog
-S <n=v> Set rules file variable n equal to value v
-t <dir> Chroots process to <dir> after initialization
-T      Test and report on the current Snort configuration
-u <uname> Run snort uid as <uname> user (or uid) after initialization
-ll     Use LTC for timestamps
-v      Be verbose
-V      Show version number
-X      Dump the raw packet data starting at the link layer
-x      Exit if Snort configuration problems occur
-y      Include year in timestamp in the alert and log files
-Z <file> Set the performonitor preprocessor file path and name
-?      Show this information

```

- The **-i** switch allows us to designate the interface that we want to use. Snort defaults to using the eth0 interface, so we only need to use this if we want to use a different one, such as wlan0.
- The **-l** switch will tell Snort where to log the data. As we will see, depending upon the configuration, Snort logs to `/var/log/snort` by default, but we can designate a different location here by placing the path after the **-l** switch.
- The **-v** switch tells Snort to be verbose, i.e., wordy, to provide us with all its information.

Step 2 Start Snort

Now that we know the basics of some of its switches, let's try running Snort; It can be run as a sniffer, packet logger, or NIDS (network intrusion detection system). Here, we'll just take a look at the sniffer (packet dump) and NIDS modes.

To run Snort in packet dump mode, type:

kali > snort -vde

```

root@kali:~# snort -vde
Running in packet dump mode

--== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "eth0".
Decoding Ethernet

--== Initialization Complete ==--

o"~)~
'-'~
'-'~

-*> Snort! <*-
Version 2.9.2 IPv6 GRE (Build 78)
By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
Copyright (C) 1998-2011 Sourcefire, Inc., et al.
Using libpcap version 1.3.0
Using PCRE version: 8.30 2012-02-04
Using ZLIB version: 1.2.7

Commencing packet processing (pid=7712)
03/17-07:16:09.197497 00:50:56:C0:00:08 -> 01:00:5E:00:00:FB type:0x800 len:0x5E
192.168.181.1:5353 -> 224.0.0.251:5353 UDP TTL:255 TOS:0x0 ID:8764 IpLen:20 DgmLen:80
Len: 52
00 00 84 00 00 00 00 01 00 00 00 00 09 5F 72 70 ....._rp
2D 6D 65 64 69 61 04 5F 74 63 70 05 6C 6F 63 61 -media._tcp.local
6C 00 00 0C 00 01 00 00 00 00 00 08 05 4B 45 49 l.....
54 48 C0 0C TH..

```

To run Snort as an NIDS, we need to tell Snort to include the configuration file and rules. In most installations, the configuration file will be at */etc/snort/snort.conf*, and that file will point to the Snort rules. We need to the **-c** switch and then the location of the configuration file.

kali > snort -vde -c /etc/snort/snort.conf

```
root@kali:~# snort -vde -c /etc/snort/snort.conf
Running in IDS mode

--== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined : [ 80:81 311 591 593 901 1220 1414 1830 2301
2381 2809 3128 3702 5250 7001 7777 7779 8000 8008 8028 8080 8088 8118 8123
8180:8181 8243 8280 8888 9090:9091 9443 9999 11371 ]
PortVar 'SHELLCODE_PORTS' defined : [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined : [ 1024:65535 ]
PortVar 'SSH_PORTS' defined : [ 22 ]
PortVar 'FTP_PORTS' defined : [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined : [ 5060:5061 5600 ]
Detection:
  Search-Method = AC-Full-Q
  Split Any/Any group = enabled
  Search-Method-Optimizations = enabled
  Maximum pattern length = 20
Tagged Packet Limit: 256
Loading dynamic engine /usr/lib/snort_dynamicengine/libsfe_engine.so... done
```

Step 3 Open the Config File

Like nearly every Linux application, Snort is configured using a configuration file that is a simple text file. Change the text in this file, save it, restart the application, and you have a new configuration.

Let's open the Snort configuration file with any text editor; I will be using Leafpad. Again, the configuration file is located at `/etc/snort/snort.conf`.

