12-2017

Enhancing Cloud Security by a Series of Mobile Applications That Provide Timely and Process Level Intervention of Real-Time Attacks

Raqeeb Abdul
St. Cloud State University, raqeebabdul10@gmail.com

Enhancing Cloud Security by a Series of Mobile Applications That Provide Timely and Process Level Intervention of Real-Time Attacks

by

Raqeeb Abdul

A Starred Paper

Submitted to the Graduate Faculaty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in Information Assurance

December, 2017

Committee Members:
Dennis Guster, Chairperson
Lynn Collen
Kasi Balasubramanian

Abstract

Cyber threat indicators that can be instantly shared in real-time may often be the only mitigating factor between preventing and succumbing to a cyber-attack. Detecting threats in cloud computing environment can be even more of a challenge given the dynamic and complex nature of hosts as well as the services running. Information security professionals have long relied on automated tools such as intrusion detection/prevention systems, SIEM (security information and event management), and vulnerability scanners to report system, application and architectural weaknesses. Although these mechanisms are widely accepted and considered effective at helping organizations stay more secure, each can also have unique limitations that can hinder in this regard. Therefore, in addition to utilizing these resources, a more proactive approach must be incorporated to bring to light possible attack vectors and hidden places where hackers may infiltrate.

This paper shares an insightful example of such lessor known attack vectors by closely examining a host routing table cache, which unveiled a great deal of information that went unrecognized by an intrusion detection system. Furthermore, the author researched and developed a robust mobile app tool that has a multitude of functions which can provide the information security community with a low-cost countermeasure that can be used in a variety of infrastructures (e.g., cloud, host-based etc.). The designed mobile app also illustrates how system administrators and other IT leaders can be alerted of brute force attacks and other rogue processes by quickly identifying and blocking the attacking IP addresses. Furthermore, it is an Android based application that also uses logs created by the Fail2Ban intrusion prevention framework for Linux. Additionally, the paper will also familiarize readers with indirect detection techniques, ways to tune and protect the routing cache, the impact of low and slow hacking techniques, as well as the need for mobile app management in a cloud.

Table of Contents

Page

Chapter Page

List of Figures

Figure Page

Chapter I: Introduction

Problem Statement

Cloud computing supported by the UNIX operating system can be quite complex. This is borne out by the fact that information stored on the system is quite frequently stored in multiple places. In some cases, this process is automatically accomplished by the operating system when a related event occurs. Such is the case when an attempt is made to log into the secure shell service on a virtual machine (VM) in a cloud. Typically, the first hop of the route being used is recorded in the dynamic routing cache table. In most configurations of UNIX or variants such as LINUX (which is used in the study) this table addition occurs automatically on the server side and there is little an attacking client process can do to stop it (Benvenuti, 2006). Detailed information regarding the configuration, operation, and tuning of router cache is available from Bernat (2011a) and provides the flexibility to tune the cache to be more effective beyond the default settings in identifying attacking routes. Of course, the cache itself could be attacked and possibly disabled by a denial of service attack so mechanisms need to be in place to protect it Bernat, 2011b). Therefore, its internal settings need to be well thought out. For example, the garbage collector settings need to be optimized so the number of cached routes cannot grow too large (Nguyen, 2004).

Nature and Significance of the Problem

To illustrate this problem, a simplistic example will be delineated below using the author's cloud based VM test-bed. First, an abbreviated routing cache table with numeric addresses appears below. In the source column, the value is compared to the whitelist of Ipv4 addresses. In the first row, the value observed is in the whitelist category as having a valid IP within the cloud. However, the value in the second row is not in the whitelist. A lookup of this

