16 June 2020

|  |  |  |  |
| --- | --- | --- | --- |
| Date: | 16 June 2020 | Name: | Srinidhi J C |
| Course: | **Introduction to Cyber Security** | USN: | 4al16ec078 |
| Topic: | |  | | --- | | What is cyber security and what is the motivation behind it? | | Secure system design and security goals | | Threats | | Semester & Section: | 8th & b |
| Github Repository: | SrinidhiJC078 |  |  |

|  |
| --- |
| FORENOON SESSION DETAILS |
| Image of session  A close up of a card  Description automatically generatedA picture containing bird, flower  Description automatically generatedA screenshot of a cell phone  Description automatically generatedA screenshot of a cell phone  Description automatically generatedA screenshot of a cell phone  Description automatically generatedA screenshot of a cell phone  Description automatically generatedA screenshot of a cell phone  Description automatically generatedA screenshot of a cell phone  Description automatically generatedA screenshot of a computer screen  Description automatically generated |

Report:

Cyber-attacks are becoming more sophisticated all the time, with cyber hackers coming up with new and determined methods of threat that are increasingly difficult to detect, making attacks more dangerous than ever before.

Statistics show that cybercrime is on the rise around the world. It’s estimated that by 2021 the annual cost of damages from[cybercrime will cost the world $6 trillion](https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/). That’s a significant jump from $3 trillion in 2015, with cyber-attacks now one of the most serious threats to any business.

No matter the size of your organization, whether you’re a start-up business or have scaled to a million-dollar company, you need to be aware of the risk of a cyber-attack.

What’s the motivation behind all this cybercrime? The results of studies done on cyberhacking show that the motivation behind 90% of attacks is about financial gain and espionage. Here’s a closer look at the most-breached industries, who is doing the hacking, and what type of data is being hacked.

## **The most breached industries**

All businesses are at risk for a cyber-attack, but there are some industries that are more at risk than others for hacking. What makes these industries more vulnerable is the type of data that’s at risk of being stolen, including financial, health, and personal information.

## **Security design goals**

Identifying the basic aspects that compose the security of an e-money system.  
**PROTOCOL SECURITY**. By this we mean liveness and safety guarantees, namely, that the protocols achieve their goals and that every participant gets its information, and is secure in the sense that the other parties which are considered adversaries do not compromise or spoil the system. This aspect is the focus of this paper.  
**INTERNAL SECURITY**. The security of the internal operation system of the issuer of electronic currency, its capability to withstand insider attacks and abuses. The internal network architecture, operation policies, employment of tamper-proof hardware as well as dual control measures and access-control and physical access limitations should be reviewed. The internal security architecture must be combined with issues such as availability, reliability, load balancing and back-up requirements.  
**NETWORK SECURITY**. The security of the network (e.g., Internet) of users and the issuer, to prevent attacks not via the protocol but rather through ``break-ins;'' these attacks exploit the lack of proper protection into the system and software holes. Careful design of the interface to the external network (firewall protection) is required. Both the internal and the network systems must be evaluated under ``Global Security Testing,'' which includes penetration attempts and security assessment of design and implementation.  
**USER SECURITY**. Security of the user's assets. The user must obviously protect his electronic currency, and the software and procedures supplied to the user have to provide for protection at a proper level (e.g., beyond password-only protection), but at the same time be user-friendly.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date: | 16 June 2020 | Name: | Srinidhi J C | |
| Course: | IOT in Python with Raspberry Pi | USN: | 4al16ec078 | |
| Topic: | Programming IoT with IFTTT | Semester & Section: | 8th & b | |
| AFTERNOON SESSION DETAILS | | | |
| Image of session | | | |

Report:

# Introduction

IFTTT Platform! As a Platform user, all have access to an ecosystem of over 600 world-class services, thousands of active developers and millions of consumers. You’ll also be able to gain insights into how people connect and use your products so you can further personalize their experiences.