```
kali > leafpad /etc/snort/snort.conf
```

```

File Edit Search Options Help
1#-----
2# VRT Rule Packages Snort.conf
3#
4# For more information visit us at:
5#   http://www.snort.org           Snort Website
6#   http://vrt-sourcefire.blogspot.com/ Sourcefire VRT Blog
7#
8#   Mailing list Contact:      snort-sigs@lists.sourceforge.net
9#   False Positive reports:    fp@sourcefire.com
10#   Snort bugs:                bugs@snort.org
11#
12#   Compatible with Snort Versions:
13#   VERSIONS : 2.9.2.0
14#
15#   Snort build options:
16#   OPTIONS : --enable-ipv6 --enable-gre --enable-mpls --enable-targetb
17#
18#   Additional information:
19#   This configuration file enables active response, to run snort in
20#   test mode -T you are required to supply an interface -i <interface>
21#   or test mode will fail to fully validate the configuration and
22#   exit with a FATAL error
23#-----
24

```

When the *snort.conf* opens in your text editor, it should look like the screenshot above. Note that many of the lines are simply comments beginning with the *#* character. If we scroll down to lines 25-37, we can see in the comments that the configuration file has nine sections.

```

25#####
26# This file contains a sample snort configuration.
27# You should take the following steps to create your own custom configura
28#
29# 1) Set the network variables.
30# 2) Configure the decoder
31# 3) Configure the base detection engine
32# 4) Configure dynamic loaded libraries
33# 5) Configure preprocessors
34# 6) Configure output plugins
35# 7) Customize your rule set
36# 8) Customize preprocessor and decoder rule set
37# 9) Customize shared object rule set
38#####
39
40#####
41# Step #1: Set the network variables. For more information, see README.v
42#####
43
44# Setup the network addresses you are protecting
45ipvar HOME_NET any
46
47# Set up the external network addresses. Leave as "any" in most situation
48ipvar EXTERNAL_NET any
49#ipvar EXTERNAL_NET !HOME_NET

```

1. **Set the network variables**
2. Configure the decoder
3. Configure the base detection engine

4. Configure dynamic loaded libraries
5. Configure preprocessors
6. **Configure output plugins**
7. **Customize your rule set**
8. Customize preprocessor and decoder rule set
9. Customize shared object rule set

In this basic configuration, we will only address steps 1, 6, and 7 in that list (bolded above). With just these three, we can get Snort running effectively in most circumstances. As we get to more advanced configurations, we will address the others.

Step 4 Set the Variables

On line 45 above, we can see "ipvar HOME_NET any." This sets the IP addresses of your network to be protected. The HOME_NET is the variable that you assign IP addresses to, and it can be a single IP, a list of IPs separated by commas, a subnet in CIDR notation, or simply left as **any**.

The best practice here is to set the HOME_NET to the subnet you are protecting in CIDR notation, such as:

ipvar HOME_NET 192.168.181.0/24

Step 5 Check the Output

If we scroll down to lines 464-485, we can see the output plugins section. Here is where we tell Snort where and how to send us logs and alerts. By default, line 471 is commented out and 481 is active.


```

462 uu_decode_depth 0
463
464 #####
465 # Step #6: Configure output plugins
466 # For more information, see Snort Manual, Configuring Snort - Output Modu
467 #####
468
469 # unified2
470 # Recommended for most installs
471 output unified2: filename merged.log, limit 128, nostamp, mpls_event_typ
472
473 # Additional configuration for specific types of installs
474 # output alert_unified2: filename snort.alert, limit 128, nostamp
475 # output log_unified2: filename snort.log, limit 128, nostamp
476
477 # syslog
478 # output alert_syslog: LOG_AUTH LOG_ALERT
479
480 # pcap
481 # output log_tcpdump: tcpdump.log
482
483 # database
484 # output database: alert, <db_type>, user=<username> password=<password>
485 # output database: log, <db_type>, user=<username> password=<password> te

```

In this default configuration, Snort sends logs in tcpdump format to the */var/log/snort* directory. Line 471 enables what Snort calls unified logging. This type of logging logs both the complete packet and the alerts. For now, let's uncomment this type of output (unified2) and comment out line 481. Eventually, we will configure the output to go to a database of our choice (MS SQL, MySQL, Oracle, etc.).

Step 6 Disable Rules

Many times, to get Snort to run properly, we need to adjust the rules that it relies upon. Sometimes a rule or rule set will throw errors, so we need to comment out a rule or rule set temporarily. If we scroll down to line 504, this begins the "Customize your rule set" step of the configuration file.

Notice line 511 for local rules. These are rules that we can add to Snort's rule set in our customized configuration.