IP address via the whois command indicates that it is leased through Digital Ocean. This low cost internet service has been used in the past by hackers to devise and test new techniques (Munsell, 2014) so it follows that further investigation is warranted. Please note, however, that the record includes a contact to report abuse so the ISP is taking some responsibility in the event their services are misused.

```
buster@bros:/rhome/classes$ route -Cn | more
Kernel IP routing cache
Source          Destination     Gateway         Flags Metric Ref    Use Iface
199.17.59.234   199.17.59.195   199.17.59.195       0      1    4776 eth0
188.226.139.158 199.17.59.234   199.17.59.234   l   0      0       3 lo

buster@bros:~$ cat whitelist_rt.local
IP for cloud        199.17.59.0         # Public Class C for all cloud zones
IP for Parent org   199.17.0.0          # Public Class Cs for parent org
IP for lo           127.0.0.0           # Loop back on VM
IP for internal     10.0.0.0            # Private Class A for internal nets
Ip for internal     192.0.0.0           # Private Class C for internal nets
buster@bros:~$ whois 188.226.139.158 | more
% Information related to '188.226.128.0 - 188.226.191.255'
% Abuse contact for '188.226.128.0 - 188.226.191.255' is 'abuse@digitalocean.com'

inetnum:        188.226.128.0 - 188.226.191.255
netname:        DIGITALOCEAN-AMS-4
descr:          Digital Ocean, Inc.
country:        NL
```

Figure 1. Abbreviated Routing Cache Table

The most prevalent type of attack on this VM is a brute force secure shell attack. Which, for the sake of simplicity, we can assume is detected by and logged by the Fail2Ban sub-process within the syslog facility. The event logic to log is simply three failed login

attempts and once this occurs the offending IP address is locked for 10 minutes. In the example below, an address starting with 222 met the intrusion status twice.

```
log for Fail2Ban v0.8.6
2016-01-03 09:05:05,677 Fail2Ban.actions: WARNING [ssh] Ban 222.186.21.73
2016-01-03 09:15:06,569 Fail2Ban.actions: WARNING [ssh] Unban 222.186.21.73
2016-01-03 09:16:59,760 Fail2Ban.actions: WARNING [ssh] Ban 222.186.21.73
2016-01-03 09:27:00,679 Fail2Ban.actions: WARNING [ssh] Unban 222.186.21.73
```

Figure 2. Failed Login Attempts Log

However, the 188 address described above does not appear in this log at all, meaning that based on this basic logic it is not defined as a security breach event. A quick analysis of the authentication log reveals that it tried to connect via the secure shell daemon, but did not get back a value from the service that could be capitalized upon and so it never returned an authentication response. This scenario was repeated three times within a couple of hour's space between the events. This slow and low volume attack scenario is consistent with sophisticated hacking techniques designed to minimize the attack footprint (Dev, 2014). Further, there were 32 instances of attacks from networks beginning with 188.226 so it may be wise to filter out all of that network traffic on a firewall level.

```
buster@bros:/var/log$ sudo cat Fail2Ban.log | grep 188
buster@bros:/var/log$

buster@bros:/var/log$ sudo cat auth.log | grep 188.226.139

Jan  5 10:49:21 bros sshd[5189]: Did not receive identification string from 188.226.139.158

Jan  5 12:50:13 bros sshd[6835]: Did not receive identification string from 188.226.139.158

Jan  5 15:35:49 bros sshd[8780]: Did not receive identification string from 188.226.139.158

buster@bros:/var/log$ sudo cat auth.log | grep 188.226. |wc –l      32
```

Figure 3. Slow and Low Volume Attack Scenario

So, based on the scenario described above, it is clear that evaluating the router cache can provide an additional tool to identify hacking attacks that may not be picked up by an intrusion detection system that might not be tuned to be overly sensitive in attempts to minimize false positives. As one would expect, developing a methodology that compliments existing intrusion detection strategies and provides quick alerts when a non-whitelisted site is detected could be a valuable addition to a cloud based security strategy.

