By: Justin Ellingwood 10 16

How To Use psad to Detect Network Intrusion Attempts on an Ubuntu VPS

Jan 10, 2014 Security, Firewall, Monitoring

Introduction

Being able to detect network activity that may indicate an intrusion attempt can help you take appropriate actions before an event occurs. Intrusion detection systems are available for this specific reason.

Intrusion detection systems are used to log suspicious connections and report when it looks like unusual activity is taking place. Some programs are used purely as a notification system, while others can actively attempt to block traffic that appear to be intent on causing harm.

The `psad` tool, which stands for port scan attack detection, is a piece of software that actively monitors your firewall logs to determine if a scan or attack event is in progress. It can then alert administrators, or take active steps to deter the threat.

In this guide, we will be exploring how to install and configure psad on an Ubuntu 12.04 VPS. The procedures should be fairly similar on other distributions.

Install psad

The psad intrusion detection system is available in Ubuntu's default repositories, so it can be easily acquired through apt:

```
sudo apt-get update
sudo apt-get install psad
```

In order to configure mail delivery to alert the administrator, you will be asked to configure the postfix mail server.

In most cases, you can select "Internet Site", and then enter the domain name associated with your server. This will be the domain portion of the name used in the "From" field in emails generated by psad.

Configure IPTables Rules

The way that psad detects activity on your server's ports is by monitoring the logs produced by a firewall application. Ubuntu ships with the iptables firewall by default, but it is completely unconfigured and is not monitoring or blocking anything by default.

Although you could get away with typing the following commands to simply enable logging, we will do a more robust configuration:

```
sudo iptables -A INPUT -j LOG
sudo iptables -A FORWARD -j LOG
```

If you entered the rules above, flush the rules before configuring so that we can start from scratch.

```
sudo iptables -F
```

You can see the current rules (which should only include the default policies at this point), by typing:

```
sudo iptables -S
```

```
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

We can now start appending our rules, mainly to the INPUT chain. We want to tell iptables to drop connections that we do not need or want. We need to add the rules to explicitly allow our authorized connections prior to adding restrictions.

The first rule will allow all traffic generated by our server, directed at our server. This type of connection is generally used for services to communicate with each other and pass information easily:

```
sudo iptables -A INPUT -i lo -j ACCEPT
```

Next, we want to add a rule to explicitly allow all traffic related to an existing connection. This will allow our current sessions to continue uninterrupted:

```
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCE
```

Next, we can add the services that we wish to keep open to the public. For SSH, we can add a line like this:

```
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

If we have a web server running on the default port 80, we can add a rule like this:

```
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

Add any other ports for legitimate, publicly accessible services you wish to leave open with the same syntax:

```
sudo iptables -A INPUT -p protocol --dport port_num -j ACCEPT
```

After you are done adding the legitimate services, we will drop all remaining connections. Anything that hits this rule did not match our rules that were for legitimate traffic.

Before we do that though, this we need to add the rule that tell iptables to begin logging traffic. This will cause iptables to log traffic that has not been handled yet.

```
sudo iptables -A INPUT -j LOG
```

We also should add this rule to the forward chain in case we end up forwarding traffic elsewhere.

```
sudo iptables -A FORWARD -j LOG
```

Finally, let's drop all extraneous traffic that hasn't matched yet. We can do this by either adding a rule that matches everything in the chain to the end like this:

```
sudo iptables -A INPUT -j DROP
```

Or, we can use the built-in policy functionality to configure what happens when a packet passes down the chain without matching any rules:

```
sudo iptables -P INPUT DROP
```

The results are functionally exactly the same.

One thing to note is that if you ever need to flush your iptables and you set up a DROP policy (instead of adding it as a rule at the bottom of the chain), you should reverse the policy before flushing:

```
sudo iptables -P INPUT ACCEPT
sudo iptables -F
```

If you fail to do this, your iptables rules will be flushed and only the default policy of dropping all incoming packets will remain. This will cut off all network traffic coming into your server, including traffic from your SSH connection.

By default, iptables does not maintain its rules between reboots, so after you have tested your configuration and are sure that it does what you would like, you can download and enable a tool that makes these rules persistent:

```
sudo apt-get install iptables-persistent
sudo service iptables-persistent start
```

Configure psad to Detect Scans

Now that we have an iptables rule set configured with what is called a "default drop" policy, we can begin configuring psad to start parsing the logs.

Open the main psad configuration file with root privileges:

```
sudo nano /etc/psad/psad.conf
```

The first things you should modify are at the top of the file. You should change the `EMAIL_ADDRESSES` parameter to match the email addresses you would like to notify when a report is generated. You should also modify the `HOSTNAME` to match your domain so that it references the correct machine:

```
EMAIL_ADDRESSES     address1@domain.com, address2@other.com;

