

Help for Installing HackerMode 2.0

Contents

Necessary Accounts and Hosting Info.....	3
Step One of Installation (PHP setup)	4
Step Two of Installation (Securing PHP files)	4
Step Three of Installation (Setting up music files)	4
Step Four of Installation (Creating your Lambda).....	6
Step Four phase 2 (Configure your Lambda)	11
Step Four Phase 3 (Creating Dynamo DB).....	14
Step Five (Creating the Alexa Voice configuration)	24
Step Five phase 1	24
Step Five Phase 2 (Connecting your Lambda function and your Alexa skill together)	31
Step Five Phase 3 (Add a Lambda trigger for your Alexa Skill)	33
Step Five Phase 4 (Connecting the ARN)	34
Install Step Six (Install Kali Side of Hacker Mode 2.0).....	38
Prerequisite (Your Kali will need both Python 3 and 2.7! Why? Because of install dependencies for 3 rd party python libraries that will break otherwise) Default Kali should have 2.7 and 3 pre-installed, but if you are building a container you will need to include it.)	38
Phase 1 Install nmap library for Python.....	38
Phase 2 Install fixed MSFRpcd Python library.....	38
Phase 3 Install Pymetasploit from original source.....	38
Phase 4 Re-install Pymetasploit from xssninja Github	38
Phase 5 Install AlexaPwn.py file.....	39
Phase 6 Install and configure AlexaPwn6cfg.py file.....	39
Phase 7 Run MSFRPC daemon	39
Phase 8 Start AlexaPwn.py.....	40
Docker Alternative for Kali Install	40
Running Hacker Mode2 soup-to nuts	40

Necessary Accounts and Hosting Info

- An [Amazon Developer](#) account. This is required to create and configure Alexa skills.
- An [Amazon Web Services \(AWS\)](#) account. This is required for hosting a skill on AWS Lambda
- A [PHP hosting account](#) pointed at a domain that you own. GoDaddy or a million other hosting accounts provide PHP in a very basic hosting site. You can even set one up on your Amazon account which would be a nice touch. You can use AWS LightSail which gives you cheap PHP hosting slightly more than half the cost of GoDaddy.
- A [domain name](#) for your site and TLS binding.
- A [TLS certificate](#) because Alexa won't talk to a http site. Prior to getting a certificate you need to have a domain pointing at your PHP site for your certificate to be associated with.

GoDaddy will allow you to purchase a certificate, or give you complex instructions on how to install Let's Encrypt, but don't expect them to make it too easy for you to get a free cert. The instructions I've seen are hard to follow and sometimes are out-dated. If you don't use Let's Encrypt a cert may set you back \$70+ USD or somewhere around that price. If you use Amazon LightSail which is an excellent choice you will need to:

1. create your LightSail LAMP instance,
 2. create a load balancer, https://lightsail.aws.amazon.com/ls/docs/en_us/articles/verify-tls-ssl-certificate-using-dns-cname-https
 - a. create a certificate,
 - b. verify domain ownership,
 - c. attach the resulting certificate to load balancer
- You will also need a Kali instance or basis for now. This is untested on other platforms.

NOTE: (Feel free to contribute to a queue replacement to add extra security and reduce the overhead of owning HackerMode 2.)

Once you have Kali, these three accounts (Alexa, AWS, PHP hosting) and your domain name pointed at your PHP site, and a certificate for the domain, you can begin the process of actually installing the parts of HackerMode 2.0!

Step One of Installation (PHP setup)

1. In the HTML root or sub directory of your PHP hosting site, drop the PHP files.
2. Modify their names to some name that ONLY you are aware of. (Unless you want other people who are aware of the project to hit URLs on your site and send random crap in) Note the filenames you changed them to, as you will need to tell Alexa and Kali later where to find the queue files.
3. Each of the PHP files includes a relative path to other PHP files. Ensure the PHP code reflects the new file names you picked in step 2.
4. Log files are created at run-time. Ensure the PHP code is updated to reflect the new names of log files and the new names of log files being written to. It would be wise to change the names of log files so that they are not publicly known. These log file locations are included in the PHP files (Kali_Read.php, Kali2Alexa.php, Alexa_Read.php, and Alexa2Kali.php)
5. You can further secure the queue PHP files by IP address whitelisting.

Step Two of Installation (Securing PHP files)

Apply whatever security settings and headers to your PHP files as you feel appropriate. The PHP queue does some basic things for security but web hygiene is needed to stay safe ultimately.

Keep your queue file names secret so as to not let others write to your queue. You may want to white-list the IP of the Kali box so that only it and AWS resources can call the queue. Future versions will support authentication and signing, but that is not in the current release. You will typically want your Apache server serving pages with the HTTP headers: **X-Content-Type-Options: nosniff** and **X-XSS-Protection: 1; mode=block** headers turned on in httpd.conf or by adjusting the PHP files directly. You will probably also want the header **X-Frame-Options DENY** header turned on. Follow this guide for how to set these headers in the httpd.conf file of Apache. <https://geekflare.com/http-header-implementation/> There are other places to set headers in PHP and I'm not a hosting expert so you may need to do some consulting of the Google...

Step Three of Installation (Setting up music files)

Create a directory on your PHP site and put the .mp3 files in it. Note the URL to the MP3 files as you will need it later when configuring your lambda function. Both of the songs I found were royalty free and don't require attribution, but the Rick Roll version is by CutieSMG. You may want to replace the default songs with your own music anyway. Audacity and the LAME encoder will help you create Alexa-style MP3s. Remember that when you cut MP3s for Alexa that you need to encode them with FFMPEG or using the site that comes up on Google that uses the FFMPEG encoder for you. Music played by Alexa in this format has to be less than 90 seconds or it won't play. There won't even be an error, they just won't work and you'll be scratching your head for days trying to figure that one out.

Step Four of Installation (Creating your Lambda)

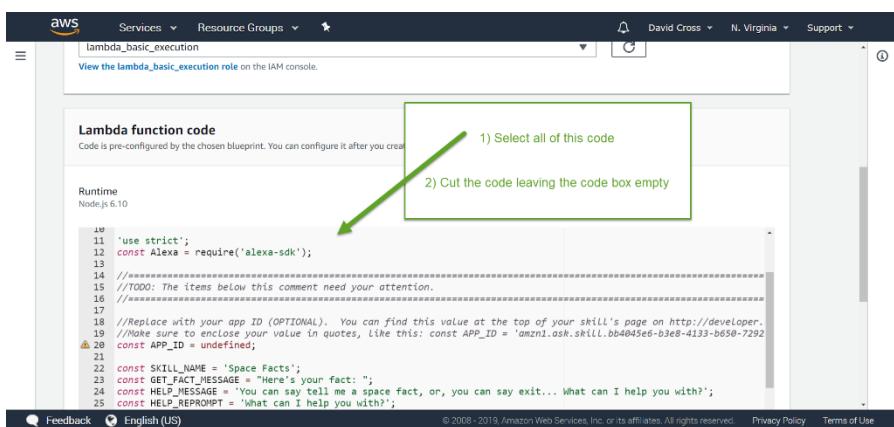
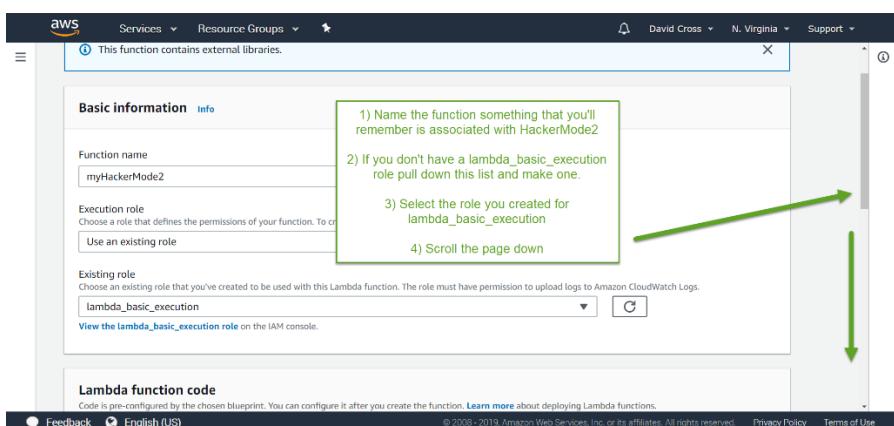
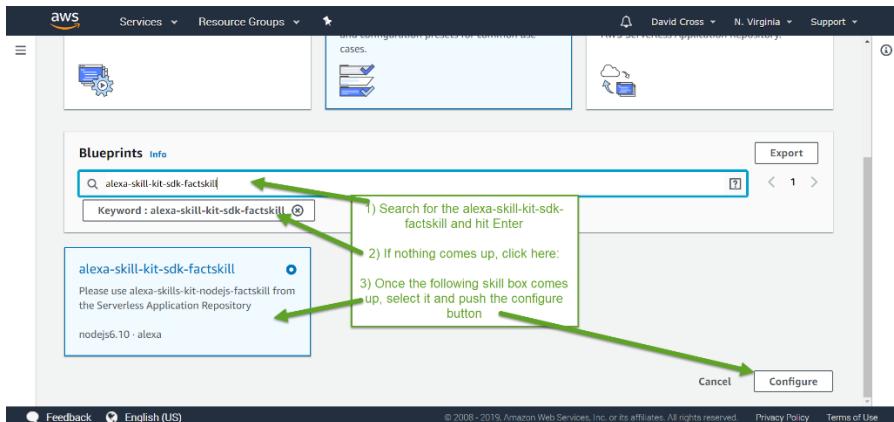
Step Four phase 1

Create the Lambda function. Ensure you are logged into AWS and then search for Lambda.

The screenshot shows the AWS Services console with the search bar at the top highlighted in green. The search term 'Search for Lambda' is typed into the bar. Below the search bar, there is a grid of service icons and names. The Lambda service icon (a blue square with a white lambda symbol) is located in the 'Compute' row. Other services visible include EC2, Lightsail, ECR, ECS, EKS, Blockchain, Amazon Managed Blockchain, Robotics, AWS RoboMaker, Analytics, Athena, EMR, CloudSearch, Elasticsearch Service, Kinesis, QuickSight, Data Pipeline, AWS Glue, MSK, Business Applications, Alexa for Business, Amazon Chime, WorkMail, End User Computing, WorkSpaces, AppStream 2.0, WorkDocs, WorkLink, Management & Governance, AWS Organizations, Security, Identity, & Compliance, and Internet Of Things.

The screenshot shows the AWS Lambda Functions page. The left sidebar has 'Functions' selected. The main area displays a table titled 'Functions (3)' with columns for 'Function name', 'Description', 'Runtime', 'Code size', and 'Last modified'. A green box highlights the 'Create function' button in the top right corner of the table header. A green arrow points from the text 'You probably won't have any lambda functions here, but either way push the Create Function button' to the 'Create function' button.

The screenshot shows the 'Create Function' wizard. The first step, 'Choose one of the following options to create your function.', is displayed. There are three options: 'Author from scratch', 'Use a blueprint', and 'Browse serverless app repository'. The 'Use a blueprint' option is highlighted with a green box and a green arrow pointing to it. Below this, a section titled 'Blueprints' shows a list of available blueprints, with 'kinesis-firehose-syslog-to-json' currently selected. Other blueprints listed include 'logmonitor-send-cloudwatch-events' and 'splunk-elb-application-access-logs-processor'.



← → 🔍 GitHub, Inc. [US] | https://github.com/xssninja/HackerMode2.0

Manage topics

1) go to github.com/xssninja/HackerMode2.0

2) Select the AWS Lambda directory

AWS Lambda First commit 19 days ago
Alexa Skill JSON First commit 19 days ago
Kali Adding a line to make the alexapwn.py executable 23 hours ago
PHP Site First commit 19 days ago
LICENSE Initial commit 19 days ago

Help people interested in this repository understand your project by adding a README.