Objective of the Study

Therefore, this paper will build on the basic attack methodology depicted above and implement a mobile application to remotely manage scripts that will evaluate the routing cache in relation to a whitelist in real time, send out alerts, log the offending events and provide basic performance information about the routing cache. This performance information such as table size, hit efficiency, and initial round trip time will be used in part to evaluate whether the cache itself has been potentially compromised. While this paper presents an interesting security problem, it also provides a series of pertinent hands-on scenarios that could be used in an educational environment. For example, the concept of a dynamic table look-up is certainly pertinent herein, but permeates throughout computing and computing security and once a basic understanding is attained that know ledge could be easily transferred to another scenario such as a dynamic ARP table.

Chapter II: Background and Review of Literature

Indirect Detection Techniques

One of the problems in devising an effective security strategy on a dynamic system such as a cloud is make sure that the detection system is quick and adaptable. In other words, attacks against a dynamic system are best detected by another dynamic system. This concept is supported by Jichkar and Chandak (2014) in their implementation of a security detection system that due to the dynamic topology of the networks any static configuration would not be sufficient. The work of Cho, Qu, and Wu (2012) built on this concept by placing the trust evaluation from their security schema on a series of dynamic systems they referred to as watchdog nodes. These nodes monitor and collect other sensors' behavior information and are tuned to dynamically spot problems within the trust interrelationships.

The work of Shuo, Jun, Kalbarczyk, and Iyer (2006) used a finite-state machine (FSM) approach to decompose programs into multiple elementary activities and used an indirect analysis of those activities to ascertain the vulnerability of the originating program. This is consistent with the prior references in this section in that a finite state machine can be used to model complex logic in dynamic systems. In this case, the FSM analysis pinpoints common characteristics among a broad range of security vulnerabilities: predictable memory layout, unprotected control data, and pointer taintedness. As one would expect, the solution lies in a more dynamic resource allocation approach which would randomize vulnerable areas such as memory layout.

The work of Saha, Lukyanenko, and Ylä-Jääski (2015), while not directly security related, is very pertinent to the core purpose of this paper. This work proposed and

implemented a new distributed architecture to efficiently use network-wide cache storage space based on a distributed caching algorithm. In the current paper the indirect detection system is based only on the router cache of a single host. In systems, which used the proposed method of Saha et al. (2015), the scope could easily be expanded form a single host to an entire cloud. The white list of known networks would just need to expand beyond those trusted by the host to those trusted by the cloud.

Tuning the Routing Cache

For quite some time, hackers have seen the value in attacking the routing structure within internets (Meyer & Partan, 2003). One simple ploy would be to launch some type of denial of service attack. Besides sound isolation of the routing tables, a good strategy to combat this scenario is to keep the cache well-tuned. This is in part measurable by looking at the percent of requests that are actually handled by the routing cache and do not require that packets be sent out to resolve the route, which is much slower. An abbreviated example from the author's cloud below reveals that the cache is tuned fairly efficiently because ~95% of the incoming requests are being cached (100 - (2053864 / 43539507) *100)). The outgoing requests reach a similar efficiency with a hit percentage of ~97%.

```
buster@bros:~$  lnstat -s1 -i1 -c-1 -f rt_cache
|rt_cache|rt_cache|          |rt_cache|rt_cache|
|  in_hit|in_slow_|          | out_hit|out_slow|
|43539507| 2053864|          |11315044|  325675|
```

Figure 4. Tuned Routing Cache

The work of Snader and Borisov (2008) provided an excellent example of what can happen if hacker attacks disrupt the expected routing performance. Specifically, they found

that when routing tunnels are allocated with limited resources this allows a malicious router operator to attack such tunnels. If the tuning metrics used are insensitive to relative load, the system does not adequately respond to changing conditions. This situation results in unreliable performance which may drive many users away.