In order to build the best possible service on IFTTT, you should get acclimated with our IFTTT terminology. If you haven’t already, we provided a quick overview to help you get started below. If you’re ready to get started, you can [skip ahead](https://platform.ifttt.com/docs#timeline) and learn how to either create an example service or build your own service.

#### Triggers

Think of **triggers** as events on your service. Some example triggers are “Any new post” or “New photo added to album” or “Any new motion.”

#### Trigger Fields

**Trigger fields** allow for users to input filters or modifiers for a given trigger. For example, when a user selects the trigger “New photo added to album,” they will be asked for the album name.

#### Ingredients:

The attributes emitted from a given trigger are called **ingredients**. For the trigger “New photo added to album,” there would be ingredients for “PhotoURL” and “TakenAt.”

#### Queries

A **query** provides a way to request additional data that a trigger does not include. For example, when the Google Calendar service’s trigger “New event added” occurs you may need information about the event’s attendees. The “List Attendees” query returns that desired data.

#### Query Fields

**Query fields** are like form fields, and they are often populated with ingredients from a trigger. For example, the “List Attendees” query contains fields for “Calendar id” and “Event id”.

#### Actions

Conversely, think of **actions** as sending data or creating resources on your service. Some example actions include “Create post” or “Upload photo” or “Turn on lights.”

#### Action Fields

When creating an Applet, a user assembles these ingredients in the action’s **action fields**. Action fields are like form fields. For the action “Create post,” there would be an action field for “Title” and another for “Post body.”

### 1. Set up your environment

You will be creating IFTTT-specific endpoints that correspond to each trigger and action in your service.

If you already have an API that you would like to build on top of, there are a couple common strategies to add an IFTTT compatibility layer:

* Deploy a “shim app” that sits between your API and IFTTT. This app will translate resources from your API into responses IFTTT can understand.
* Add endpoints to your API service. These endpoints can source data from your existing API and serve them in responses IFTTT can understand.

If you do not have an API, IFTTT’s protocol will guide its development with its straightforward requirements.

In order to enable rapid iteration, you might consider using a tool like [ngrok](https://ngrok.com/usage) or [Forward](https://forwardhq.com/) to forward IFTTT’s request to a local development environment while you are in the early stages of service development.

### 2. Create your service and connect to IFTTT

IFTTT provides a testing tool which you will be using soon to build out your endpoints. The first endpoint you’ll build is /status, which will help you verify IFTTT can reach your service.

[Navigate to your service dashboard](https://platform.ifttt.com/services) and create a new service. You can leave many of the details for later, but be sure to fill out the **API URL Prefix**.

You can export the service in our [protocol](https://platform.ifttt.com/docs/api_reference) defined by the [OpenAPI 2.0 specification](https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md). To do this, navigate to the “Service” tab and go to the “Tools” section to export your service. With the OpenAPI definitions, you can use the [Swagger Codegen](https://swagger.io/swagger-codegen/) to bootstrap your service.

Build the [status endpoint](https://platform.ifttt.com/docs/api_reference#service-status). When it’s ready, navigate to the “Test” tab, and run the “Endpoint Test.” A passing status spec indicates that IFTTT was able to reach you!

### 3. Describe your queries, triggers, and actions

Fill out the details of your service, describing the [queries, triggers, and actions](https://platform.ifttt.com/docs/glossary#glossary).

While there’s a bit of art to choosing the right query, trigger, and action functionality for your service, the following principles are quite helpful:

**Start with a handful of the most obvious queries, triggers, and actions and expand later**. Triggers like “Any new photo” or “New post by you” tend to get broader use than ones like “Any new photo with tag” or “New post by you in category.” At least one query is required in every new service since almost any trigger or action can be translated into a relevant query.

**Less fields are better**. Instead of one trigger “Ignition turned on or off” with a field for “On or off?”, it’s much more powerful to have two triggers: “Ignition turned on” and “Ignition turned off.”