Add a README

GitHub, Inc. [US] | https://github.com/xssninja/HackerMode2.0/tree/master/AWS%20Lambda

Hacking AWS Management ... Lambda Management Amazon Apps & Se... Sign In Office Fam Account Hushmail Secure E... Melanoma - Cancer...

search or jump to... Pull requests Issues Marketplace Explore

xssninja / HackerMode2.0 · Private

Code Issues 0 Pull requests 0 Projects 0 Insights Settings

Branch: master ▾

HackerMode2.0 / AWS Lambda /

xssninja First commit ... Latest commit b50f5

..
README.PNG First commit
README.txt First commit
index.js First commit
package.json First commit

branch: master · HackerMode2.0 / AWS Lambda / index.js · Raw Blame History

1 contributor

3650 lines (3671 sloc) | 286 kB

```

1 'use strict';
2 // By David Cross
3 // This really needs to be rewritten in Python
4 // (A) because I hate the call-back suckiness of early Node
5 // (B) because Python supports sync and asynch web calls which is handy
6 // (C) because Python rocks
7 // (D) because I wrote a second Alexa skill and I was in a hurry to get it coded up
8 const APP_ID = "amzn1.ask.skill.befcf5d4-d671-43be-8db7-d1f2e2bfff82"; // 1000 replies
9 const Alexa = require('alexa-sdk');
10 //const request = require('sync-request');
11 //const awsSDK = require('aws-sdk');
12 const https = require('https');
13 //var Data = require("./data");
14 var crypto = require('crypto');
15
16 // the following typically looks like: "exploit/osx/dns/upnp_location": "0", // but pymetasploit uses just the part after the exploit/
17 const exploitPorts = { "EXPLOITPORT1_BE_10": {
18   "osx/dns/upnp_location": "0",
19   "windows/dns/upnp_location": "0",
20   "windows/dce/ncsa/ms07_029_msn_zonename": "0",
21   "windows/scada/iges9_nmap": "0",
22   "windows/scada/iges9_nmap": "0"
23 }
24 }
```

Raw Blame History

Click Raw

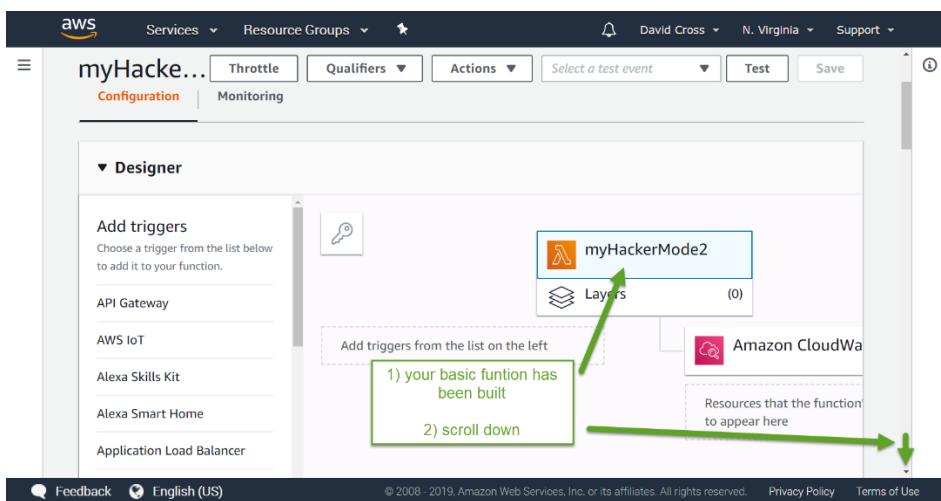
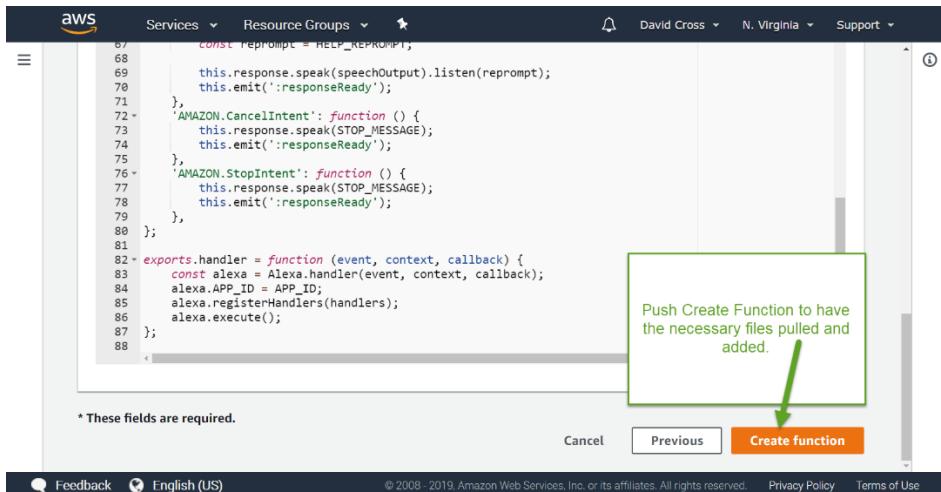
```

// By David Cross
// this really needs to be rewritten in Python
// A) because I hate the call-back suckiness of early Node
// B) because Python supports sync and asynch web calls which is handy
// C) because Python has modules
// D) because this was my second Alexa skill and I was in a hurry to get it coded up
const APP_ID = "awx1.ask.skill.beef";
const Alexa = require('ask-sdk');
//const https = require('https');
//const AWS = require('aws-sdk');
//var Data = require("./data");
//var crypto = require('crypto');

// the following typically looks like: "exploit/osx/macos/upnp location"
//const https = require('https');
//const AWS = require('aws-sdk');
//const Data = require("./data");
//var crypto = require('crypto');

// windows/dcerpc/m07_029 ends_zonename": "0",
//windows/scadacfg$0_msc": "0",
//windows/freewall/blacklist_ip": "1",
//windows/freewall/blacklist_ip": "1",
//linux/ftp/proftn_replace": "21",
//linux/ftp/proftn_telnet_lace": "21",
//mainline/ftp/tls_fix": "21",
//multi/ftp/ftpexec_bash_exec": "21",
//multi/ftp/ftptpd_site_exec_format": "21",
//osx/ftp/webstar_ftp_user": "21",
//unix/ftp/ncftpd_ftpd": "21",
//win/ftp/214_buckshot": "21",
//windows/ftp/xdemon_ftp_user": "21",
//windows/ftp/ability_server_star": "21",
//windows/ftp/binary_star": "21",
//windows/ftp/cesset2_md5": "21",
//windows/ftp/commd_ftp_fntstr": "21",
//windows/ftp/dreamft_fntstr": "21"

```



myHacke... Throttle Qualifiers Actions Select a test event Test Save

```

index.js
1 /* eslint-disable func-names */
2 /* eslint quote-props: ["error", "consistent"] */
3 /**
4 * This sample demonstrates a simple skill built with the Amazon Alexa Skills
5 * nodejs skill development kit.
6 * This sample supports multiple languages, (en-US, en-GB, de-DE).
7 * The Intent Schema, Custom Slots and Sample Utterances for this skill, as well
8 * as testing instructions are located at https://github.com/alexa/skill-sample-
9 * */
10
11 'use strict';
12 const Alexa = require('alexa-sdk');
13
14 /**
15 //TODO: The items below this comment need your attention.
16 //=====
17
18 //Replace with your app ID (OPTIONAL). You can find this value at the top of
19 //Make sure to enclose your value in quotes, like this: const APP_ID = 'amzn1.
20 const APP_ID = undefined;
21
22 const SKILL_NAME = 'Space Facts';
23 const GET_FACT_MESSAGE = "Here's your fact: ";
24 const HELP_MESSAGE = 'You can say tell me a space fact, or, you can say exit.
25 const HELD_RESPONSE = 'What can I help you with?';

```

Feedback English (US) © 2008-2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

myHacke... Throttle Qualifiers Actions Select a test event Test Save

```

index.js
1 /* eslint-disable func-names */
2 /* eslint quote-props: ["error", "consistent"] */
3 /**
4 * This sample demonstrates a simple skill built with the Amazon Alexa Skills
5 * nodejs skill development kit.
6 * This sample supports multiple languages, (en-US, en-GB, de-DE).
7 * The Intent Schema, Custom Slots and Sample Utterances for this skill, as well
8 * as testing instructions are located at https://github.com/alexa/skill-sample-
9 * */
10
11 'use strict';
12 const Alexa = require('alexa-sdk');
13
14 /**
15 //TODO: The items below this comment need your attention.
16 //=====
17
18 //Replace with your app ID (OPTIONAL). You can find this value at the top of
19 //Make sure to enclose your value in quotes, like this: const APP_ID = 'amzn1.
20 const APP_ID = undefined;
21
22 const SKILL_NAME = 'Space Facts';
23 const GET_FACT_MESSAGE = "Here's your fact: ";
24 const HELP_MESSAGE = 'You can say tell me a space fact, or, you can say exit.
25 const HELD_RESPONSE = 'What can I help you with?';

```

Feedback English (US) © 2008-2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

myHacke... Throttle Qualifiers Actions Select a test event Test Save

```

index.js
3815 IP_LOOKUP_REPEAT_MESSAGE: "You can say repeat if you want to hear
3816 IP_LOOKUP_ERROR_MESSAGE: "I'm sorry, I didn't receive a result
3817 IP_LOOKUP_NOT_FOUND_WITH_ITEM_NAME: "I can't find the IP address
3818 IP_LOOKUP_NOT_FOUND_WITHOUT_ITEM_NAME: "I don't know that IP address
3819 IP_LOOKUP_NOT_FOUND_REPROMPT: "Can I help you with something else?
3820 IP_LOOKUP_DISPLAY_CARD_TITLE: "Lookup on IP address %s",
3821 IP_LOOKUP_BAD_MISSING_IP: "I didn't get all of the IP address. ",
3822 IP_LOOKUP_BAD_MISSING_IP_1: "The IP address is invalid. ",
3823 IP_LOOKUP_MISSING_ONE: "I'm missing the first number. ",
3824 IP_LOOKUP_MISSING_TWO: "I'm missing the second number. ",
3825 IP_LOOKUP_MISSING_THREE: "I'm missing the third number. ",
3826 IP_LOOKUP_MISSING_FOUR: "I'm missing the fourth number. ",
3827 IP_LOOKUP_TOOLBIG_ONE: "The first number is too large. ",
3828 IP_LOOKUP_TOOLBIG_TWO: "The second number is too large. ",
3829 IP_LOOKUP_TOOLBIG_THREE: "The third number is too large. ",
3830 IP_LOOKUP_TOOLBIG_FOUR: "The fourth number is too large. ",
3831 IP_LOOKUP_LOCAL_IP: "I'm just making a wild guess, but I think the IP
3832 IP_LOOKUP_EMPTY_RESPONSE: "The response seems to indicate the IP
3833
3834 },
3835 );
3836 };
3837
3838 exports.handler = (event, context, callback) => {
3839   // ...
390

```

Feedback English (US) © 2008-2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Step Four phase 2 (Configure your Lambda)

Configure the Lambda function.

myHackerM... Throttle Qualifiers Actions Select a test event Test Save

You can use tags to group and filter your functions. A tag consists of a case-sensitive key-value pair. Learn more

lambda-console:blueprint	alexa-skill-kit-sdk-factskill	Remove
Key	Value	Remove

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

Use an existing role

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

lambda_basic_execution

Basic settings

Description: Please use alexa-skills-kit-nodejs-factskill fr

Memory (MB): 128 MB

Timeout: 120 seconds

myHackerM... Throttle Qualifiers Actions Select a test event Test Save

109 "windows/smtp/poops_overflow1": "25",
110 "windows/wins/ms04_045_wins": "42",
111 "freedesdf/tacacs/xatcacsd_report": "49",
112 "windows/ftp/attftp": "69",
113 "windows/ftp/futuresoft_tftp_traversal": "69",
114 "windows/ftp/dlink_long_filename": "69",
115 "windows/ftp/futuresoft_transfemode": "69",
116 "windows/ftp/netdecision_tftp_traversal": "69",
117 ... 1:1 JavaScript Tabs: 4