```
504 # Step #7: Customize your rule set
505 # For more information, see Snort Manual, Writing Snort Rules
506 #
507 # NOTE: All categories are enabled in this conf file
508 #####
509
510 # site specific rules
511 include $RULE_PATH/local.rules
512
513 #include $RULE_PATH/attack-responses.rules
514 #include $RULE_PATH/backdoor.rules
515 #include $RULE_PATH/bad-traffic.rules
516 # include $RULE_PATH/blacklist.rules
517 # include $RULE_PATH/bothnet-cnc.rules
518 # include $RULE_PATH/chat.rules
519 # include $RULE_PATH/content-replace.rules
520 #include $RULE_PATH/ddos.rules
521 # include $RULE_PATH/dns.rules
522 # include $RULE_PATH/dos.rules
523 # include $RULE_PATH/community-dos.rules
524 # include $RULE_PATH/exploit.rules
525 # include $RULE_PATH/community-exploit.rules
526 # include $RULE_PATH/finger.rules
527 # include $RULE_PATH/ftp.rules
```

To keep Snort from using a rule set, simply comment out the "include" part. Notice that there are many legacy rule sets that are commented out, but can become active simply by removing the # before them.

When we are done making our changes to the configuration file, we simply save the text file.

Step 7 Test the Configuration

Before we put Snort into production, let's test our new configuration. We can use the **-T** switch followed by the configuration file to test our Snort configuration.

```
kali > snort -T -c /etc/snort/snort.conf
```



```

root@kali:~# snort -T -c /etc/snort/snort.conf
Running in Test mode

--== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined : [ 80:81 311 591 593 901 1220 1414 1830 2301
2381 2809 3128 3702 5250 7001 7777 7779 8000 8008 8028 8080 8088 8118 8123
8180:8181 8243 8280 8888 9090:9091 9443 9999 11371 ]
PortVar 'SHELLCODE_PORTS' defined : [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined : [ 1024:65535 ]
PortVar 'SSH_PORTS' defined : [ 22 ]
PortVar 'FTP_PORTS' defined : [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined : [ 5060:5061 5600 ]
Detection:
  Search-Method = AC-Full-Q
  Split Any/Any group = enabled
  Search-Method-Optimizations = enabled
  Maximum pattern length = 20
Tagged Packet Limit: 256
Loading dynamic engine /usr/lib/snort dynamicengine/libsfe engine.so... done

```

Note that Snort has started in Test mode.

```

--== Initialization Complete ==--

o"~)~
....

-*> Snort! <*-
Version 2.9.2 IPv6 GRE (Build 78)
By Martin Roesch & The Snort Team: http://www.snort.org/snort-team
Copyright (C) 1998-2011 Sourcefire, Inc., et al.
Using libpcap version 1.3.0
Using PCRE version: 8.30 2012-02-04
Using ZLIB version: 1.2.7

Rules Engine: SF_SNORT DETECTION ENGINE Version 1.15 <Build 18>
Preprocessor Object: SF_SSH (IPv6) Version 1.1 <Build 3>
Preprocessor Object: SF_REPUTATION (IPv6) Version 1.1 <Build 1>
Preprocessor Object: SF_IMAP (IPv6) Version 1.0 <Build 1>
Preprocessor Object: SF_GTP (IPv6) Version 1.1 <Build 1>
Preprocessor Object: SF_DNP3 (IPv6) Version 1.1 <Build 1>
Preprocessor Object: SF_DCERPC2 (IPv6) Version 1.0 <Build 3>
Preprocessor Object: SF_SMTP (IPv6) Version 1.1 <Build 9>
Preprocessor Object: SF_POP (IPv6) Version 1.0 <Build 1>
Preprocessor Object: SF_MODBUS (IPv6) Version 1.1 <Build 1>
Preprocessor Object: SF_SDF (IPv6) Version 1.1 <Build 1>
Preprocessor Object: SF_SIP (IPv6) Version 1.1 <Build 1>
Preprocessor Object: SF_SSLPP (IPv6) Version 1.1 <Build 4>
Preprocessor Object: SF_DNS (IPv6) Version 1.1 <Build 4>
Preprocessor Object: SF_FTPTELNET (IPv6) Version 1.2 <Build 13>

