Live Streaming of Unsecured Webcams for Ambience and Security Advisory

Markiyan Varhola, Andre Hirsch, Eino Taimen, Melike Ermis
Department of Information Systems
Hanyang University
Seoul, Republic of Korea

***Abstract*—There are many webcams available online that are unsecured, which can often be used by intruders to case the surrounding area or to investigate the private lives of the people in the webcam stream. However, many people are unaware of such streams, and some streams are set up deliberately to offer the users live information such as traffic views or weather info. By combining multiple webcam streams, a live stream that switches webcams every set number of seconds can be used as an ambient installation that allows users to watch a large number of webcams without the hassle of finding them.**

Keywords—webcams, unsecured, live, stream, ambient, effect, security, advisory

I. Overview

There are many webcams available online that are unsecured, which can often be used by intruders to case the surrounding area or to investigate the private lives of the people in the webcam stream. However, many people are unaware of such streams, and some streams are set up deliberately to offer the users live information such as traffic views or weather info. By combining multiple webcam streams, a live stream that switches webcams every set number of seconds can be used as an ambient installation that allows users to watch a large number of webcams without the hassle of finding them.

II. Resource Requirements

1. Streams of unsecured webcams

Websites like shodan.org and insecam.org have easily accessible webcam streams of unsecured webcams, which can be used for capturing images and video for the stream.

2. Web Server

An Amazon AWS EC2 instance can be used as a basic web and program server, and offers a free tier for minimal workload. It can be used to store the downloaded video and to run the live streaming software.

3. Live Streaming Software

Software such as FFMPEG can be used to stream live video from an input file via methods such as HLS and RTMP. This allows for low latency and is supported by multiple online streaming services.

4. Use of Youtube Live API or Facebook Live API

The web interface will greatly rely on the use of existing APIs that will offer functionality such as live commenting, the ability to rewind and video on demand.

III. Goals

1. *Create software to download streams of specified length from insecam.org*
2. *Set up RTMP-based streaming server on an AWS EC2 instance*
3. *Design a website for serving the content of the project*
4. *Create a Youtube Live stream to offload the bandwidth usage*

IV. Requirements

1. Development environment

The project will be build on a **Linux-based** system due to the availability of free and open-source software packages that are used in the project. The OS will be **Ubuntu 16.04 Server Edition**, hosted on an **Amazon AWS EC2** instance.

The specification of the EC2 instance will be based on the **t2.micro** tier, which is available for free under the AWS Free Tier pricing. The instance will have **1.0 GB of Memory**, **1 vCPU**, and **30GB of storage space.**

The software that will be used in the project will be free and open-source. For software that we write, we will use the **Ruby** programming language, (version **2.4.1**). Other software will include **nginx** as a web server, **ffmpeg** to stream content, and **Youtube Live** in order to offload the bandwidth of the streaming content.

The total cost of the project is expected to be zero in the short term (less than 1 year), as all of the software is free in cost. Due to the scale of the project, we will not require many resources to serve the content, and the free offerings are sufficient.

2. *Software in Use*

The software that inspired this project is by a developer named Derek Arnold. His software is designed as a post-internet art project in which he created a bot to take screenshots of random webcams found on insecam.org and processed them with imagemagick.

https://medium.com/@derekarnold/remote-viewing-5cb161cdef4a

The developer's other project would also take video samples from insecam, and process them into small, 1-minute long segments which would then be uploaded to a youtube channel called "Treasure Column". The videos are available at the following URL:

https://www.youtube.com/channel/UCKNW6jeGUfPUg_UsyAsTaPA

V. Task Distribution

- Markiyan Varhola
  - Software Developer
    - Develop and debug software
    - Implement each specification
- Andre Hirsch
  - User
    - Testing the software
    - Providing feedback to the developer
    - Introducing more possible features for better and easier handling
- Melike Ermis
  - Customer
    - Create requirements
    - Propose features
- Eino Taimen
  - Development Manager
    - Managing the development of the software
    - Ensure software is on time and under budget