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. Learn more

Key	Value	Remove
-----	-------	--------

Encryption configuration

Tags

myHackerM... Throttle Qualifiers Actions Select a test event Test Save

You can use tags to group and filter your functions. A tag consists of a case-sensitive key-value pair. Learn more

lambda-console:blueprint	HackerMode2	Remove
Key	Value	Remove

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

Use an existing role

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

lambda_basic_execution

Basic settings

Description: HackerMode NodeJS program to run metasploit

Memory (MB): 128 MB

Timeout: 120 seconds

Screenshot of the AWS Lambda function configuration page for "myHackerM...". The "Environment variables" section is shown. A green box highlights the "Save" button at the top right. Red arrows point from the following environment variable pairs to a callout box:

- AlexaReadQueuePage: Alexa_Read.php
- AlexaToKaliQueuePage: Alexa2Kali.php
- KaliReadQueuePage: Kali_Read.php
- KaliToAlexaQueuePage: Kali2Alexa.php

The callout box contains instructions:

- Set the AlexaReadQueuePage variable to the name of the Alexa_Read.php page
- Set the AlexaToKaliQueuePage variable to Alexa2Kali.php
- Set the next two exactly as specified in the image and Save the changes

Screenshot of the AWS Lambda function configuration page for "myHackerM...". The "Environment variables" section is shown. A green box highlights the "Save" button at the top right. Red arrows point from the following environment variable pairs to a callout box:

- AlexaReadQueuePage: Alexa_Read.php
- AlexaToKaliQueuePage: Alexa2Kali.php
- KaliReadQueuePage: Kali_Read.php
- KaliToAlexaQueuePage: Kali2Alexa.php
- LocalSubnetPrefix1: 192
- LocalSubnetPrefix2: 168
- MyName: Lord Vader

The callout box contains instructions:

- Set the Subnet prefix for where you want to scan
- Set the second number of the subnet IP address
- Set the name that you want Alexa to call you.

Screenshot of the AWS Lambda function configuration page for "myHackerM...". The "Environment variables" section is shown. A green box highlights the "Save" button at the top right. Red arrows point from the following environment variable pairs to a callout box:

- KaliReadQueuePage: Kali_Read.php
- KaliToAlexaQueuePage: Kali2Alexa.php
- LocalSubnetPrefix1: 192
- LocalSubnetPrefix2: 168
- MyName: Lord Vader
- PlayElevatorMusic: www.xss.ninja/mp3/1.mp3
- PlaySomethingFun: www.xss.ninja/mp3/2.mp3
- RickRollPath: www.xss.ninja/mp3/outro.mp3
- SecretKey: ThereAreNoStringsOnMe!
- SecretURL: www.xss.ninja

The callout box contains instructions:

- Set the url (without the https) to the music file that will play when the PlaySomethingFun function kicks off. This file needs to live on a HTTPS web site and needs to be publicly available on the internet. It also must be the proper MP3 encoding for Alexa and less than 90 seconds in length.
- Set the URL to the rickroll sound file. It must be the Alexa MP3 format. It also must be hosted publicly under HTTPS and must be less than 90 seconds in length.
- The Secret key is a key that signs your requests.
- The URL to the HTTPS web site that hosts the AlexaRead queue pages and Kali2...php files

* Lambda functions won't reach out to standard HTTP sites due to security concerns to put a cert on your site or use Let'sEncrypt on it*

When you're all done it should look something like this minus the green boxes! (The green boxes indicate information that is sensitive!)

myHackerMode2

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings with code. [Learn more](#)

AlexaReadQueuePage	Alexa_ReadQueuePage.php	The PHP pages that act as Queue read and Queue send	Remove
AlexaToKaliQueuePage	Alexa2KaliQueuePage.php		Remove
IPStackAccessKey		Get your own IPStack Access key and paste it here. 10K per month requests for FREE! So why?	Remove
LocalSubnetPrefix1	192	The first two prefix numbers of your subnet that you want to hack. Your Kali instance must be in this range as well for your protection!	Remove
LocalSubnetPrefix2	168		Remove
MyName	Lord Vader	Not supported yet, but soon!	Remove
PlayElevatorMusic	www.xss.ninja/BBBelevatormusic		Remove
PlaySomethingFun	www.xss.ninja/bbb.mp3		Remove
RickRollPath	www.xss.ninja/rroll.mp3		Remove
SecretKey		Put in your own secret key here... Just anything to "sign" your requests with.	Remove
SecretURL	www.xss.ninja	The root domain of your PHP hosting site. (Without a domain your certificate won't work!)	Remove
Key	Value		Remove

► Encryption configuration

Tags

In the image above, note that you probably want your queue pages to be named something unique to your install. (Security by obscurity!) And if you want to use the IP Lookup ability, you'll want to register at <https://ipstack.com/quickstart> and get an IPStack Access Key. You will enter that secret value in the environment variable as shown above... (Don't post it anywhere or they'll block you!)

Step Four Phase 3 (Creating Dynamo DB)

Create Dynamo DB and rights for session preservation.

The screenshot shows the AWS DynamoDB service dashboard. On the left, there's a sidebar with options like 'Dashboard', 'Tables', 'Backups', etc. The main area has a callout box with three steps:

- 1) Select "Services" and search for DynamoDB
- 2) Select DynamoDB
- 3) Select Create Table

A green arrow points from the first step to the 'Create table' button, which is highlighted with a blue box.

The screenshot shows the 'Create DynamoDB table' wizard. It has two main sections:

- Table name***: KaliSessionData
- Primary key***: userId

Below these, there's a 'Table settings' section with a checkbox for 'Use default settings' and a note about secondary indexes and provisioned capacity.

A callout box contains the following steps:

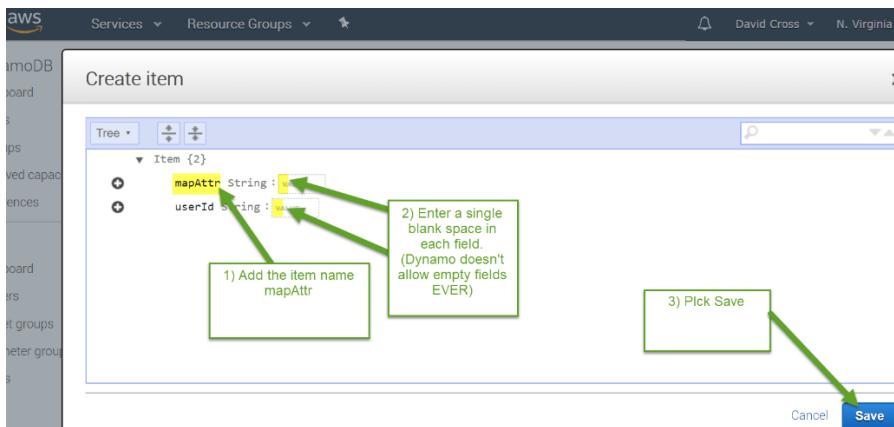
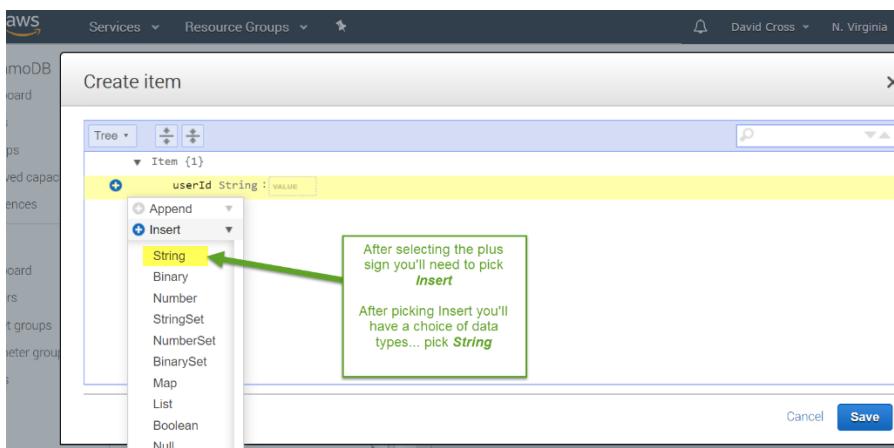
- 1) Name the table: KaliSessionData
- 2) Name the primary key or partition key: userId
- 3) Scroll the page down

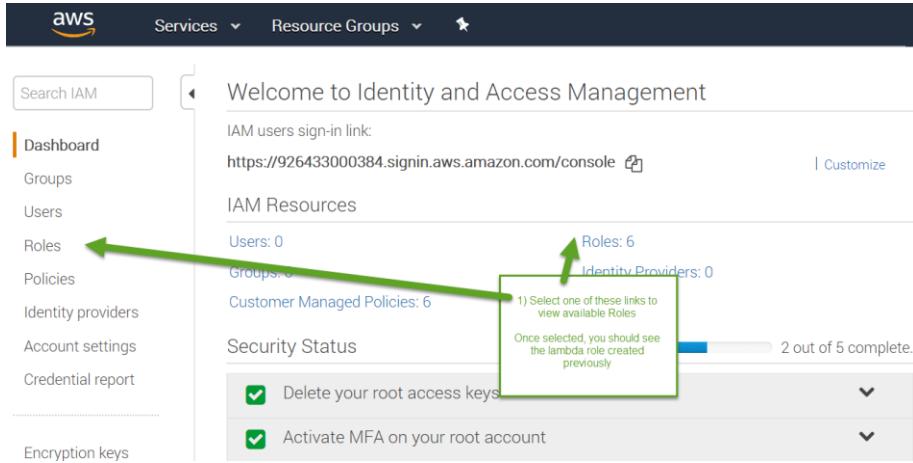
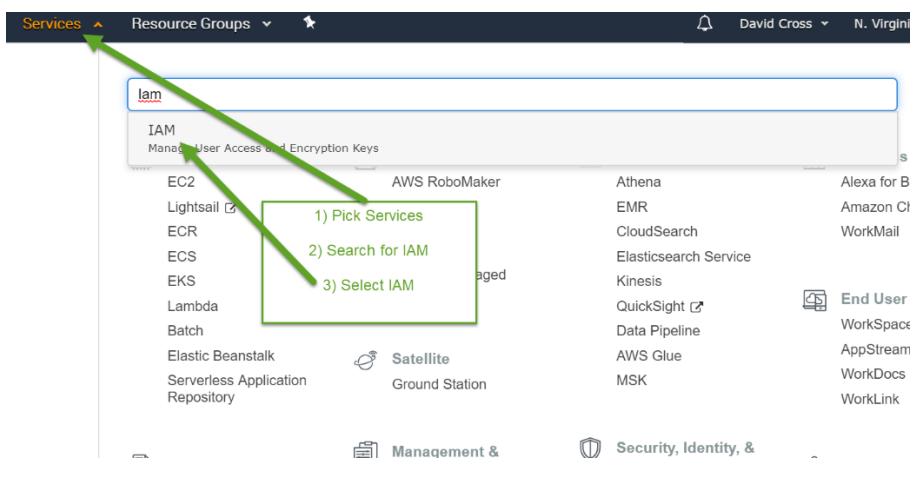
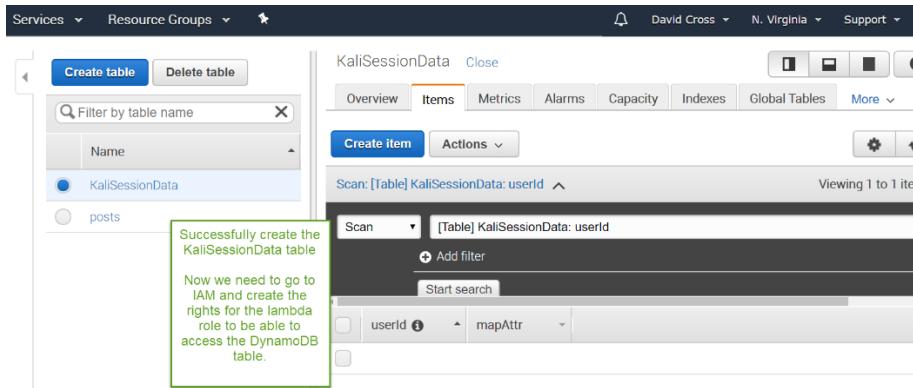
Green arrows point from the first two steps to the respective input fields ('Table name' and 'Primary key'). A green arrow also points from the third step towards the bottom of the page.