**Remember**, you can easily make changes before launching–consider this first pass a rough draft. Your main goal is to communicate to IFTTT what resources you will need to implement.

### 4. Use IFTTT’s endpoint testing tool

IFTTT’s **endpoint testing tool** will enable you to rapidly develop your endpoints with the peace of mind that they are compatible with IFTTT.

The workflow of the testing tool is straightforward:

1. IFTTT makes a request to {{api\_url\_prefix}}/ifttt/v1/test/setup
2. This endpoint’s response provides a valid access token for a user (if applicable) as well as valid **sample** values for each trigger and action field in your service.
3. IFTTT uses the access token (if applicable) and sample values to make test requests against your API.

You can find the endpoint testing tool under the “Test” tab on your service’s dashboard.

[See our documentation](https://platform.ifttt.com/docs/testing) for more details on setting up testing with your API.

Use this tool to build out the endpoints for your triggers and actions, referring to our [protocol reference](https://platform.ifttt.com/docs/api_reference) as needed.

### 5. Set up OAuth integration

IFTTT is fully compatible with a couple of the common flavors of OAuth 2.0. For more details on our implementation, see the [authentication section](https://platform.ifttt.com/docs/api_reference#service-authentication) of our protocol reference.

When you’re ready, use the **authentication testing tool** under the “Test” tab on your service’s dashboard to ensure the integration is working properly.

### 6. Implement the Realtime API

With IFTTT’s Realtime API, you can have Applets involving user-oriented triggers from your service run near-instantly. [See our documentation](https://platform.ifttt.com/docs/api_reference#realtime-api) for instructions on implementing Realtime.

### 7. Test your service end to end

It’s advisable that you create Applets using each trigger and action from your service on IFTTT to ensure that things work as expected. Your user will have special preview access to use the service on IFTTT.

### 8. Add service administrators

If we need to contact you about the service, we’ll use the information you provide us under the Service Administrators section.

### 9. Submit your service for review

Once you’re confident your service is going to delight users, submit it! You can find the submit link underneath the “Publish” tab on your service’s dashboard.

### 10. Create connections

Connections power integrations between your service and another service.

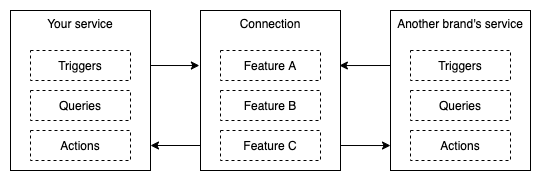
##### Prerequisites:

* A service on IFTTT with valid user authentication. Get started quickly with [Doorkeeper](https://github.com/kevinebaugh/doorkeeper-provider-app-ifttt#doorkeeper-provider-app--ifttt) (Ruby), [Authlib](https://github.com/authlib/example-oauth2-server) (Python), or any other OAuth 2 provider.

#### 1) Choose which services you would like to connect

You may already have a service in mind that you would like to connect to yours. In this example, we're going to build a connection that enables you to add events to Google Calendar on behalf of your users.

Your connection powers an embeddable integration between your service and another service:



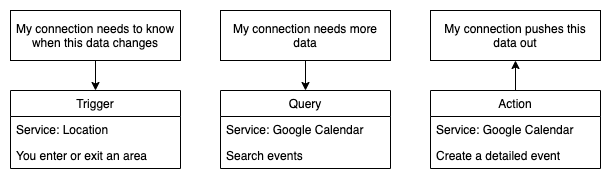
#### 2) Build your connection

[Create a new connection](https://platform.ifttt.com/mkt/connections/new) and search for and add the Google Calendar service, then click Continue.