While Bernat (2011a) stated that when Linux is used as a router, the inefficiency of the route cache can hinder the performances of your host he also states that information on how to tune that cache is scarce and it is difficult to find up-to-date information on how the route cache works. Of course, this makes tuning it problematic. As often is the case with mundane computing topics, O'Reilly Publishing comes to the rescue with "Understanding Linux Network Internals" (Benventuti, 2006). Specifically, Chapter 33 deals with these issues. However, Bernat (2011a) provided a decent overview in regard to the basic of tuning router cache to help ensure secure operations. Specifically, decisions need to be made in regard to setting the hash table size, target average length of the queue, delay between garbage collection runs and when to remove an outdated entry. Of special note from a security perspective is the importance of tuning the garbage collection process which helps makes sure that unused resources are recovered in real time which lessens the impact of a denial of service attack.

Protecting the Routing Cache

Authors Zhang, Mao, and Wang (2007) discussed the need to protect against routing misbehaviors by using route normalization to protect local network traffic from erroneous and malicious routing traffic. Backbone network elements such as intrusion detection systems, firewalls, and routers all depended on the integrity and cleanliness of mechanisms like the

routing cache. If it goes awry, then the whole network can be compromised, therefore, it's paramount to protect routing caches due to its inherent trustworthiness. Zhang et al. (2007) also explain how using a RouteNormalizer that mitigates violations with routing loops, missing mandatory attributes, nexthop violations, and export policy violations can mitigate and detect routing anomalies too.

Impact of Low and Slow Hacking Techniques

Even the most robust intrusion detection systems can be deceived by the low and slow hacking techniques that stay just enough under the radar yet often do the most damage. This is one of the main reasons why tuning and protecting the routing cache is vital to seeking out hidden places malicious hackers can lurk. An example of such surreptitious tactics is found in advanced persistent threats (APT's). APT's take "a low-and-slow approach" using social engineering "to gain access to a network and steal information quietly" (Teller, 2012). Symantec (n.d.) described APT's as, "An advanced persistent threat uses multiple phases to break into a network, avoid detection, and harvest valuable information over the long term. This information below details the attack phases, methods, and motivations that differentiate APTs from other targeted attacks" (¶ 1). Symantec further explained how APT's are carried out using the following methodology:

1. *Incursion*. Attackers break into network by using social engineering to deliver targeted malware to vulnerable systems and people.
2. *Discovery*. Once in, the attackers stay very much under the radar to avoid detection. They then map the organization's defenses from the inside and create a battle plan and deploy multiple parallel kill chains to ensure success.

3. *Capture*. Attackers then access unprotected systems and capture information over an extended period. They may also install malware to secretly acquire data or disrupt operations.
4. *Exfiltration*. Captured information is sent back to attack team's home base for analysis and further exploitation fraud and/or worse.

Larry Clinton, President of Internet Security Alliance (2013), described APT's as highly skilled, day job hackers who are going to get into systems and they are very persistent. Additionally, Clinton (2013) described how they "stay for dinner and breakfast and don't ever go away. They're going to go quiet and companies will think they're gone but they are not. They're not interested necessarily in taking down a system but rather using and gathering information about company systems and will "call home" with new information to advance their threat. They are very skilled at hiding under the radar of anti-virus software. They are usually going after a company's Internet Protocol (TCP/IP), interested in identity thefts, wire fraud and harvesting other types of classified information (Clinton, 2013)

Need for Mobile App Management in a Cloud

Given that attacks against routing resources can be disastrous and must be dealt with in a timely manner, the interface used to alert the system administrator to such attacks must be agile (Cho et al., 2012). Therefore, the goal of any remote system administration management software should be to facilitate ease of use, speed and security. Because equipment rooms are seldom staffed anymore, the logical solution is to provide management capabilities from a mobile device. In the paper herein, the examples are based on Linux platforms and it therefore is logical to evaluate the available mobile apps that are compatible with Linux systems. There

are several mobile apps available, but typically they depend on ssh or some type of remote virtual terminal program to gain access to the operating system. This situation would require that the appropriate command be entered, which may not be all that fast given the keyboard characteristics of some mobile devices (Geier, 2015). However, as previously stated a mobile device will be critical in an obtaining an effective solution because it is readily available and easily accessible.