The screenshot shows the 'Create table' wizard in the AWS DynamoDB console. In the 'Table settings' step, a primary key named 'userId' is defined as a String type. A note states: 'You do not have the required role to enable Auto Scaling by default. Please refer to documentation.' A green callout box points to the 'Create' button, which is highlighted with a green border and contains the text: 'Click Create to build the table. (It will take a moment to complete)'.

The screenshot shows the 'Overview' tab for the 'KaliSessionData' table. It displays a message: 'Table is being created'. A green callout box points to this message with the text: '1) Table is being created
2) The next thing we'll need to do is add an item'.

The screenshot shows the 'Create item' interface for the 'KaliSessionData' table. A green callout box points to the 'Create item' button with the text: 'Create a new item to hold session data for the Alexa skill... the userId will hold the userId for the Alexa session.'





The screenshot shows the AWS IAM Roles page. On the left, there's a sidebar with links like Dashboard, Groups, Users, Policies, Identity providers, Account settings, Credential report, and Encryption keys. The main content area has a heading 'What are IAM roles?'. It explains that IAM roles are a secure way to grant permissions to entities. A green box highlights the text 'IAM roles issue keys that are valid for short durations' and a green arrow points down to the 'Additional resources' section.

What are IAM roles?

IAM roles are a secure way to grant permissions to entities that you trust. Examples of entities include the following:

- IAM user in another account
- Application code running on an EC2 instance that needs to perform actions on AWS resources
- An AWS service that needs to act on resources in your account to provide its features
- Users from a corporate directory who use AWS services

IAM roles issue keys that are valid for short durations. This is a more secure way to grant access.

Additional resources:

- IAM Roles FAQ
- IAM Roles Documentation
- Tutorial: Setting Up Cross Account Access
- Common Scenarios for Roles

The screenshot shows the 'Create role' page. The sidebar is identical to the previous one. The main area has 'Create role' and 'Delete role' buttons. Below them is a search bar and a table with columns for 'Role name', 'Description', and 'Trusted entities'. A green box highlights the first row, which is a Service-Linked role for AWS Support. A green arrow points down to the 'lambda_basic_execution' row, which is highlighted in blue.

Role name	Description	Trusted entities
AWSServiceRoleForSup...	Enables resource access for AWS to provide billing, ...	AWS service: support (Service-Linked role)
You may or may not have other roles available but you'll want to make a copy of the lambda role or, if this is your only lambda function, you can modify the lambda role.	ed Advisor Service to help...	AWS service: trustedadvisor (Service-Linked role)
		AWS service: lambda
lambda_basic_execution		AWS service: lambda

The screenshot shows the details of a specific IAM role. The sidebar is identical. The main area shows the role's ARN, path, creation time, and maximum session duration. Below this is the 'Permissions' tab, which is selected. A green arrow points to the 'Attach policies' button. A green box highlights the 'Permissions policies (1 policy applied)' section, which lists the attached policy 'oneClick_lambda_basic_execution_1487667798728'. A green arrow points down to the 'Inline policy' section.

Permissions

Permissions policies (1 policy applied)

Attach policies

Policy name

oneClick_lambda_basic_execution_1487667798728

Inline policy

Add permissions to lambda_basic_execution

Attach Permissions

[Create policy](#)

Filter policies

Policy name	Description
AmazonDynamoDBFullAccess	Provides full access to Amazon DynamoDB
AmazonDynamoDBReadOnlyAccess	Provides full access to Amazon DynamoDB
AmazonDynamoDBReadWriteAccess	Provides read only access to Amazon DynamoDB
AWSLambdaDynamoDBExecutionRole	AWS managed None

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor or JSON editor.

Visual editor JSON

Expand all | Collapse all

▼ Select a service

► Service [Choose a service](#)

Choose the service which will be DynamoDB

Actions Choose a service before defining actions

Resources Choose actions before applying resources

Request conditions Choose actions before specifying conditions

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON.

Visual editor JSON

Import

Expand all | Collapse all

▼ Select a service

► Service Select a service below

close Enter service name

DynamoDB [Select](#)

Search for DynamoDB and select it from the service menu

Actions Choose a service before defining actions

Services ▾ Resource Groups ▾ ⌂ David Cross ▾ Global ▾

► Service DynamoDB

Actions Specify the actions allowed in DynamoDB ⓘ

close Filter actions

Manual actions (add actions)

All DynamoDB actions (dynamodb:*)

Access level

List

- ListBackups ⓘ
- ListGlobalTables ⓘ
- ListTables ⓘ

Read

- BatchGetItem ⓘ
- ConditionCheckItem ⓘ
- DescribeBackup ⓘ
- DescribeContinuousBackups ⓘ
- DescribeGlobalTable ⓘ
- DescribeGlobalTableSettings ⓘ
- DescribeLimits ⓘ
- DescribeReservedCapacity ⓘ
- DescribeStream ⓘ
- DescribeTable ⓘ
- DescribeTimeToLive ⓘ
- GetItem ⓘ
- GetRecords ⓘ
- GetShardIterator ⓘ
- ListStreams ⓘ
- Query ⓘ
- Scan ⓘ

Write

- BatchWriteItem ⓘ
- CreateBackup ⓘ
- CreateGlobalTable ⓘ
- CreateTable ⓘ
- DeleteBackup ⓘ
- DeleteItem ⓘ
- DeleteTable ⓘ
- PurchaseReservedCapacity ⓘ
- PutItem ⓘ
- RestoreTableFromBackup ⓘ
- RestoreTableToPointInTime ⓘ
- TagResource ⓘ
- UntagResource ⓘ
- UpdateContinuousBackups ⓘ
- UpdateGlobalTable ⓘ
- UpdateGlobalTableSettings ⓘ
- UpdateItem ⓘ
- UpdateTable ⓘ
- UpdateTimeToLive ⓘ

Switch to deny permissions

Expand all | Collapse

Expand all Action segments

Resource Groups ▾ ⌂ David Cross ▾ Global ▾ Support

Read (4 selected)

- BatchGetItem ⓘ
- ConditionCheckItem ⓘ
- DescribeBackup ⓘ
- DescribeContinuousBackups ⓘ
- DescribeGlobalTable ⓘ
- DescribeGlobalTableSettings ⓘ
- DescribeLimits ⓘ
- DescribeReservedCapacity ⓘ
- DescribeStream ⓘ
- DescribeTable ⓘ
- DescribeTimeToLive ⓘ
- GetItem ⓘ
- GetRecords ⓘ
- GetShardIterator ⓘ
- ListStreams ⓘ
- Query ⓘ
- Scan ⓘ

Write

- BatchWriteItem ⓘ
- CreateBackup ⓘ
- CreateGlobalTable ⓘ
- CreateTable ⓘ
- DeleteBackup ⓘ
- DeleteItem ⓘ
- DeleteTable ⓘ
- PurchaseReservedCapacity ⓘ
- PutItem ⓘ
- RestoreTableFromBackup ⓘ
- RestoreTableToPointInTime ⓘ
- TagResource ⓘ
- UntagResource ⓘ
- UpdateContinuousBackups ⓘ
- UpdateGlobalTable ⓘ
- UpdateGlobalTableSettings ⓘ
- UpdateItem ⓘ
- UpdateTable ⓘ
- UpdateTimeToLive ⓘ

Select the following items from the Read section

Resource Groups ▾ ⌂ David Cross ▾ Global ▾ Support

Write (5 selected)

- BatchWriteItem ⓘ
- CreateBackup ⓘ
- CreateGlobalTable ⓘ
- CreateTable ⓘ
- DeleteBackup ⓘ
- DeleteItem ⓘ
- DeleteTable ⓘ
- PurchaseReservedCapacity ⓘ
- PutItem ⓘ
- RestoreTableFromBackup ⓘ
- RestoreTableToPointInTime ⓘ
- TagResource ⓘ
- UntagResource ⓘ
- UpdateContinuousBackups ⓘ
- UpdateGlobalTable ⓘ
- UpdateGlobalTableSettings ⓘ
- UpdateItem ⓘ
- UpdateTable ⓘ
- UpdateTimeToLive ⓘ

Select the following items from the Write segment...

Resources You chose actions that require the stream resource type.

Resource Groups ▾ 🔍

David Cross ▾ Global ▾ Support

DeleteBackup ⓘ TagResource ⓘ UpdateTimeToLive ⓘ

DeleteItem ⓘ UntagResource ⓘ

DeleteTable ⓘ UpdateContinuousBackups ⓘ

▶ Resources You chose actions that require the stream resource type.
You also chose actions that require the table resource type.

Request conditions Specify request conditions (optional)

Check out the warnings on Policy... by expanding this segment and move on to the next diagram

Add additional permissions

Cancel Review policy

DynamoDB (3 actions)

▶ Service DynamoDB

▶ Actions Read

- GetItem
- GetRecords
- Query
- Scan

Write

- BatchWriteItem
- CreateTable
- DeleteItem
- PutItem
- UpdateItem

▼ Resources Specific close All resources

1) Select the All resources option from Resources
2) Select Preview Policy

Services ▾ Resource Groups ▾ 🔍

David Cross ▾ Global ▾ Support

Review policy

Name* AlexaSessionDBReadWrite

Use alphanumeric and '+-, @_-' characters. Maximum 128 characters.

Description This pol should be further restricted once its working.]

Maximum 1000 characters. Use alphanumeric and '+-, @_-' characters.

Enter a name and description being careful to not use quotes or single quotes in the description.

Then scroll down to finish policy creation.

Summary

Service	Access level	Resource	Request con
Allow (1 of 174 services) Show remaining 173	Limited: Read, Write	All resources	None

DynamoDB

Description: This pol should be further restricted once its working.

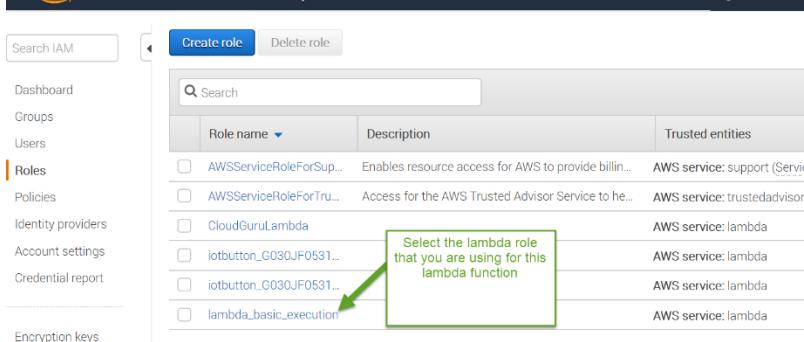
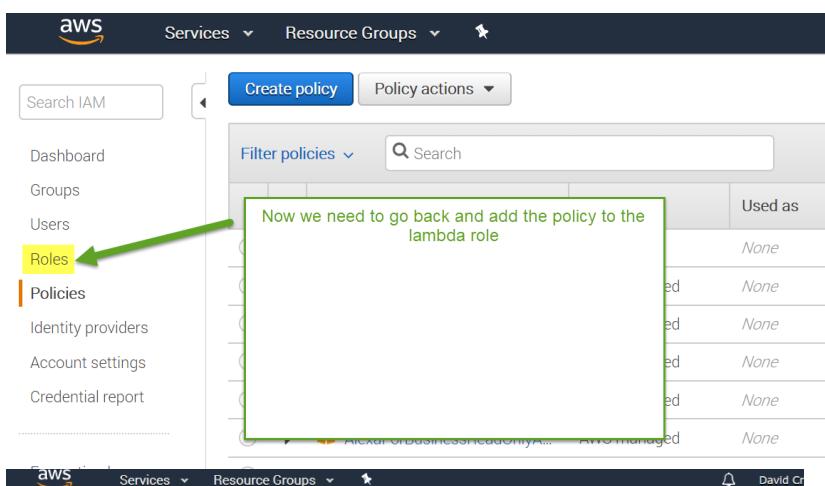
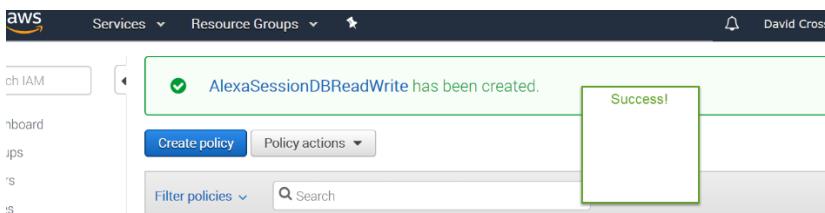
Maximum 1000 characters. Use alphanumeric and '+,-,@,' characters.