Snort successfully validated the configuration!
Snort exiting
root@kali:~#

```

As we can see, Snort tested our configuration and validated it. We can now go into production!

Now you have a Snort application that is ready for intrusion detection. We still need to configure Snort to automatically update its rule set each day, send its output to a database of our choice, write our own rules, and then analyze the output through a GUI. So keep coming back, my tenderfoot hackers.

Snort IDS for the Aspiring Hacker, Part 3 (Sending Intrusion Alerts to MySQL)

By [occupytheweb](#) 04/15/2016 5:52 am

Welcome back, my hacker novitiates!

If you have been following this [new Snort series](#), you know that Snort is the world's most widely used intrusion detection/protection system. Now a part of the world's largest network equipment company, Cisco, it is likely to be found everywhere in one form or another. This makes a compelling argument for learning how to use it, as it will likely be a necessity in any security-related position.

In the previous tutorials in this series, we [installed](#) and [configured Snort](#) on our Kali system. In this tutorial, we will be taking the next step in building a production-level, professional IDS—getting intrusion notifications we can easily analyze.

Snort has many capabilities and configurations, but we want an IDS that can be used in a secure, professional way to alert us of intrusions. To do so, we will need to send our alerts to a database where we can analyze them. We will be setting up a print spooler specifically designed for Snort called [Barnyard2](#) and using it to send the data to a MySQL database.

In the previous guides, I used Kali as my base OS. Unfortunately, some of the libraries necessary to run Barnyard2 do not work well on Kali. As a result, subsequent tutorials in this series will be with Ubuntu. Everything you have done up to this point in setting up Snort works well in Ubuntu without modification. In reality, this makes much more sense as you would *not* likely be using an attack system like Kali as your server for Snort in a production environment.

Why Barnyard2?

Converting binary data to a human-readable form (usually binary-to-ASCII) is very CPU-intensive for any application. This is particularly true for Snort, and since we want to dedicate as much of our resources to packet capture and analysis as possible, we would like to lay off as much of these CPU-intensive tasks as we can. That's what Barnyard2 is designed for.

In a production environment, we want to send the event data from Snort to a database. Snort is capable of using any ODBC database such as Oracle, Microsoft's SQL Server,

PostgreSQL, and MySQL. Here we will be using MySQL as this is the most common configuration for Snort. If we can send the alerts to a database, then we can query the database with another tool to be able to make sense of this information. For instance, we can query for what rules were alerted on, what IP addresses did the attacks come from, what severity level were the intrusions, etc.

Here, we will configure Snort to process its alerts in binary form—it's simplest and least CPU-intensive form—and then use Barnyard2 to read these events, convert to human-readable form, and place them into a MySQL database.

I want to note before we proceed that Snort will work without Barnyard2, but it will slow down its functioning—and potentially not process and analyze some packets in a busy environment—which may be dangerous.

Step 1 Install Barnyard Prerequisites

Before we begin, we need to install some libraries and applications that are Barnyard2 prerequisites.

ubuntu > sudo apt-get install -y mysql-server libmysqlclient-dev mysql-client autoconf libtool

```
@ubuntu:~$ sudo apt-get install -y mysql-server libmysqlclient-dev mysql-cl
ient autoconf libtool
```

ubuntu > sudo apt-get install libpcap-dev libmysql-dev libprelude-dev

```
@ubuntu:/$ sudo apt-get install libpcap-dev libmysql-dev libprelude-dev
```

Step 2 Install Git

We will be downloading and installing the latest version of Barnyard2 from [GitHub](https://github.com). If you don't already have **git** on your system, you will need to install it now.

ubuntu> sudo apt-get update
ubuntu> sudo apt-get install git

Step 3 Edit Snort's Configuration File

To direct our alerts to a database, we will need to edit the *snort.conf* file. Open it with any text editor and go to the output section (section #6). There, we will be telling Snort to use our MySQL database (that we will be creating later in this tutorial with the username and password of your choice).

In this example, I have used some simple selections for the database name, user, and password, all **snort**. Of course, replace those with your own selections (see Step 6 below).

```
#####  
# Step #6: Configure output plugins  
# For more information, see Snort Manual, Configuring Snort - Output  
# Modules  
#####  
  
# unified2  
# Recommended for most installs  
# output unified2: filename merged.log, limit 128, nostamp,  
# mpls_event_types, vlan_event_types  
  
# Additional configuration for specific types of installs  
# output alert_unified2: filename snort.alert, limit 128, nostamp  
# output log_unified2: filename snort.log, limit 128, nostamp  
  
output database log,mysql, user=snort password=snort dbname=snort  
host=localhost
```

Step 4 Download Barnyard2

Barnyard2 is a print spooler that reduces the overhead of the Snort daemon to write the alerts to a human-readable form. Instead, it allows Snort to write these alerts in the far more efficient binary form, then Barnyard2 takes those binary files and converts them to a human-readable form. Lastly, it places them in a MySQL database for later analysis.

Let's go ahead and get Barnyard2 from GitHub.