HOSTNAME            your_domain.com;
```

Be sure to end each of your lines with a semi-colon (;) for psad to read the file correctly.

One section that you probably want to take a look at is the "danger levels" declarations. These levels are a way for psad to categorize threat levels.

They are automatically determined by the number of packets involved in an event, but you can assign certain types of traffic to be a certain danger level also. The default thresholds for each level to be reached are:

```
DANGER_LEVEL1           5;
DANGER_LEVEL2           15;
DANGER_LEVEL3           150;
DANGER_LEVEL4           1500;
DANGER_LEVEL5           10000;
```

You can change these levels depending on the level of sensitivity you would like psad to use for alerts.

You can also configure how sensitive psad will be through the `PORT_RANGE_SCAN_THRESHOLD` parameter. This determines the number of ports in a range that must be scanned before an alert is raised. The default is that an alert is raised after two ports are scanned.

```
PORT_RANGE_SCAN_THRESHOLD
```

SCROLL TO TOP

One of the most important things to configure is the `IPT_SYSLOG_FILE` parameter, because it currently is not pointed at a file that syslog uses by default.

Modify this to point at the syslog file, where psad will actually have an opportunity to look through active logs:

```
IPT_SYSLOG_FILE         /var/log/syslog;
```

If you use certain ports for things like port knocking, you should tell psad to ignore attempts on these ports so that you do not trigger alerts through routine activities:

```
IGNORE_PORTS            ports_or_range_to_ignore;
```

You can similarly ignore messages based on other things through the appropriately named `IGNORE_PROTOCOLS`, `IGNORE_INTERFACES`, and `IGNORE_LOG_PREFIXES` parameters.

If you find that you are getting alerts too often, you can set the threshold on emails by adjusting what level must be reached before an email is sent:

```
MIN_DANGER_LEVEL            1;  # Controls psad logging and email alerts
EMAIL_ALERT_DANGER_LEVEL    1;  # Applies only for email alerts
```

You can also limit the number of emails directly by setting this:

```
EMAIL_LIMIT                 0;
```

Zero means there is no limit. This limit is the number of emails that can be generated by threats from a single IP address.

For the time being, let's save and close the file.

Implement psad Intrusion Detection

Now that we have a basic psad configuration in place, complete with alert capabilities, we can implement our policies and activate our system.

Before we begin, we should update psad's signature definitions so that it can correctly recognize known attack types. Do this by calling:

```
sudo psad --sig-update
```

This will fetch the latest files and update the database.

Now, we need to restart the service to use these updates and to implement our configuration changes. Type:

```
sudo service psad restart
```

This will implement our log monitoring. To see the current status of psad detected events, type:

```
sudo service psad status
```

```
[+] psadwatchd (pid: 3737)  %CPU: 0.0  %MEM: 0.0
    Running since: Fri Jan 10 15:36:04 2014

[+] psad (pid: 3735)  %CPU: 0.0  %MEM: 0.3
    Running since: Fri Jan 10 15:36:04 2014
    Command line arguments: [none specified]
    Alert email address(es): example@domain.com
```

```
[+] Version: psad v2.1.7

[+] Top 50 signature matches:
        [NONE]

[+] Top 25 attackers:
        [NONE]

[+] Top 20 scanned ports:
        [NONE]

[+] iptables log prefix counters:
        [NONE]

    Total packet counters: tcp: 0, udp: 0, icmp: 0

[+] IP Status Detail:
        [NONE]

    Total scan sources: 0
    Total scan destinations: 0

[+] These results are available in: /var/log/psad/status.out
```

As you can see, nothing has been found yet. We can also see that detected events are logged to files located at `/var/log/psad/`.

Perform a Test Port Scan

From another computer, we should try to scan our server's ports to generate some hits on the firewall. We can do this with the `nmap` utility.

We will do a SYN tcp port scan from another machine. We will tell it to assume that our host is up by passing it the `-PN` option:

```
sudo nmap -PN -sS server_domain_or_ip
```

```
Starting Nmap 5.51 ( http://nmap.org ) at 2014-01-10 15:54 EST
Nmap scan report for server_domain_or_ip
```

```
Host is up (0.013s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 6.84 seconds
```

As you can see, this scan indicates what I configured for my firewall. Every port is labeled as "filtered" indicating that it is protected by a firewall, except for the SSH port, which is exposed.

On your server, you should re-run the status command:

```
sudo service psad status
```

You should see a lot larger list of alerts. Because the event was just a 1000 port scan, it triggered a signature match for a lot of different threats. For a more pointed attack, which is focused on a specific port or entry point, the signature would be much more helpful.

If you set up email alerts, you should have also have received an email or two. If you have a domain associated with the computer you scanned from, you should see a "who is" report on the owner associated with the scanning.

You can use this to attempt to contact the owner of the IP or perhaps the ISP or hosting provider.

Implement Intrusion Prevention

Now that we have verified that we can detect activity trying to access our server, we can optionally implement a prevention mechanism where psad can automatically modify iptables rules to ban scanners.

Before we do this, we should take a look at the `auto_dl` file:

```
sudo nano /etc/psad/auto_dl
```

This file specifies what danger level we should automatically set certain IP addresses. For instance, if we have an attacker who continuously tries to probe our system, we can set them to danger level 5 automatically:

```
attacker_ip          5;
```

On the other hand, you can basically exempt certain IP addresses from eliciting a reaction from psad. We could add the localhost here and set it to "0" if we hadn't added a rule in our iptables explicitly.

For our purposes, since we are going to be setting psad to automatically block traffic from a detected threat IP, we should add our home computer to this list so that we do not lock ourselves out:

```
local_computer_ip        0;
```

Save and close the file when you are finished.

Open up the psad configuration file again:

```
sudo nano /etc/psad/psad.conf
```

Search for the parameter called `ENABLE_AUTO_IDS`. This is the rule that allows psad to modify our firewall to block certain addresses. If you want to automatically do this, you can change it like this:

```
ENABLE_AUTO_IDS         Y;
```

Next, we want to decide what constitutes a threat level large enough to block the offending IP. We can do that by adjusting this parameter:

```
AUTO_IDS_DANGER_LEVEL       5;
```

Another important option is the total block time in seconds:

```
AUTO_BLOCK_TIMEOUT          3600;
```

This will block them for an hour by default.

Test Intrusion Prevention

We can test out how this works by temporarily banning ourselves. In the same configuration file, we are going to set these parameters:

```
ENABLE_AUTOIDS              Y;
AUTO_IDS_DANGER_LEVEL       4;
AUTO_BLOCK_TIMEOUT          60;
```

This will turn on automatic firewall configuration, will set the threshold to danger level 4, which we hit with a normal SYN scan, and set the block time for 60 seconds.

Save and close the file.

Open the `auto_dl` file if you added your home IP address, and temporarily comment that out.

```
# local_computer_ip      0;
```

Now, restart psad to make it re-read these files:

```
sudo service psad restart
```

From your home computer, you can re-run the scan you did last time:

```
sudo nmap -PN -sS server_domain_or_ip
```

At this point, after a few seconds, if you are connected to your psad machine through SSH, your connection should be dropped. You will not be able to connect again for 60 seconds.

This is because psad took action when your scan had hit enough ports to hit danger level 4. It modified the iptables rules to divert traffic to other chains that temporarily blocked your IP.

Once you are able to log in again, you can see the remnants of this diversion by checking out your iptable rules:

```
sudo iptables -S
```

```
-P INPUT DROP
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-N PSAD_BLOCK_FORWARD
-N PSAD_BLOCK_INPUT
-N PSAD_BLOCK_OUTPUT
-A INPUT -j PSAD_BLOCK_INPUT
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -j LOG
-A FORWARD -j PSAD_BLOCK_FORWARD
-A FORWARD -j LOG
-A OUTPUT -j PSAD_BLOCK_OUTPUT
```

As you can see, more chains were created and all input was directed into one of these chains. During the ban, this chain would have dropped connection attempts for your home connection's IP.

Now that you have tested this functionality, revert it back to what you'd like to use. You should probably increase the ban length if you are planning on actually using this functionality. Additionally, you should re-add the IP addresses that you know you will be connecting to:

```
local_computer_ip           0;
```

```
ENABLE_AUTOIDS              Y;
```

```
AUTO_IDS_DANGER_LEVEL       5;
AUTO_BLOCK_TIMEOUT          3600;
```

Don't forget to restart psad when you are done configuring your service for real-world application:

```
sudo service psad restart
```

Keep in mind, there are certain types of attacks that can spoof the source IP address. This means that an attacker who suspects that you have auto-blocking functionality enabled could cause you to accidentally ban legitimate sites or services. Be very careful and weigh the costs and benefits of this type of a configuration.

Conclusion

By correctly configuring a network intrusion detection tool like psad, you increase your chances of getting the needed warnings about threats before a problem actually happens. Tools like psad can give you advanced warning and can deal with some situations automatically.

The key to using psad effectively is to configure danger levels and email alerts appropriately, and then follow up on any problems. This tool, coupled with other intrusion detection resources like tripwire can provide fairly good coverage to be able to detect intrusion attempts.

By Justin Ellingwood

Author:
Justin Ellingwood

Related Tutorials

How Fail2Ban Works to Protect Services on a Linux Server

How To Secure Nginx with Let's Encrypt on Ubuntu 16.04

How To Automatically Firewall DigitalOcean Private Network Interfaces with Droplan

Adding Logstash Filters To Improve Centralized Logging (Logstash Forwarder)

How To Install Graylog 1.x on CentOS 7

16 Comments

Leave a comment...

Logged in as:

Notify me of replies to my comment

Comment

shaun *March 16, 2014*

I did 'sudo iptables -F' and I was disconnected from SSH and can't access it now...

kamaln7 *March 17, 2014*

@Shaun: You can use the Remote Console (https://cloud.digitalocean.com/droplets, select your droplet, and click the Remote Console button at the top-right corner of the page) to regain access to your droplet.

world.blue.communication *June 11, 2014*

Hi!

Cool stuff! I am trying to set up it, but psad does not show any alert when I run "sudo service psad status" even after the scan "sudo nmap -PN -sS server_domain_or_ip" from an other machine... What that could be?

Thanks!

Alex

> bb7b5b9 *December 27, 2015*
>
> Alex, I was facing your same problem. I think I got to an answer in a comment below...

sanjeev *September 11, 2014*

Been trying to follow this for a centos 7 Machine. I found the psad RPM so downloaded and installed it. But haven't found the iptables-persistent RPM. Any ideas, please?

> sanjeev *September 11, 2014*
>
> Or rather than installing iptables-persistent, isn't there a way to run the rules on startup?

Probably that could be a nice workaround.

thde *September 16, 2014*

On CentOS you have to save your configuration of iptables in a different way: https://www.centos.org/docs/5/html/5.1/Deployment_Guide/s1-iptables-saving.html

thde *September 16, 2014*

Could the logging of iptables produce a performance issue on high load servers?

LeoWinterDE *November 16, 2014*

yes, but depends on the server.

cbunting99 *October 23, 2014*

You really don't need to use iptable settings.. You can use UFW, https://www.digitalocean.com/community/tutorials/how-to-setup-a-firewall-with-ufw-on-an-ubuntu-and-debian-cloud-server along with the settings from the above article for psad.

phucdh86 *November 14, 2014*

I have message from support with content
"We have blocked someone from your IP space for abuse. Reason: Port Scanning"
How can't I do to fix my server with problem? I try use uwf to block all port, open only port 80 to fix this problem. But I want to find exactly what programs are running port scans from my server to fix the problem thoroughly.

shlo *November 20, 2014*

Hi,

I install psad 2.2.3 on server running RedHat 6.4.

The firewall log the DROP lines and psad send mail about them.

The problems are:

1. Each time I run *service psad restart* I get:
   Shutting down the psad psadwatchd daemon: * [FAILED]**
   Shutting down the psad daemon: [OK]

Starting psad: [OK]
*

2. When I run *service psad status*, to see the current status of psad detected events, I get:
   psad (pid 11335) is running...
   psadwatchd is stopped

Why?
Thanks.

Sinklar *March 4, 2015*

Hello,

I installed psad on Ubuntu 14.04 and I get this error by email: `You may just need to add a default logging rule to the /sbin/iptables 'filter' 'INPUT' chain (...)` . I *think* I configured it correctly...

This is the end portion of my iptables:

```
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -j LOG
-A INPUT -j DROP
-A FORWARD -j LOG
```

And my complete ip6tables (even if I didn't enable IPv6 when I created the droplet):

```
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -j LOG
-A INPUT -j DROP
-A FORWARD -j LOG
```

I see in the psad status that it is logging events.
Is there something missing?

Thanks!

Sinklar *March 6, 2015*

I'm not sure but I think the problem comes from Fail2ban, which is adding rules before the other iptables ones.

```
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-N fail2ban-ssh
-N fail2ban-ssh-ddos
-A INPUT -p tcp -m multiport --dports 222 -j fail2ban-ssh-ddos
-A INPUT -p tcp -m multiport --dports 222 -j fail2ban-ssh
-A INPUT -j LOG
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 222 -j ACCEPT
(…)
```

Any idea how I can make the LOG rule stay before Fail2ban rules?

bb7b5b9 *December 27, 2015*

Hi Sinklar! Your question inspired a solution because PSAD was not working for me after `even after the scan "sudo nmap -PN -sS server_domain_or_ip" from an other machine...` which Alex also reported in a previous comments.

Apparently the solution was indeed how you suggested, that is put the `-A INPUT -j LOG` rule before fail2ban's rules.

So to achieve this, you have to simple do (ubuntu) (as root, or use sudo)

1- export your current iptable rules with `iptables-save > myrules`
2- edit, reorder and save the iptables rules written in the file with `nano myrules` or use any other editor...
3- restore iptables rules from the edited file with `iptables-restore < myrules`

you can then save this rules using `service iptables-persistent save`

after doing this I was able to see my own portscan... :D

bb7b5b9 *December 27, 2015*

I tried to follow this tutorial, but as Alex reported " but psad does not show any alert when I run "sudo service psad status" even after the scan "sudo nmap -PN -sS server*domain*or_ip" from an other machine... "

What can it be? Maybe because it's an Amazon server?

Thanks!