Summary

Service	Access level	Resource	Request con...
Allow (1 of 174 services) Show remaining 173	Limited: Read, Write	All resources	Create the policy
DynamoDB	Limited: Read, Write		None

Create the policy

Cancel Previous Create policy



Roles > lambda_basic_execution

Summary

Role ARN	arn:aws:iam::926433000384:role/lambda_basic_execution
Role description	Edit
Instance Profile ARNs	Edit
Path	/
Creation time	2017-02-21 02:03 MDT
Maximum CLI/API session duration	1 hour Edit

Permissions **Trust relationships** **Revoke sessions**

▼ Permissions policies (1 policy applied)

Attach policies

Policy name ▾ Policy type ▾

Add permissions to lambda_basic_execution

Attach Permissions

Create policy

Filter policies ▾ Search Showing 438 results

	Policy name	Type	Used as	Description
<input type="checkbox"/>	AlexaForBusinessFull...	AWS managed	None	Grants full access to AlexaForBusiness resources an...
<input type="checkbox"/>	AlexaForBusinessGat...	AWS managed	None	Provide gateway execution access to AlexaForBusine...
<input type="checkbox"/>	AlexaForBusinessRea...	AWS managed	None	Provide read only access to AlexaForBusiness services
<input checked="" type="checkbox"/>	AlexaSessionDBRead...	Customer managed	None	1) Select the broad policy we created (that needs to be restricted more later) 2) Click Attach Policy to give this policy to the lambda_basic_execution role This policy should be further restricted once its working.

Cancel **Attach policy**

Roles > lambda_basic_execution

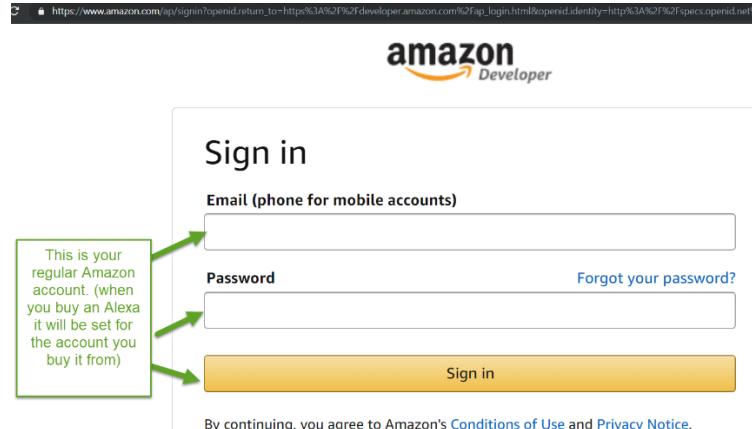
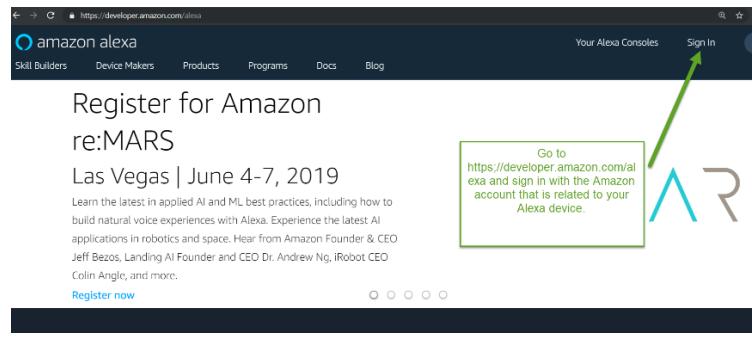
Summary

Policy AlexaSessionDBReadWrite has been attached for the lambda_basic_execution.

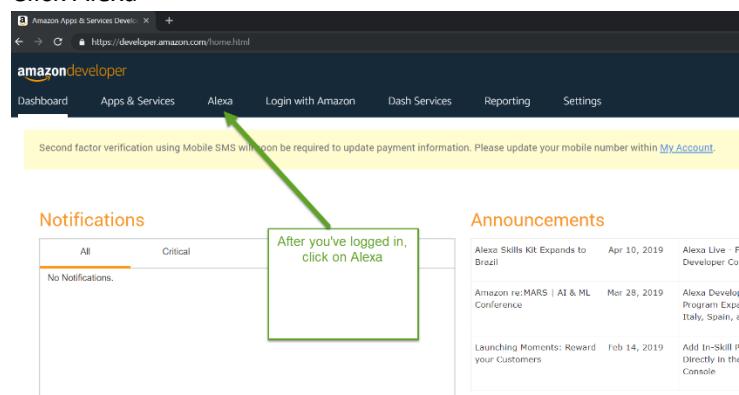
Step Five (Creating the Alexa Voice configuration)

Step Five phase 1

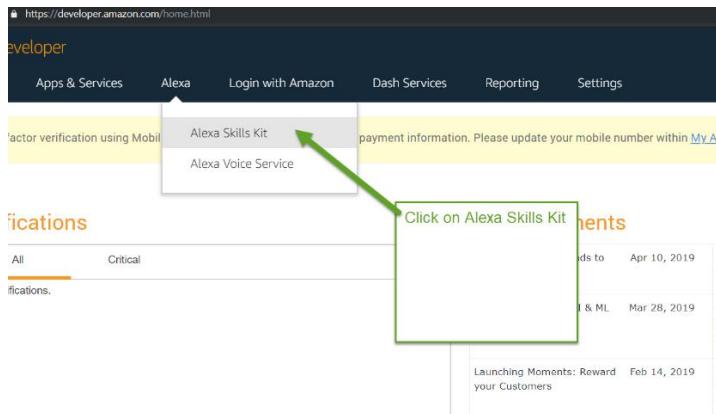
Log into the Alexa Skill site using the account you set up in the pre-requisite steps.



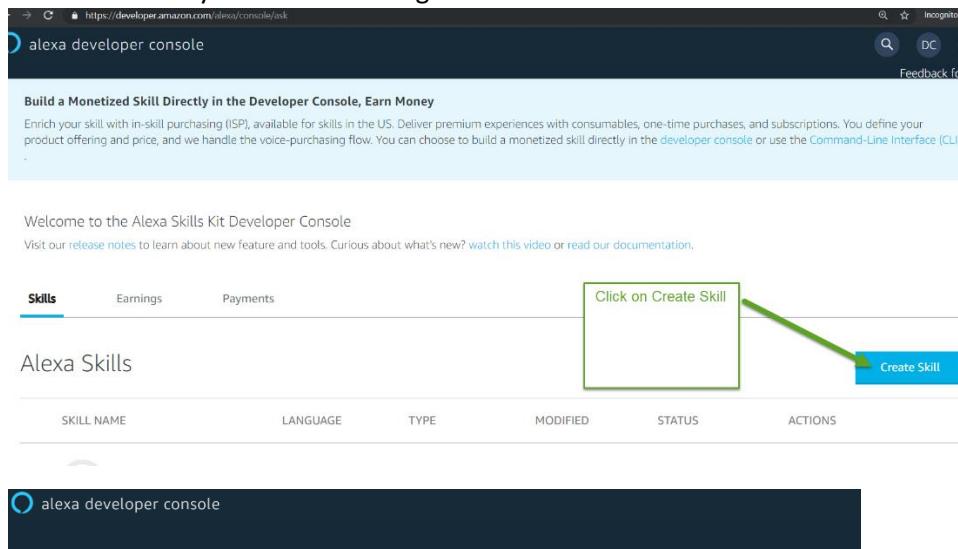
Click Alexa



Go to the Alexa Skills Kit



Create a skill as you see in the image below:



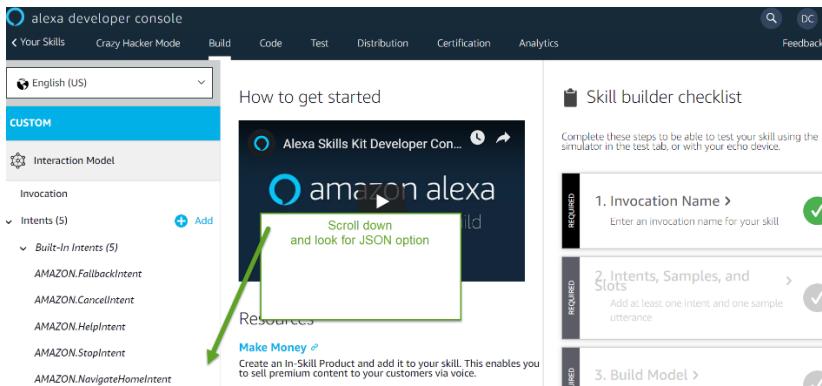
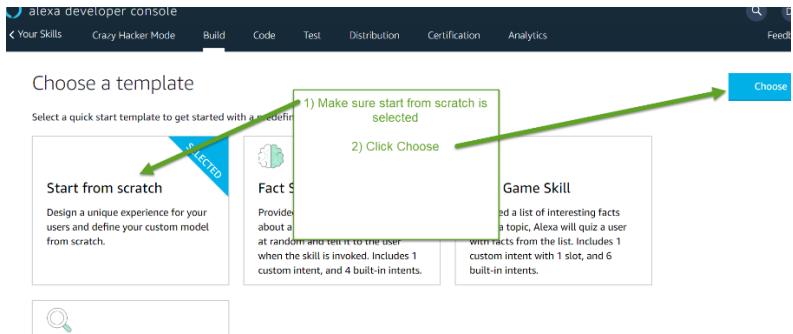
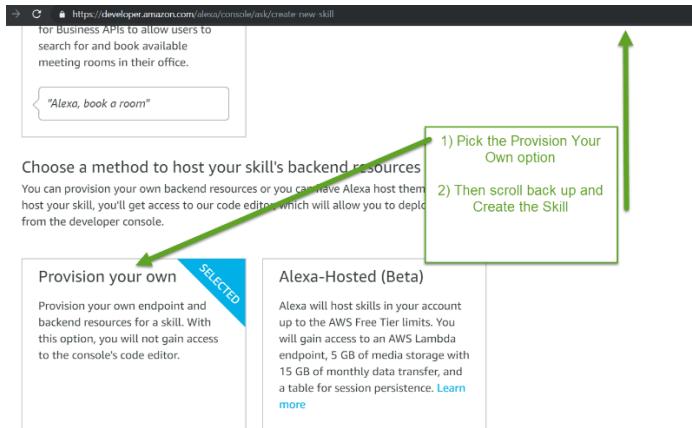
Create a new skill

Skill name
Crazy Hacker Mode
17/50 characters

Default language
English (US)

Pick a invocation name
In this case the skill only supports English.
Scroll down

Choose a model to add to your skill
There are many ways to start building a skill. You can design your own custom model or start with a pre-built model. Pre-built models are interaction models that contain a package of intents and utterances that you can add to your skill.