```
ubuntu > git clone git://github.com/firnsy/barnyard2.git
```



```
@ubuntu:~$ git clone git://github.com/firnsy/barnyard2.git
Cloning into 'barnyard2'...
remote: Counting objects: 1246, done.
remote: Total 1246 (delta 0), reused 0 (delta 0), pack-reused 1246
Receiving objects: 100% (1246/1246), 1.16 MiB | 642.00 KiB/s, done.
Resolving deltas: 100% (835/835), done.
Checking connectivity... done.
keith@ubuntu:~$
```

Now, let's check to see whether it was downloaded and installed by doing a [long listing](#) on this directory.

ubuntu > ls -l

```
drwxrwxr-x 10 4096 Apr 12 13:23 barnyard2
drwxr-xr-x 17 4096 Mar  2 15:45 Desktop
drwxr-xr-x  2 4096 Jan  5 10:16 Documents
drwxr-xr-x  6 4096 Feb 24 14:48 Downloads
-rw-r--r--  1 8980 Jan  5 09:55 examples.desktop
drwxr-xr-x  2 4096 Jan  5 10:16 Music
drwxr-xr-x  2 4096 Jan  5 10:16 Pictures
drwxr-xr-x  2 4096 Jan  5 10:16 Public
drwxr-xr-x  2 4096 Jan  5 10:16 Templates
drwxr-xr-x  2 4096 Jan  5 10:16 Videos
```

As you can see, it has created a directory named *barnyard2*. Let's navigate to it and list its contents.


ubuntu > cd barnyard2

ubuntu > ls -l

```
-rwxrwxr-x 1 471 Apr 12 13:23 autogen.sh
-rw-rw-r-- 1 34200 Apr 12 13:23 configure.ac
-rw-rw-r-- 1 20997 Apr 12 13:23 COPYING
drwxrwxr-x 2 4096 Apr 12 13:23 doc
drwxrwxr-x 2 4096 Apr 12 13:23 etc
-rw-rw-r-- 1 17987 Apr 12 13:23 LICENSE
drwxrwxr-x 2 4096 Apr 12 13:23 m4
-rw-rw-r-- 1 212 Apr 12 13:23 Makefile.am
-rw-rw-r-- 1 7266 Apr 12 13:23 README
-rw-rw-r-- 1 13144 Apr 12 13:23 RELEASE.NOTES
drwxrwxr-x 2 4096 Apr 12 13:23 rpm
drwxrwxr-x 2 4096 Apr 12 13:23 schemas
drwxrwxr-x 5 4096 Apr 12 13:23 src
drwxrwxr-x 2 4096 Apr 12 13:23 tools
```

Notice the very first file named *autogen.sh*. Let's execute that script.

ubuntu > ./autogen.sh

A terminal window with a dark purple background. The prompt is @ubuntu:~/barnyard2\$. The command ./autogen.sh has been executed. The output shows libtoolize finding and copying various files, and autoreconf entering the directory and running aclocal with specific flags.

```
@ubuntu:~/barnyard2$ ./autogen.sh
Found libtoolize
libtoolize: putting auxiliary files in `.'.
libtoolize: copying file `./ltmain.sh'
libtoolize: putting macros in AC_CONFIG_MACRO_DIR, `m4'.
libtoolize: copying file `m4/libtool.m4'
libtoolize: copying file `m4/ltoptions.m4'
libtoolize: copying file `m4/ltsugar.m4'
libtoolize: copying file `m4/ltversion.m4'
libtoolize: copying file `m4/lt~obsolete.m4'
autoreconf: Entering directory `.'
autoreconf: configure.ac: not using Gettext
autoreconf: running: aclocal --force -I m4
```

Next, type the following line.

ubuntu > CFLAGS = '-lpthread'

Then, run the appropriate configure command for your system.

If you are using a 64-bit architecture—and I hope you are—this is the appropriate configure command:

**ubuntu> ./configure --with-mysql-libraries=/usr/lib/x86_64-linux-gnu --
prefix=\$HOME/barnyard2-install**

If you are using a 32-bit architecture, then the configure command changes slightly to:

**ubuntu > ./configure --with-mysql-libraries=/usr/lib/i386-linux-gnu --
prefix=\$HOME/barnyard2-install**


