

# Get Meterpreter Session Alert over slack

---

 [hackingarticles.in/get-meterpreter-session-alert-over-slack](https://hackingarticles.in/get-meterpreter-session-alert-over-slack)

Raj

May 13, 2019

You're going to learn ShellHerder in this post. It is a technique used to monitor all the sessions of Metasploit/Meterpreter. The basic idea to create it, that new incoming sessions could be easily monitored when Intruder cannot access the listener. This approach is quite helpful when a Pen-tester wants to get an alert for live phishing campaigns or another attack by monitoring for new sessions.

## Table of Content

---

### Introduction to ShellHerder

### Registering on Slack

- Add WebHooks App
- Configure WebHooks App

### Download & Configure ShellHerder

### Working Demo

### Introduction to ShellHerder

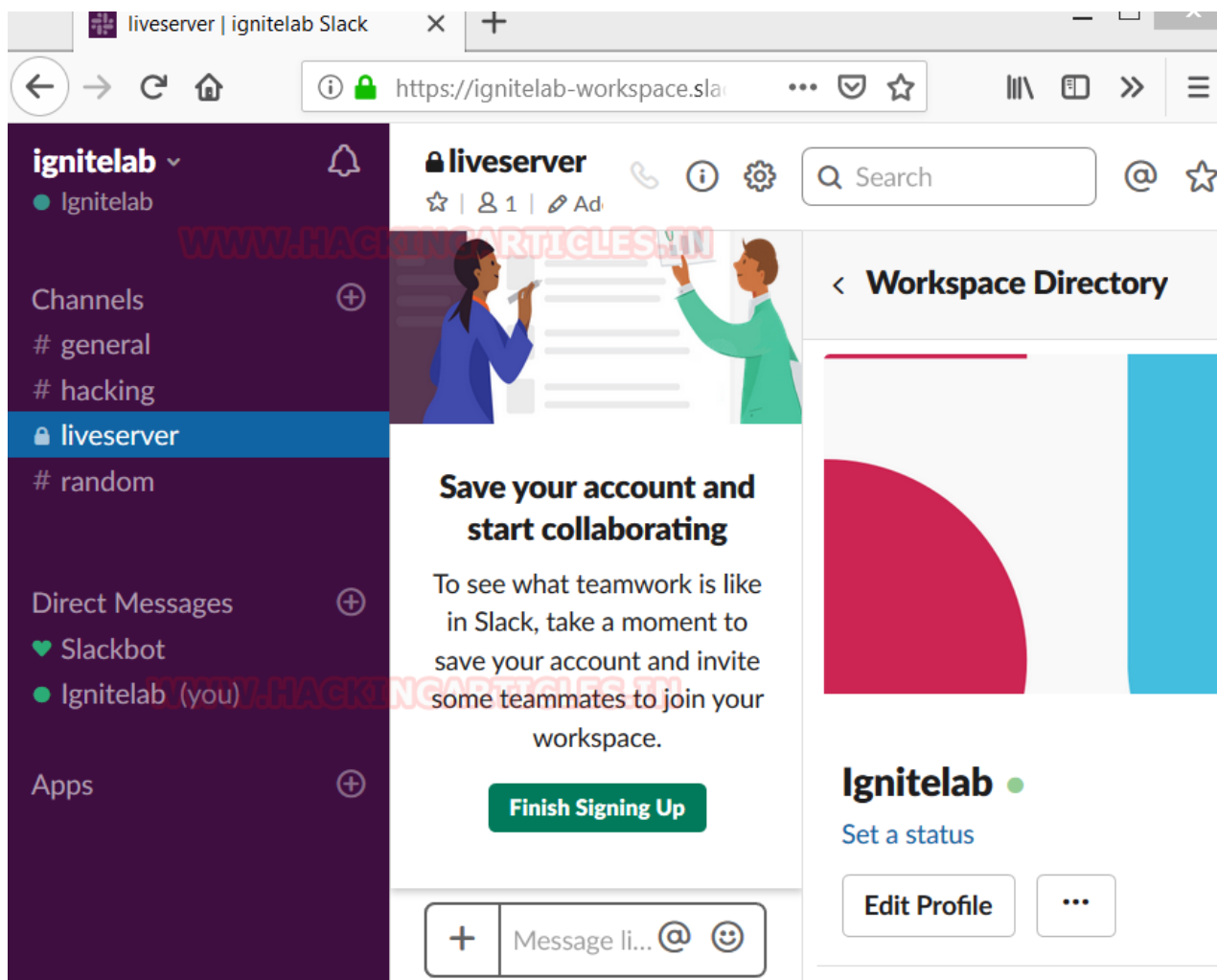
---

ShellHerder uses session subscriptions to monitor activity and then sends an alert to Slack using Slack's Incoming WebHooks. The alert is sent using the WebHook URL and a POST request and will tag a specified username and provide the computer name of the server with the session.

### Registering on Slack

---

We need a workspace on slack to use slack. To do this we need to register on slack. To create a new workspace on slack, [click here](#). This will require an email address. After that, it is required to create a channel. Here, we named our channel "live server".



## Add WebHooks App

To receive the updates from the Metasploit, we need to an app installed in the channel. Webhooks is the app that is perfect for this job. Now in order to add Webhooks, we first clicked on the Add an app Button inside our channel. Now, we will search for incoming Webhook and add it.

Manage apps...