The screenshot shows the Alexa Developer Console interface. On the left, there's a sidebar with a 'CUSTOM' tab selected. Under 'Interaction Model', it lists 'Invocation', 'Intents (5)', and 'Slot Types (0)'. A green arrow points from the 'Slot Types (0)' link to a callout box containing the text 'Click on JSON Editor'. Another green arrow points from the 'JSON Editor' link to the same callout box. The main content area has a large 'amazon alexa' logo and the text 'Developer Console: Build'. Below the logo, there's a callout box with the text 'Create an In-Skill Product and add it to your skill. This e...' and a 'Documentation' link.

Alexa Developer Console

CUSTOM

Interaction Model

Invocation

Intents (5)

+ Add

Built-In Intents (5)

AMAZON.FallbackIntent

AMAZON.CancelIntent

AMAZON.HelpIntent

AMAZON.StopIntent

AMAZON.NavigateHomeIntent

Slot Types (0)

+ Add

JSON Editor

Click on JSON Editor

Create an In-Skill Product and add it to your skill. This e...

Documentation

Refer to our technical documents for detailed guides or custom skills.

Sample Alexa Projects

Whatever your experience, you can get started quickly.

We want to replace this field with the JSON file from the HackerMode 2 open source project

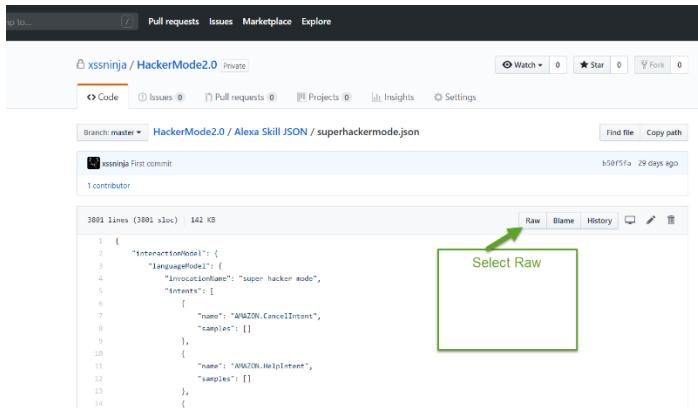
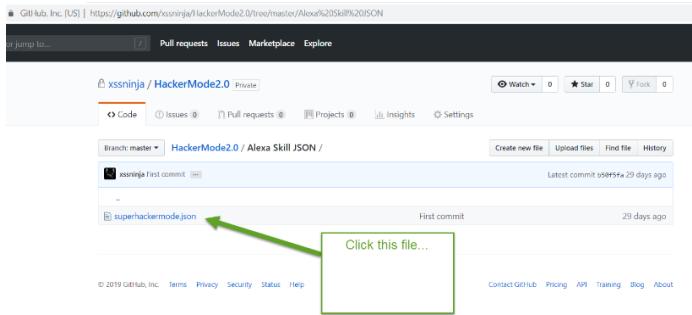
Drag and drop a .json file

```
1  "version": "1.0",  
2  "invocationModel": {  
3    "language": "en-US",  
4    "invocations": [  
5      "intents": [  
6        {  
7          "name": "AMAZON.FallbackIntent",  
8          "samples": []  
9        },  
10        {  
11          "name": "AMAZON.CancelIntent",  
12          "samples": []  
13        },  
14        {  
15          "name": "AMAZON.HelpIntent",  
16          "samples": []  
17        },  
18        {  
19          "name": "AMAZON.StopIntent",  
20          "samples": []  
21        },  
22        {  
23          "name": "AMAZON.NavigateHomeIntent",  
24          "samples": []  
25        }  
26      ],  
27      "types": {}  
28    }  
29  }
```

Get the JSON file here at
Github
[xssninja/HackerMode2.0](#)

Click on the Alexa Skill JSON
folder

Commit	Author	Date
Adding a line to make the alexopen.py executable	...	29 days ago
AWS Lambda	First commit	29 days ago
Alexa Skill JSON	First commit	29 days ago
Kali	Adding a line to make the alexopen.py executable	12 days ago
PHP Site	First commit	29 days ago
LICENSE	Initial commit	a month ago



https://raw.githubusercontent.com/xssninja/HackerMode2.0/master/Alexa%20Skill%20JSON/superhackermode.json?token=Ayxd0IDQ8RZM-DU

```

1 {
2   "interactionModel": {
3     "languageModel": {
4       "invocationName": "super hacker mode",
5       "intents": [
6         {
7           "name": "AMAZON.CancelIntent",
8           "samples": []
9         },
10        {
11          "name": "AMAZON.HelpIntent",
12          "samples": []
13        },
14      ]
15    }
16  }
17}

```

A screenshot of the raw JSON file content. A green arrow points from a callout box labeled 'Press Ctrl-A and Ctrl-C select all and copy to the clipboard' to the beginning of the JSON code.

Alexa Developer Console +

https://developer.amazon.com/alexa/console/ask/build/custom/amzn1.ask.skill.dd150dc5-6d00-418e-8e91-6e105a4cfce/development/en_US/json-editor

CUSTOM

Interaction Model

Click here to learn more about the schema definition for interaction models.

Invocation

Intents (5) + Add

- Built-In Intents (5)**
 - AMAZON.FallbackIntent
 - AMAZON.CancelIntent
 - AMAZON.HelpIntent
 - AMAZON.StopIntent
 - AMAZON.NavigateHomeIntent

Slot Types (0) + Add

JSON Editor

```

1  {
2    "interactionModel": {
3      "languageModel": {
4        "invocationName": "crazy hacker"
5      }
6      "intents": [
7        {
8          "name": "AMAZON.FallbackIntent",
9          "samples": []
10         },
11        {
12          "name": "AMAZON.CancelIntent",
13          "samples": []
14         },
15        {
16          "name": "AMAZON.HelpIntent",
17          "samples": []
18         },
19        {
20          "name": "AMAZON.StopIntent",
21          "samples": []
22         },
23        {
24          "name": "AMAZON.NavigateHomeIntent",
25          "samples": []
26         }
27     ],
28     "types": []
}

```

Interfaces

Endpoint

Going back to the Alexa skill page prepare to select the JSON data and replace it with the JSON from Hacker Mode that's in the copy buffer.

(see next image)

Alexa developer console +

https://developer.amazon.com/alexa/console/ask/build/custom/amzn1.ask.skill.dd150dc5-6d00-418e-8e91-6e105a4cfce/development/en_US/json-editor

CUSTOM

Interaction Model

Click here to learn more about the schema definition for interaction models.

Invocation

Intents (5) + Add

- Built-In Intents (5)**
 - AMAZON.FallbackIntent
 - AMAZON.CancelIntent
 - AMAZON.HelpIntent
 - AMAZON.StopIntent
 - AMAZON.NavigateHomeIntent

Slot Types (0) + Add

JSON Editor

Drag and drop a .json file

```

{
  "name": {
    "value": "content_type options"
  },
  "name": {
    "value": "x. d. n. s. prefetch control"
  },
  "name": {
    "value": "d. n. s. prefetch"
  },
  "name": {
    "value": "x. frame options"
  },
  "name": {
    "value": "x. x. s. s. protection"
  },
  "name": {
    "value": "cross site scripting protection"
  }
}

```

Interfaces

Endpoint

After selecting all the text in this box using Ctrl-A and replacing it with the JSON in the copy buffer using Ctrl-C, you should see a portion of the new JSON in your view

Alexa developer console +

https://developer.amazon.com/alexa/console/ask/build/custom/amzn1.ask.skill.dd150dc5-6d00-418e-8e91-6e105a4cfce/development/en_US/json-editor

CUSTOM

Interaction Model

Click here to learn more about the schema definition for interaction models.

Invocation

Intents (5) + Add

- Built-In Intents (5)**
 - AMAZON.FallbackIntent
 - AMAZON.CancelIntent
 - AMAZON.HelpIntent
 - AMAZON.StopIntent
 - AMAZON.NavigateHomeIntent

Slot Types (0) + Add

JSON Editor

Drag and drop a .json file

1) Scroll back to the top of the main page of the Alexa developer console (you should see the Save and Build Model buttons)

2) Click Save

3) Click Build Model and wait to see if there are any errors. It should complete with warnings but no errors.

Save Model

Build Model

"content type options"

"x. d. n. s. prefetch control"

"d. n. s. prefetch"

"x. frame options"

"x. x. s. s. protection"

"cross site scripting protection"

alex developer console

Build Code Test Distribution Certification Analytics

English (US) Save Model Building Utterances

CUSTOM Interaction Model Click here to learn more

Invocation

Intents (36) + Add

- ASCIIEncodingIntent
- CHARItem
- HexEncodingIntent
- CHARItem

JSON Editor While building you'll see some basic warnings but no errors. Drag and dr

```
{
  "version": "1.0",
  "intents": [
    {
      "name": "SetRPortIntent",
      "slots": [
        {
          "name": "RPport",
          "value": "age"
        }
      ],
      "utterances": [
        {
          "text": "Set RPort to age"
        }
      ]
    },
    {
      "name": "SetExploitNameIntent",
      "slots": [
        {
          "name": "ExploitName",
          "value": "cache control"
        }
      ],
      "utterances": [
        {
          "text": "Set Exploit Name to cache control"
        }
      ]
    }
  ]
}
```

UnreferencedSlot The slot "RPport" in intent "SetRPortIntent" referenced in any slot or intent sample. Saturday, April 15, 2020

UnreferencedSlot The slot "ExploitName" in intent "SetExploitNameIntent" not referenced in any slot or intent sample. Saturday, April 15, 2020

BuiltInIntentAddedToModel

alex developer console

Your Skills Crazy Hacker Mode Build Code Test Distribution Certification Analytics

English (US) Save Model Build Model

CUSTOM Interaction Model

Invocation

Intents (36) + Add

- ASCIIEncodingIntent
- CHARItem
- HexEncodingIntent
- CHARItem

JSON Editor Go back to Your Skills just to make sure everything worked and the skill has been added... Drag and dr

```
{
  "version": "1.0",
  "intents": [
    {
      "name": "SetRPortIntent",
      "slots": [
        {
          "name": "RPport",
          "value": "age"
        }
      ],
      "utterances": [
        {
          "text": "Set RPort to age"
        }
      ]
    },
    {
      "name": "SetExploitNameIntent",
      "slots": [
        {
          "name": "ExploitName",
          "value": "cache control"
        }
      ],
      "utterances": [
        {
          "text": "Set Exploit Name to cache control"
        }
      ]
    }
  ]
}
```

UnreferencedSlot The slot "RPport" in intent "SetRPortIntent" referenced in any slot or intent sample. Saturday, April 15, 2020

UnreferencedSlot The slot "ExploitName" in intent "SetExploitNameIntent" not referenced in any slot or intent sample. Saturday, April 15, 2020

BuiltInIntentAddedToModel

https://developer.amazon.com/alexa/console/ask

Build a Monetized Skill Directly in the Developer Console, Earn Money

Welcome to the Alexa Skills Kit Developer Console

Skills Earnings Payments

Alexa Skills

SKILL NAME	LANGUAGE	MODIFIED	STATUS	ACTIONS
Crazy Hacker Mode	English (US)	2019-04-13	Build Successful	View Skill ID Go to build

Skills Earnings Payments

Alexa Skills

SKILL NAME	LANGUAGE	MODIFIED	STATUS	ACTIONS
Crazy Hacker Mode	English (US)	2019-04-13	Build Successful	View Skill ID Go to build

Looks like it worked! Click on the skill ID because now we need to gather some data from the skill detail to connect the lambda function

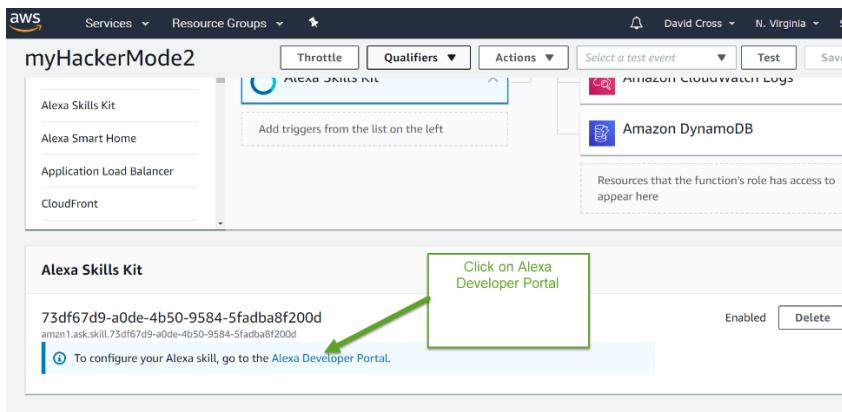
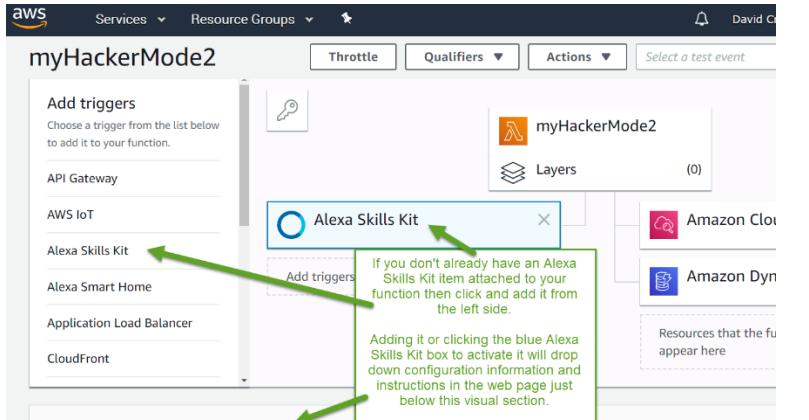
If you make any new changes, you will need to rebuild your model for them to take effect.