VI. Specifications

- Stream Downloader
  - The stream downloader will perform the following steps:
    - Parse insecam.org and generate a link to a random stream.
    - Download the stream by watching the stream for a predetermined amount of time and then saving the output file.
- RTMP Server
  - The RTMP server will be created with nginx, which will serve both as a web and RTMP server. NGINX will be compiled with the nginx-rtmp module to allow for the streaming of content. The content server will be able to stream the output to a location such as rtmp://localhost/live, and the RTMP server will then allow outside viewers to connect and watch the stream.
- Content Server
  - The content will be downloaded locally onto the same machine that hosts the RTMP server and the Stream Downloader. A separate directory will be created for the downloading and storage of videos. The main constraint is the limit of hard drive space on the machine, which will be remedied by using a method for cycling videos or deleting ones that are too old.
- Website
  - The website will serve as a web interface for the project. It will host the stream and allow users to watch the live video.
  - The live video will be first sent to Youtube Live, and the link and player will be embedded on the website in order to save bandwidth and costs.

VII. Arcitecture Design & Implementation

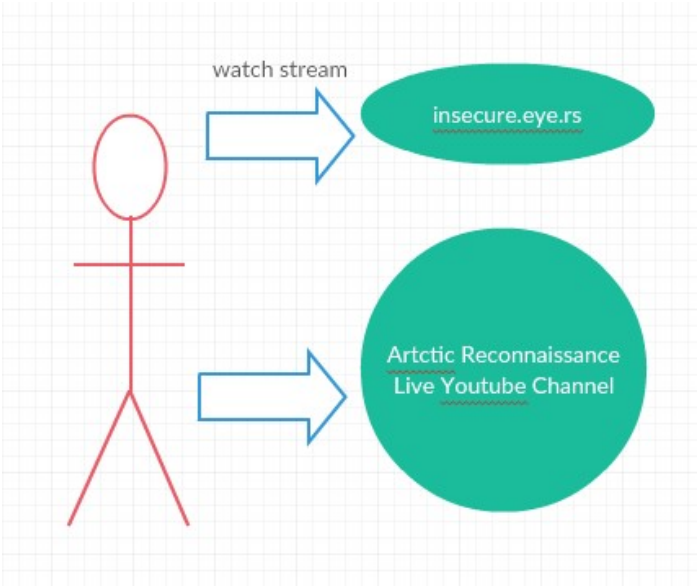
1. *Overall Architecture*

Module Names	Module Names
RTMP Server Stream Downloader Content Server	The stream downloader is run, downloads a video, then the video is sent to the content server, which processes the video, and then sends it to the RTMP Server to be distributed to the web.
Content Folder	This directory holds the downloaded videos, used by the Content Server

2. *Directory Structure*

Directory	File Names	Modules in Use	Structure Description
./server/	start_server.sh insecam_random.rb download.sh	RTMP Server Stream Downloader Content Server	Most server files are within this directory
./downloads/	{date}_{time}.flv (downloaded videos)	content	This directory holds the downloaded videos

VIII. Use Cases



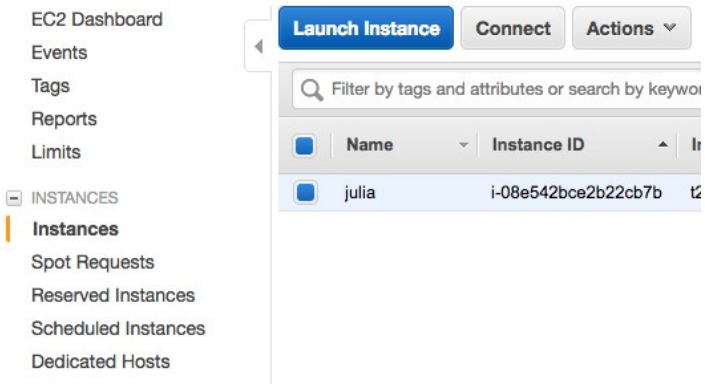
- Ambient Art Instillation
  - Provide an ambience for events via stylized live video stream.
    - Set up monitors / TV screens
    - Connect to computer
    - Open up the project's website
    - Open the video in full-screen
    - Done
- Security Advisory Visualization
  - A live stream that shows a variety of unsecured webcams that cycle periodically, can be used to discover unsecured webcams that should be secured.