# Browse Apps

View App Directory

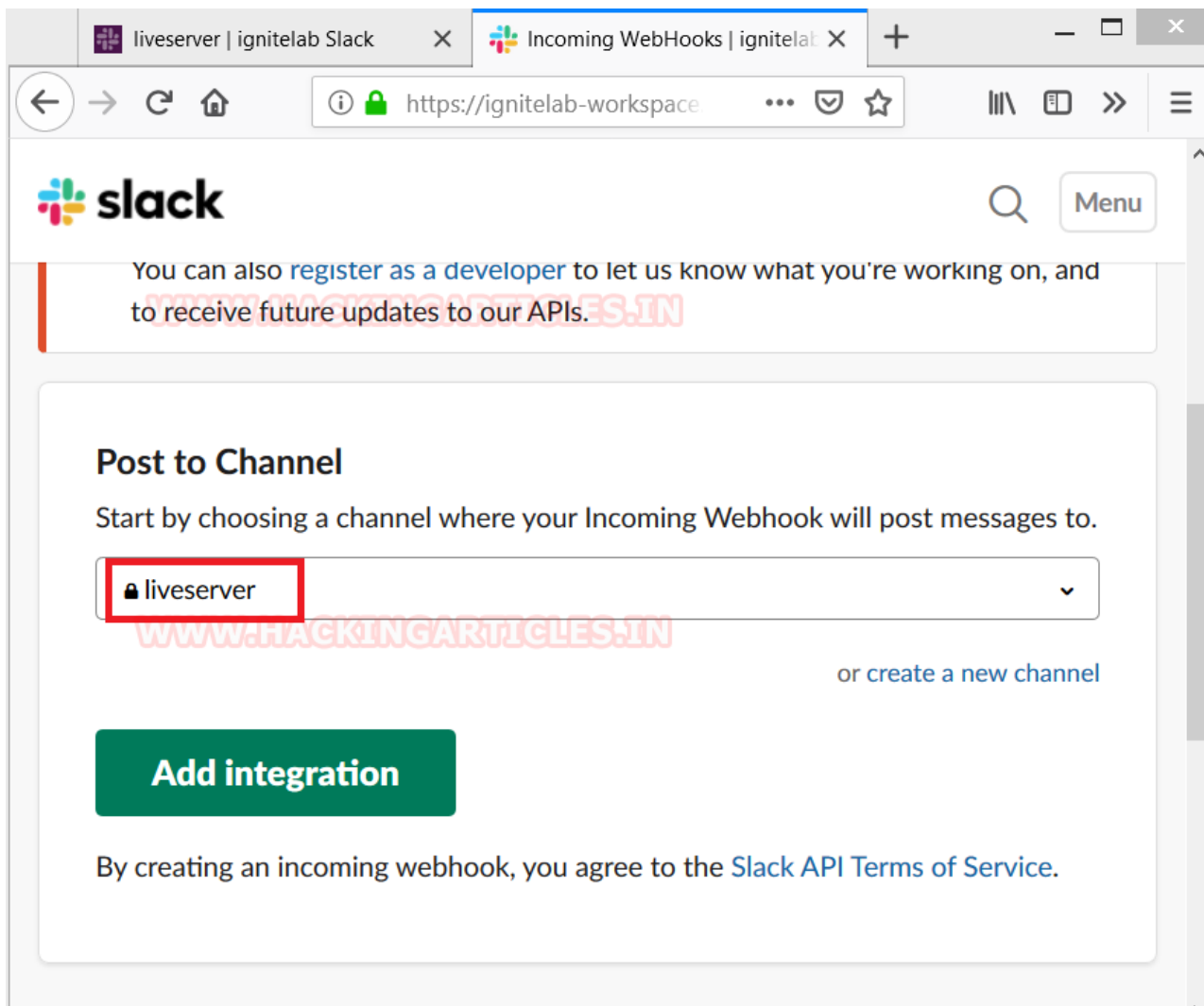
incoming webhook

From the App Directory

|   |   |         |
|---|---|---------|
|    | <b>Incoming WebHooks</b><br>Send data into Slack in real-time.                    | Install |
|    | <b>Brutalismbot</b><br>Mirror posts from /r/brutalism                             | Install |
|  | <b>Scribe</b><br>The most configurable assistant to manage your inbox, CRM and... | Install |

## Configure WebHooks App

After adding the Webhooks, we will be asked to configure some settings for the app. This will include the configuring the channel on which the incoming notifications will be broadcasted. Here we select our channel and click on the Add Integration Button.



After clicking the Add integration button, we will be presented with the WebHooks URL. **Copy** this URL, we are going to need it while we configure Notify.

## Integration Settings

### Post to Channel

Messages that are sent to the incoming webhook will be posted here.

liveserver

or [create a new channel](#)

### Webhook URL

Send your JSON payloads to this URL.

[Show setup instructions](#)

https://hooks.slack.com/services/TJ17T5NB0/BJ19AEBQF/hnv57U0I2uHkz1xjtl

[Copy URL](#) • [Regenerate](#)

### Descriptive Label

Use this label to provide extra context in your list of integrations (optional).

Optional description of this integration

### Customize Name

Choose the username that this integration will post as.

incoming-webhook