Step Five Phase 2 (Connecting your Lambda function and your Alexa skill together)

Log into the AWS console using your account created in the prerequisite steps.

The image consists of three vertically stacked screenshots of the AWS Management Console:

- Screenshot 1: Alexa Developer Console**
A screenshot of a web browser showing the Alexa Developer Console. A green arrow points from the top of the browser window to the address bar, which displays "aws.amazon.com/console". A callout box with the text "Open a new tab and go to your AWS console and login if you're not already logged in." is overlaid on the left side of the screen.
- Screenshot 2: AWS Management Console Home**
The AWS Management Console home page. A green arrow points from the top right of the header to the "N. Virginia" region selector. Another green arrow points from the search bar on the left to the "lambda" search result. A callout box with the text "North Virginia is the location supported by Alexa based lambda skills for my part of the world and this is my default login Region. If Lambda for Alexa is supported in a different region for you select the region that Amazon says works for your language and part of the world." is overlaid on the search results.
- Screenshot 3: AWS Lambda Functions**
The AWS Lambda Functions page, showing four functions: "myServerlessWebsite", "myHTMLEncode", "myFunc", and "myHackerMode2". A green arrow points from the bottom left of the table to the "myHackerMode2" function name. A callout box with the text "Here was the lambda function we created previously. Click the name and that will put you into the configuration and coding screen for it... We will need to link it to the Alexa template we created with the JSON file a moment ago." is overlaid on the function list.



https://www.amazon.com/ap/signin?openid.pape.preferred_auth_policies=Singlefactor&clientContext=135-0356453-6700763&openid.pape.max_auth_age=7200&openid.id_token=73df67d9-a0de-4b50-9584-5fadba8f200d_amzn1.ask.skill.73df67d9-a0de-4b50-9584-5fadba8f200d

Sign in

Switch accounts

David Cross

Sign in below even though you're probably still signed in... this will take you to the right option to match your skill name with the lambda

Keep me signed in. Details ▾

By continuing, you agree to Amazon's [Conditions of Use](#) and [Privacy Notice](#).

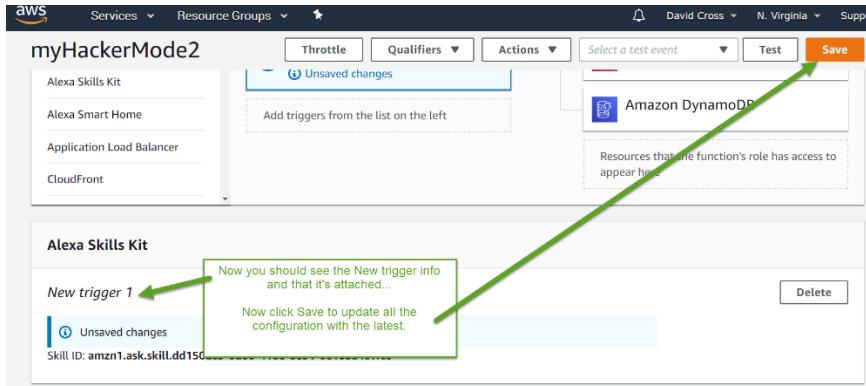
SKILL NAME	LANGUAGE	MODIFIED	STATUS	ACTIONS
Crazy Hacker Mode	English (US)	2019-04-13	In Development	Analytics Edit Delete

The screenshot shows the Alexa Skills Kit dashboard. A skill named "Crazy Hacker Mode" is listed. A context menu is open over the "Skill ID" column for this skill. The menu includes options: "Copy" (highlighted with a red box), "Search Google for 'amzn1.ask.skill.dd150dc5-6d00-418e-8e91...'", and "Print...".

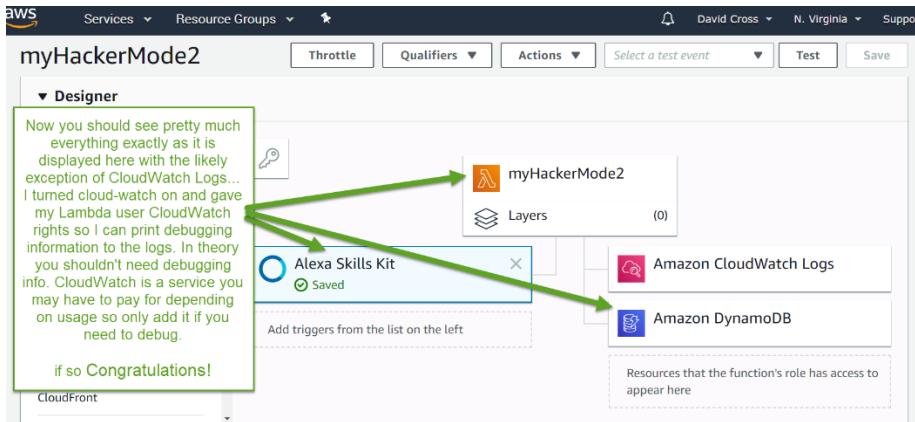
Step Five Phase 3 (Add a Lambda trigger for your Alexa Skill)

The screenshot shows the AWS Lambda function configuration page for "myHackerMode2". In the "Configure triggers" section, the "Alexa Skills Kit" trigger is selected. A callout box points to a note at the bottom of the screen: "Skill ID verification is an easy way to verify the Skill ID in an incoming request from a Skill. To set this up, enter the Skill ID (also called Application ID) of your skill located in your Alexa Skills Kit dashboard. Learn more."

The screenshot shows the AWS Lambda function configuration page for "myHackerMode2". Under "Skill ID verification", the "Enable (recommended)" option is selected. The "Skill ID" input field contains "amzn1.ask.skill.dd150dc5-6d00-418e-8e91...". A callout box points to the "Skill ID" field with instructions: "1) Paste in the skill ID we copied from the Alexa skill side a minute ago" and "2) Then click Add".

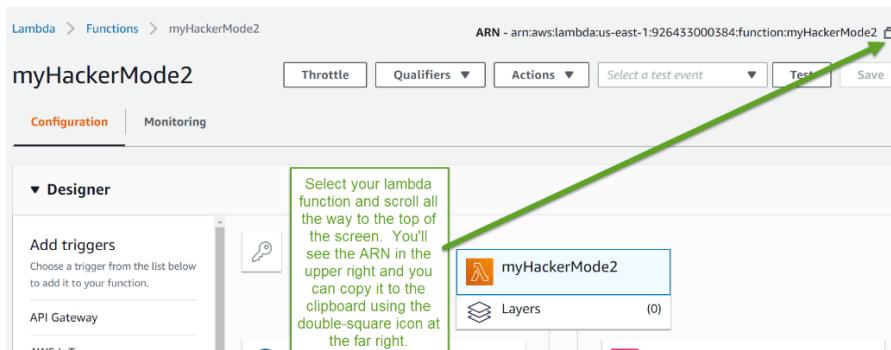


You should see things pretty much as below.



Step Five Phase 4 (Connecting the ARN)

Connecting the Lambda function and the Skill together.



Now go back to the Alexa Skills site and prepare to connect the Lambda and the Skill together.

The screenshot shows two main sections of the Alexa developer console:

- Skill builder checklist:** A sidebar on the right containing three required steps:
 - 1. Invocation Name:** Enter an invocation name f
 - 2. Intents, Samples, & Slots:** Add at least one intent and utterance
 - 3. Build Model >** Successfully build your inte
- JSON Editor:** The main workspace where the skill's configuration is being edited. It includes tabs for **Interfaces**, **Endpoint** (which is highlighted with a green arrow), and **Intent History**. A callout box with a green border and arrow points to the **Endpoint** tab with the text: "Click the Endpoint configuration".

The URL in the browser bar is: https://developer.amazon.com/alexasdk/ask/build/custom/amzn1.ask.skill.dd150dc5-6d00-418e-8e91-6e105a4c1fce/development/en_US/dashboard

Interaction Model

vocation
Intents (37) **+ Add**

- ASCIIEncodingIntent **CHARItem**
- CHARItem **CHARItem**
- HexEncodingIntent **CHARItem**
- CHARItem **CHARItem**
- HTMLEncodingIntent **CHARItem**
- CHARItem **HTTPResponseIntent**
- HTTPResponseIntent **RCItem**

Endpoint

The Endpoint will receive POST requests when a user interacts with your Alexa Skill. This endpoint defines the parameters that your service can use to perform logic and generate a JSON-formatted response. You can host your own HTTPS web service endpoint as long as it meets the requirements described [here](#).

Service Endpoint Type

Select how you will host your skill's service endpoint.

AWS Lambda ARN (Recommended)

HTTPS (Optional)

Click on AWS Lambda ARN

Amazon needs a bit more information about where the Lambda is running. This will normally be the North VA area if you are in the USA as that region has both Dynamo DB support and Alexa Lambda support.

https://developer.amazon.com/alexa/console/ask/build/custom/amzn1.ask.skill.dd150dc5-6d00-418e-8e91-6e105a4c1fce/development/en_US/endpoint

Service Endpoint Type

Select how you will host your skill's service endpoint.