```

keith@ubuntu:~/barnyard2$ ./configure --with-mysql-libraries=/usr/lib/x86_64-linux-gnu --prefix=$HOME/barnyard2-install
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking how to print strings... printf
checking for style of include used by make... GNU
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...

```

There is one more library that Ubuntu needs for Barnyard2 called *libdumbnet-dev*. Let's get it from the repository.

sudo apt-get install libdumbnet-dev

Because the **make** script for Barnyard2 is expecting the dependency file named *dnet.h*, we need to create a symbolic link between *dumbnet.h* and *dnet.h* (the file names changed since the script was written).

ubuntu > ln -s /usr/include/dumbnet.h /usr/include/dnet.h

Then, update the libraries.

ubuntu> sudo ldconfig

Now, we can make Barnyard2.

ubuntu > make

```

keith@ubuntu:/snort_source/barnyard2$ make
make all-recursive
make[1]: Entering directory `/snort_source/barnyard2'
Making all in src
make[2]: Entering directory `/snort_source/barnyard2/src'
Making all in sfutil
make[3]: Entering directory `/snort_source/barnyard2/src/sfutil'
make[3]: Nothing to be done for `all'.
make[3]: Leaving directory `/snort_source/barnyard2/src/sfutil'
Making all in output-plugins

```

Finally, we need to make and install.

ubuntu > sudo make install

Step 5 Configure Barnyard2

We need to do some basic configuration on Barnyard2 to make certain it runs properly. First, we need to copy Barnyard2's configuration file to the */etc/snort* directory.

ubuntu > sudo cp /snort_source/etc/barnyard2.conf /etc/snort

Now we need to create a file that Barnyard2 needs in the *var/log* directory. This is the bookmark file.

ubuntu > touch /var/log/snort/barnyard2.waldo

```
@ubuntu:/etc$ sudo cp /snort_source/barnyard2/etc/barnyard2.conf /etc/snort
@ubuntu:/etc$ touch /var/log/snort/barnyard2.waldo
```

Step 6 Set up MySQL

Now that Barnyard2 is installed, compiled, and configured, we need to set up MySQL to receive the alerts. To do so, we need to:

- create a database
- create a database schema for the alerts
- create a user
- grant the user the appropriate privileges

Let's start by logging into the MySQL database system.

ubuntu > sudo mysql -u root -p

When prompted for a password, enter **snort**.

```
@ubuntu:~$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 48
Server version: 5.5.47-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```

You are now in the MySQL system and should receive a MySQL prompt. Let's create a database for the Snort system to use.

mysql > create database snort;

(Note that **snort** here is simply the name of the database we will be storing our alerts in. It could be named anything, but let's make it easy to remember.)

Then let's tell the system we want to use that database.

mysql > use snort;

```
mysql> create database snort
-> ;
Query OK, 1 row affected (7.18 sec)

mysql> use snort;
Database changed
mysql>
```

Barnyard2 comes with a script to create the schema for the Snort database. It is at `/snort_source/barnyard2/schemas/create_mysql`. We can run the script by typing:

mysql > /snort_source/barnyard2/schemas/create_mysql

```
mysql> source /snort_source/barnyard2/schemas/create_mysql
Query OK, 0 rows affected (1.21 sec)

Query OK, 1 row affected (0.16 sec)

Query OK, 0 rows affected (0.21 sec)

Query OK, 0 rows affected (0.19 sec)

Query OK, 0 rows affected (0.15 sec)

Query OK, 0 rows affected (0.22 sec)

Query OK, 0 rows affected (0.15 sec)

Query OK, 0 rows affected (0.19 sec)

Query OK, 0 rows affected (0.08 sec)

Query OK, 0 rows affected (0.14 sec)

Query OK, 0 rows affected (0.17 sec)
```

Next, we need to create a user for the snort database in MySQL.

mysql > CREATE USER 'snort'@'localhost' IDENTIFIED BY 'snort'

This command creates a user **snort** on the localhost server that uses the password **snort**. The name of the user and password can vary, but must be consistent with what you entered into the *snort.conf* file in Step 3 above.

Now to give that user the necessary permissions.

mysql > grant create, insert, select, delete, update on snort.* to 'snort'@'localhost';

This gives the user **snort** the rights to create objects, insert data, select data, delete data, and update data in the database **snort** at localhost.

```
mysql> CREATE USER 'snort'@'localhost' IDENTIFIED BY 'snort'
-> ;
Query OK, 0 rows affected (0.40 sec)

mysql> grant create,insert,select,delete, update on snort.* to 'snort'@'localhost';
Query OK, 0 rows affected (0.04 sec)

mysql> █
```

That's it! Now we have set up Snort to send its alerts to a file where Barnyard2 will pick them up, convert them to human-readable form, and place them in our Snort database in MySQL.