## Download & Configure ShellHerder

Now, we need to work upon our Kali Linux. We are going to use Shell Herder to connect to slack. This Metasploit plugin is aimed to keep an eye on the sessions. All including the ones which are opened or closed. It uses session subscriptions to monitor activities and can be linked to slack, which we just got setup.

```
git clone https://github.com/chrismaddalena/ShellHerder.git
```

```
root@kali:~# git clone https://github.com/chrismaddalena/ShellHerder.git
Cloning into 'ShellHerder'...
remote: Enumerating objects: 11, done.
remote: Total 11 (delta 0), reused 0 (delta 0), pack-reused 11
Unpacking objects: 100% (11/11), done.
root@kali:~# cp ShellHerder/ShellHerder.rb /usr/share/metasploit-framework/plugins/notify.rb
```

After downloading Shell Herder via git clone, we moved the directory inside the Metasploit Framework. So that we can use it directly inside the Framework. After copying the directory, we open an instance of the Metasploit Framework and load the notify plugin as shown in the image given image.

```

msf5 > load notify
[*] Successfully loaded plugin: notify
msf5 > help

notify Commands
=====

Command          Description
-----
notify_help      Displays help
notify_save      Save Settings to YAML File /root/.msf4/Notify.yaml.
notify_set_source Set source for identifying the source of the message.
notify_set_user   Set Slack username for messages.
notify_set_webhook Sets Slack Webhook URL.
notify_show_options Shows currently set parameters.
notify_start      Start Notify Plugin after saving settings.
notify_stop       Stop monitoring for new sessions.
notify_test       Send test message to make sure configuration is working.