AWS Lambda ARN (Recommended)

Your Skill ID amzn1.ask.skill.dd150dc5-6d00-418e-8e91-6e105a4c1fce

[Copy to Clipboard](#)

Default Region (Required)

arn:aws:lambda:<location>:<aws_account_id>:function:<

North America (Optional)

arn:aws:lambda:us-east-1:<aws_account_id>:function:<

Europe and India (Optional)

arn:aws:lambda:eu-west-1:<aws_account_id>:function:<

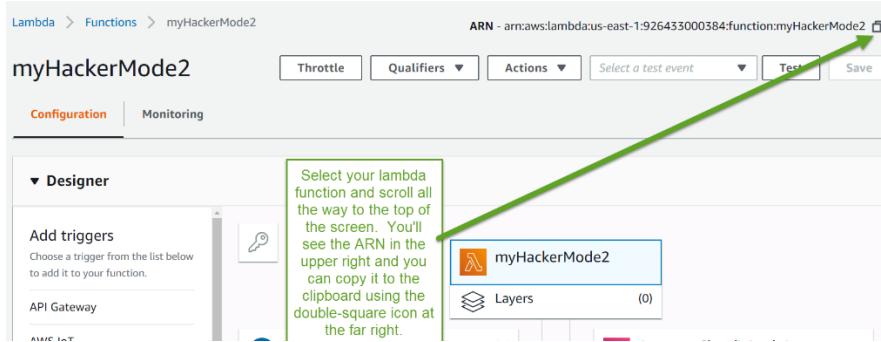
Far East (Optional)

arn:aws:lambda:ap-northeast-1:<aws_account_id>:funct

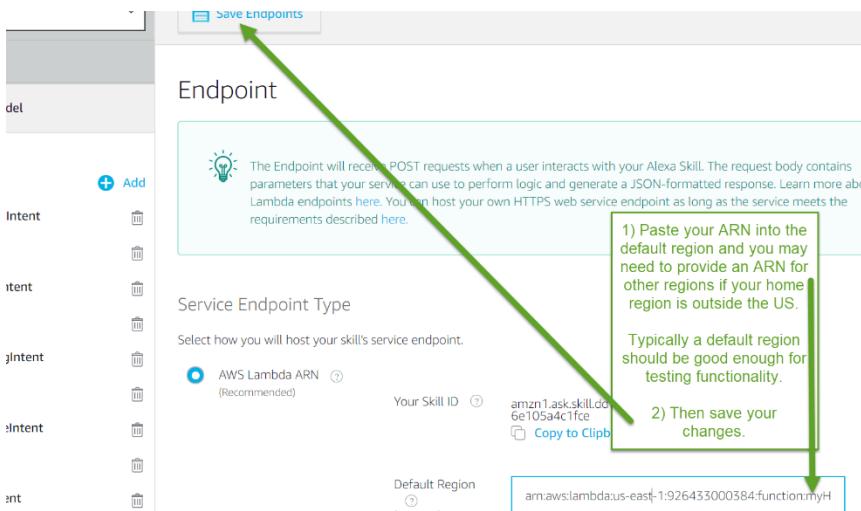
Enter the ARNs for the regions that you'll be working with and or that support your country or Region.

You'll need to get the lambda arn from the Lambda configuration so switch over to that side to grab the ARN.

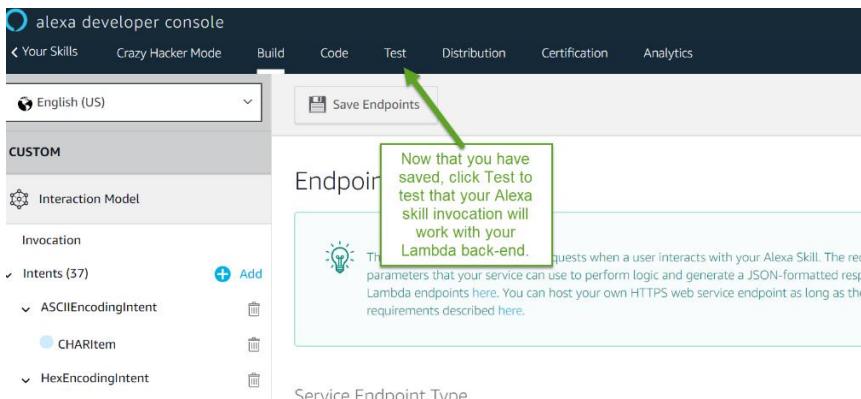
Previously we copied the ARN from the AWS Lambda information but here is the place to copy it from if you missed that step.



Paste that ARN from the Lambda side into the Skill account information screen as seen below:



Now that we have them connected we can test the connection...



Install Step Six (Install Kali Side of Hacker Mode 2.0)

Prerequisite (Your Kali will need both Python 3 and 2.7! Why? Because of install dependencies for 3rd party python libraries that will break otherwise) Default Kali should have 2.7 and 3 pre-installed, but if you are building a container you will need to include it.)

Phase 1 Install nmap library for Python

Install nmap for python 2.7:

- **pip install python-nmap**

(<http://xael.org/norman/python/python-nmap/>)

Phase 2 Install fixed MSFRpcd Python library

- **python msfrpcd.py install**

Phase 3 Install Pymetasploit from original source

See the following steps:

2. **pip install pymetasploit**

Phase 4 Re-install Pymetasploit from XSSNinja Github

The original pymetasploit has a flaw that is fixed by the Mikaayenson version of the PyMetasploit repo. I forked it because I want to add more custom features to it, and I want it to be available in the event someone patches the others with unexpected “fixes”, but you can install the corrected code easily from the fork on my Github as shown below:

```
root@kali:~/Downloads# pip install git+https://github.com/xssninja/pymetasploit
Collecting git+https://github.com/xssninja/pymetasploit
  Cloning https://github.com/xssninja/pymetasploit to /tmp/pip-req-build-xeb8_0
    Downloading https://files.pythonhosted.org/packages/8a/20/6eca72d1a5830336f84acab8198e5a3f4715cd1c7fc36d3cc7f7185091/mspack-python-0.5.6.tar.gz (138kB)
     100% |██████████| 143kB 1.9MB/s
Building wheels for collected packages: pymetasploit, msgpack-python
  Running setup.py bdist_wheel for pymetasploit ... done
    Stored in directory: /tmp/pip-ephem-wheel-cache-s1lWfz/wheels/1f/6b/99/c9ad9822e15b21772e7b4cae0916050e393a6b730e45cc87d5
  Running setup.py bdist_wheel for msgpack-python ... done
    Stored in directory: /root/.cache/pip/wheels/d5/de/86/7fa56fd12511be47ea0800f3502bc879df1e63ab168e0406
Successfully built pymetasploit msgpack-python
Installing collected packages: msgpack-python, pymetasploit
Successfully installed msgpack-python-0.5.6 pymetasploit-1.1
root@kali:~/Downloads#
```

Phase 5 Install AlexaPwn.py file

- Grab AlexaPwn.py from Github and put it where you want.

Phase 6 Install and configure AlexaPwn6cfg.py file

- Grab AlexaPwn6cfg.py from Github and put it in the same directory as alexapwn.py.
- Configure AlexaPwn6cfg.py with the paths to your Alexa-bound PHP queue path and the subnet prefix of your target subnet.
- Compile Alexapwn6cfg.py to Alexapwn6cfg.pyc or it won't be importable into alexapwn.py
- Edit the alexapwn6cfg.py file to point to the hosted php files Kali2Alexa.php and Kali_Read.php

```
root@kali:~/Documents/alexapwn# gedit alexapwn6cfg.py
```

```
alexapwn6cfg.py
~/Documents/alexapwn
```

```
#!/usr/bin/env python
kalicfg = {'Kali2Alexa': 'https://www.10.10.10.10/ninja/Kali2Alexa.php',
           'Kali_Read': 'https://www.10.10.10.10/ninja/Kali_Read.php',
           'default_lport': '4000',
           'default_msfrpc_port': 55553}
```

- Compile it with Python 2.7 like so:
- `root@kali:~/Documents/alexapwn# python -m py_compile alexapwn6cfg.py`
- Ignoring the extra crap in this directory that you don't have in yours... note that the file in .pyc form should exist and be importable into Alexapwn.py

```
root@kali:~/Documents/alexapwn# ls -scl (built-in)
2to3-2.7          10 json: <module 'json' from 'json'>      alexapwn3.py      alexapwn6.py
activate-global-python-argcomplete  11 json: <module 'argcomplete' from 'argcomplete'>  alexapwn4.py      alexapwn-min.py
alexapwn2cfg.py   12 json: <module 'pprint' from 'pprint'>      alexapwn5a.py    alexapwn_orig.py
alexapwn2.py      13 json: <module 'socket' from 'socket'>      alexapwn5b.py    alexapwn_pyc
alexapwn2-with_new_tags.py  14 json: <module 'os' from 'os'>      alexapwn5.py     originaldev packa
alexapwn3cfg.py   15 json: <module 'socket' from 'socket'>      alexapwn6cfg.py  python2.7/socket.pyc'
alexapwn3cfg.pyc  16 json: <module 'sys' from 'sys'>      alexapwn6cfg.pyc
```

Phase 7 Run MSFRPC daemon

AlexaPwn.py depends on MSFRPCD to run. Start it before launching the skill.

- `msfrpcd -S -f -U msf -P test`

If you want to save this script and chmod it to executable, you can drop it on your desktop or otherwise script it to start to launch MSFRPCD for you.

Phase 8 Start AlexaPwn.py

Run the python side of HackerMode on Kali with Python 2.7. If you have 2.7 as your default Python it should be as easy as:

- **Python alexapwn.py**

Docker Alternative for Kali Install

Install the Docker Container created by James Blackburn for the Kali side of things and avoid having to do any of the Kali config/install stuff.

Here is the URL to the repo: https://hub.docker.com/r/eli23/kali_pwn

Running Hacker Mode2 soup-to nuts

1. Start your Kali instance and make sure you have internet and are in the subnet range you configured in your Lambda. (ifconfig and ping are helpful obviously)
2. Start Msfrpcd in the terminal in Kali if it's not already running: **msfrpcd -S -f -U msf -P test**
3. Wait a second and start: **python alexapwn.py** in the terminal and watch for errors:
 1. If the config file won't load, make sure it's been updated and compiled so Python can load it.
 2. If it warns of MSFrpdc not being ready, make sure you wait a few seconds for the daemon to fully load before running alexapwn.py
 3. If it warns of failures reaching the queue PHP page online, check your path in the config (you may need to recompile the config if you had to make a change). Also, hit the queue page from a browser to be sure your site is up and the path is OK.
4. **Ensure you have permission to scan stuff in your Kali subnet. If you don't you are going to pwn your life!**
5. Ensure your victim machine is online and your subnet is correct for targeting it. Ping is your friend.
IT WILL HAVE TO BE IN THE SAME SUBNET AS YOUR KALI! That is unless you customize the project to target external systems. Beware however, that if you say the wrong IP, or Alexa misunderstands you and you just start pwning a box that's not yours... you'll be in a world of hurt. Don't be that person! You know the song: "Don't go chasin' waterfalls, stick to the rivers and the lakes that you're used to..." You probably look better in colors other than orange if you know what I mean!
6. Make sure Alexa is on, connected, and your are not muted... Invoke the skill by saying "**Alexa, start Hacker Mode**"
7. If the skill responds you can start immediately with hacking a box using an instruction like: "hack IP address 137.121" NOTE THAT THE FIRST TWO IP NUMBERS NOT PRESENT ARE ONES YOU CONFIGURE IN THE ALEXA SKILL CONFIGURATION PORTION. THEY ARE SET IN CONFIG AND NOT EASILY CHANGEABLE BY DESIGN.

8. If the Alexa skill immediately quits it's probably because the configuration for the QUEUE URL is wrong. Check it and fix it in your Lambda runtime variables config which is located below the code window in the lambda code editor window of your AWS account.
9. You should see a command to Nmap a box appear in your Kali and Alexa should play music. If you don't hear music you messed up the lambda config for where to load the MP3 file from. (You can make your own MP3s for it with Audacity, LAME and or some help from a mp3 to mp3 converting website online as Alexa likes a weird variety of MP3 that is special.)
10. You can ask Alexa questions about things like "what runs on port 1433?" "what is the hex encoding for double-quote" "what is HTTP response code 403?" "Rick Roll David" "what is the URL encoding for space" "what runs on port 443?" "what does the header x-xss protection do?" and a million other things and infinite variations on those. If the skill times out then just wake it back up with "Alexa start Hacker Mode" and resume from there.
11. As the hack progresses, and since Alexa a) can't stay open for more than 8 seconds, and b) doesn't know everything that's happening moment to moment during the hack you will need to ask it to check by saying "**how's the hack going?**"
12. Alexa should read back open ports or tell you it doesn't have information yet. From this point, Alexa will give Kali the next instructions which will be a list of exploits to try that appear to match the open ports and service names found during the scan portion of the interaction.
13. You'll notice the Kali side will start chugging though exploits. If you start to see interesting things on the screen it's probably gotten a session.
14. By having the skill active and asking "**How's the hack going?**" Alexa will read the progress of the hack back to you including success or failure in getting hashes and list out the users if it was able to escalate privileges successfully.
15. THAT'S IT! Repeat at your leisure and explore the code because there is a wealth of knowledge you can ask Alexa about on things that are pretty relevant when you're web app hacking at the very least.