Add a new [feature](https://platform.ifttt.com/docs/glossary#feature), then configure it to include the following:

* [Trigger](https://platform.ifttt.com/docs/glossary#trigger): Any event starts
* [Query](https://platform.ifttt.com/docs/glossary#query): List Events For a Date Range
* [Action](https://platform.ifttt.com/docs/glossary#action): Quick add event

While planning your connections, it's important to consider the relationship between triggers, queries, and actions. Here, an example illustrates a connection's feature enabling Google Calendar to "Effortlessly track time spent at work":



Learn more about using the connection creation tool [here](https://platform.ifttt.com/docs/connections).

#### 3) Enable your connection

Click on **Want to test or use this connection yourself? Enable it on IFTTT**, and enable the connection.

Head back to your connection's page on the platform, and view the **API logs**. You should see one event in the logs:

POST /ifttt\_developers/ifttt/v1/webhooks/connection/enabled

This likely resulted in a 404. That's to be expected unless you've set up the [connection enabled webhook](https://platform.ifttt.com/docs/connect_api#enabled-webhook). As you can see, that webhook will be fired anytime a user enables your connection.

#### 4) Test the trigger, query, and action

Take note of the **connection ID**, your **user ID**, and your **IFTTT service key** to use in the following requests. You can also find curl commands on your connection's page which you can copy/paste into a command line.

###### Test the Any event starts trigger

The [/test endpoint](https://platform.ifttt.com/docs/connect_api#test-trigger-event-webhook) simulates an event that would send a request to the [webhook endpoint](https://platform.ifttt.com/docs/connect_api#webhook-endpoint) you'll build later. In this example, sending a valid request to the /test endpoint would simulate an event starting for the user who has enabled your connection.

* HTTP REQUEST
* POST /v2/connections/{{YourConnectionID}}/triggers/google\_calendar.any\_event\_starts/test?user\_id={{YourUserID}}
* Host: connect.ifttt.com
* IFTTT-Service-Key: {{YourServiceKey}}
* {}
* EXPECTED RESPONSE
* HTTP/1.1 204 No Content
* Status: 204 No Content

###### Test the List Events For a Date Range query

This query will return a JSON object consisting of the Google Calendar events in the date range provided on the calendar you selected when you enabled the connection above.

* HTTP REQUEST
* POST /v2/connections/{{YourConnectionID}}/queries/google\_calendar.list\_events\_for\_date\_range/perform?user\_id={{YourUserID}}
* Host: connect.ifttt.com
* IFTTT-Service-Key: {{YourServiceKey}}
* {
* "fields": {
* "start\_date": "2020-01-01",
* "end\_date": "2020-01-05"
* }
* }
* EXPECTED RESPONSE
* {
* "type": "list",
* "data": [{
* "event\_id": "2fkfq7gs2q7779sh4jfn0oqed8",
* "title": "Test!",
* "description": "",
* "location": "",
* "event\_start": "2020-02-01T19:00:00.000Z",
* "event\_end": "2020-02-01T19:30:00.000Z",
* "event\_url": "https://www.google.com/calendar/event?eid=N3FzZnFuZGo3a3EwZjlzNDcyOGZnb2gyN2Uga2V2aW5AaWZ0dHQuY29t",
* "hangouts\_url": null,
* "created\_at": "2020-02-27T18:39:53.000Z",
* "attendees": {
* "type": "query",
* "query\_id": "google\_calendar.list\_attendees",
* "fields": {
* "calendar": "GAwRPru0NDARdGQFAHAQsbslltHmdmLOQAlmA5gGvy",
* "event": "2fkfq7gs2q7779sh4jfn0oqed8"
* }
* }
* }]
* }

###### Run the Quick add event action

This action will create a Google Calendar event on the calendar you selected when you enabled the connection above.

* HTTP REQUEST
* POST /v2/connections/{{YourConnectionID}}/actions/google\_calendar.quick\_add\_event/run?user\_id={{YourUserID}}
* Host: connect.ifttt.com
* IFTTT-Service-Key: {{YourServiceKey}}
* {}
* EXPECTED RESPONSE
* HTTP/1.1 204 No Content

Status: 204 No Content