```

Now, we will use the command `notify_show_options` to check for any pre-configured settings. Now that we can't find any. It was time to set the Webhook URL, which we copied earlier and add it inside the notify plugin. Also, we set the Slack User id and Source. After entering the relevant data, use the save command to save the configuration. Now that we have configured the Notify, Let us send a test message to see if the configuration is correct and working.

```

load notify
notify_show_options
notify_set_webhook <Above webhook URL>
notify_set_user @Ignitelab
notify_set_source Kali-Linux
notify_save
notify_test

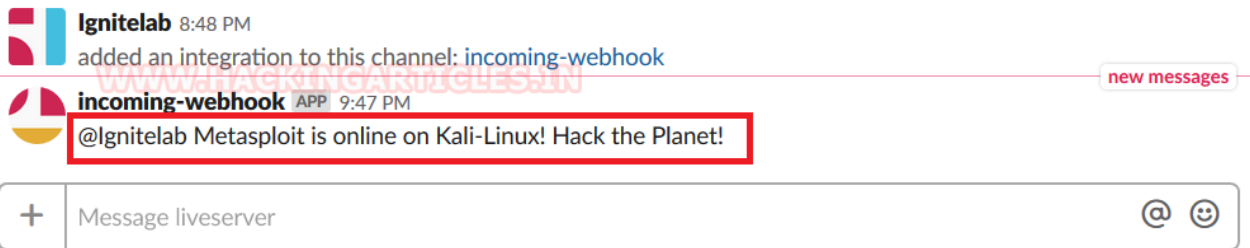
```

```

msf5 > load notify
[*] Successfully loaded plugin: notify
msf5 > notify_show_options
[*] Parameters:
[+] Webhook URL:
[+] Slack User:
[+] Source:
msf5 > notify_set_webhook https://hooks.slack.com/services/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
[*] Setting the Webhook URL to https://hooks.slack.com/services/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
msf5 > notify_set_user @Ignitelab
[*] Setting the Slack handle to @Ignitelab
msf5 > notify_set_source Kali-Linux
[*] Setting the Source to Kali-Linux
msf5 > notify_save
[*] Saving options to config file
[+] All settings saved to /root/.msf4/Notify.yaml
msf5 > notify_test
[*] Sending tests message
msf5 >

```

As we can see in the given image that, the slack received the test message we sent via Notify.



## Working Demo

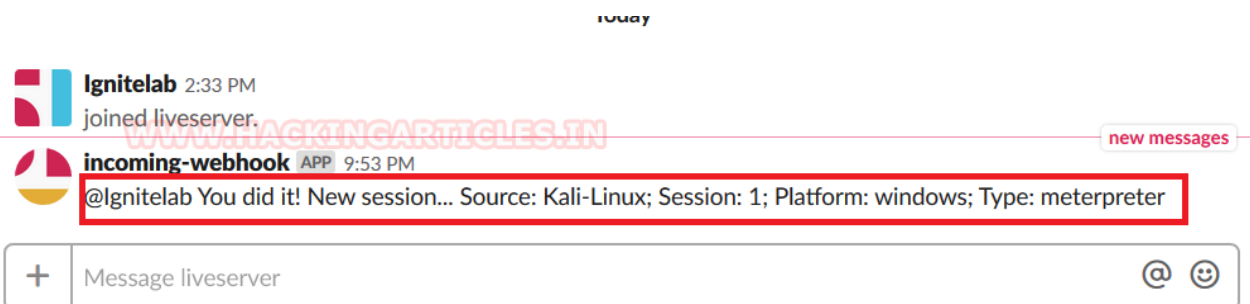
Now, to test the real working of Notify, we will exploit a machine, so that we can observe, whether or not it will notify us, when we get a session. We are exploiting a Windows machine using web delivery.

```
msf5 exploit(multi/script/web_delivery) >
[*] 192.168.174.1 web_delivery - Delivering Payload
[*] Sending stage (179779 bytes) to 192.168.174.1
[*] Meterpreter session 1 opened (192.168.174.137:4444 -> 192.168.174.1:5)
[*] @Ignitelab You did it! New session... Source: Kali-Linux; Session: 1;

msf5 exploit(multi/script/web_delivery) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer      : JARVIS
OS            : Windows 8.1 (Build 9600)
Architecture : x64
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter > |
```

As we expected, we got the notification on our slack channel, as soon as we got the session.



**Author:** Sanjeet Kumar is an Information Security Analyst | Pentester | Researcher  
Contact [Here](#)