IX. Installation Guide

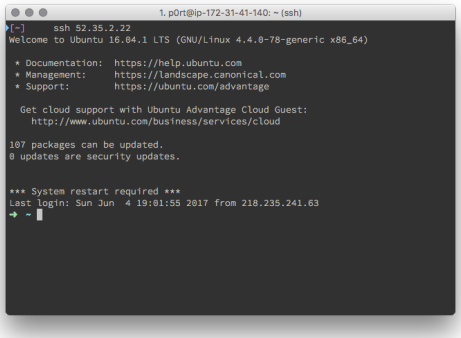
1. *AWS EC2 Instance*

To set up insecam-live, you must first create an account with Amazon AWS (https://aws.amazon.com/). Once you have completed this process, go into the AWS console via the web interface and select Amazon EC2 service.

On the left side of the EC2 console, select the instances tab, and click Launch Instance.



The process will guide you through creating an AWS instance. Select the desired instance type, and for the Operating System, select Ubuntu 16.04 Server Edition. Create a volume for storage, and a security policy to allow all inbound and outbound connections, and download a private key for logging into the server.
Once this is completed, log in to your server using your private key via SSH.



2. *nginx-rtmp module*

Make sure you have the necessary tools to build nginx using the following command:

```
$ sudo apt-get install build-essential libpcre3 libpcre3-dev libssl-dev
```

From your home directory, download the nginx source code:

```
$ wget http://nginx.org/download/nginx-1.13.1.tar.gz
```

Next, get the RTMP module source code from git:

```
$ wget https://github.com/arut/nginx-rtmp-module/archive/master.zip
```

Unpack/unzip them both, and enter the nginx directory:

```
$ tar -zxvf nginx-1.13.1.tar.gz
$ unzip master.zip
$ cd nginx-1.13.1
```

Now build nginx:

```
$ ./configure --with-http_ssl_module --add-module=../nginx-rtmp-module-master
$ make
$ sudo make install
```

And nginx is installed. By default it installs to /usr/local/nginx, to start the server run the following command:

```
$ sudo /usr/local/nginx/sbin/nginx
```

And to test to make sure nginx is running, point your browser to http://<your server ip>/ and you should get the "Welcome to nginx!" page.

3. *nginx-rtmp module configuration*

Open your config file, located by default at /usr/local/nginx/conf/nginx.conf and add the following at the very end of the file:

```
rtmp {
        server {
            listen 1935;
            chunk_size 4096;
            application live {
                live on;
                record off;
            }
        }
}
```

This is an extremely basic configuration with a "live" application that simply forwards the RTMP stream on to whoever requests it.

Restart nginx with:

```
$ sudo /usr/local/nginx/sbin/nginx -s stop
$ sudo /usr/local/nginx/sbin/nginx
```

Now nginx should be configured properly and running on your system. You can simply push your stream to rtmp://your-up/rtmp/live.

4. *insecam-live configuration*

Download insecam live from the following URL:
https://github.com/mvarhola/insecam-live/archive/master.zip

Unzip the project file and enter the directory via the following commands:

```
$ unzip master.zip
$ cd master/
```

Add execution permissions to each file by running the following command:

```
$ chmod +x *
```

5. *Running the server*

Run the scripts in the following order:

```
$ ./insecure_download.sh
$ ./start_server.sh
```

Then, you can view your stream via the following URL in a media player of your choice:

```
rtmp://<your_ip_address>/rtmp/live
```

The webcam stream should play the downloaded videos and remove the oldest video whenever there are more than 20 videos.