The primary goals in adapting a mobile device to support the remote identification (via the router cache) and mitigation of brute force attacks are typically related to identifying and quickly blocking the attacking IP addresses, the ease of use, and, of course, added security. In the case of not meeting these goals, then the solution would provide limited value. A similar security APP was devised by Guster, Abdul, and Rice (2015) except it was designed to identify and mediate rogue processes.

In that case, the challenge was to protect the system while a true decision about the offending rogue process was being made by the system admin. Of course, this decision will take time certainly the mobile APP approach would minimize some of the end-to-end delay associated with logging in remotely via a virtual terminal program or a browser. In such cases there will be some degree of asynchronous human think time. In Guster et al. (2015), a strategy was pursued to just suspend the rogue process, and then if required, killed by the sys admin via the mobile app. A similar approach will be applied herein where the block is temporary and can be made permanent if desired by the system administrator.

Chapter III: Methodology

Because quickness of analysis is critical, the Android mobile application strategy facilitates the availability of information pertaining to basic router cache performance information, incursions recorded in logs, and the efficiency of the routing cache configuration. To evaluate the implementation in an effective manner, the design will be presented in the following modules:

1. Efficiency of the router cache
2. Evaluating Routing cache from a mobile application
3. Authentication logs
4. Fail2Ban logs
    a. Push notification for Fail2Ban Logs
    b. Inserting records to database
5. Mitigating VM Denial of Service from a mobile application.
    a. Identifying Sub-Targets
    b. Tying the Sub-Targets Together with a PID
    c. Identifying a Strategy for Killing PIDs from a Mobile Device
    d. Proactively Identify a Rogue Process
    e. Server side
    f. Client Side
6. Authentication and Encryption for accessing mobile application

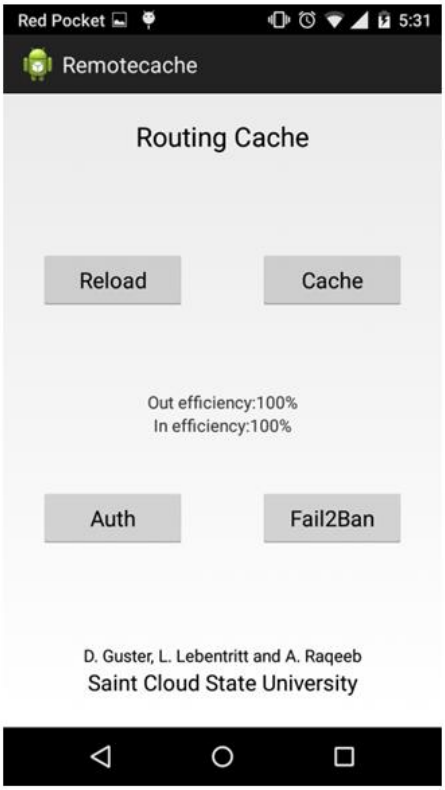

Figure 5. Mobile App – Routing Cache

All the modules above are discussed from two different perspectives: the client side (Mobile application) and Server side (Linux Host). The server side is implemented using shell scripts and PHP scripts.

Efficiency of the Router Cache

The router cache efficiency can be determined for outgoing and incoming connections. So the efficiency of the router cache is calculated on the server side. The PHP script on the server side fetches the cache from the following location "/proc/net/stat/rt_cache." The output is not structured. So, to parse this data, we have identified common delimiters like a comma, dot, pipe, etc. This PHP script also exposes a REST API so as to consume it on the mobile application. The REST API returns the data in JSON format.

URL: http://<server>/performance.php

Method: GET

URL Params: <none>

Header: Authorization: Basic + Base64(encrypt(username):encrypt(password))

Response: {"in_efficency":"100","out_efficency":"100"}

Figure 6. PHP Script for Efficiency of Router Cache

Below, the strategy for calculating the in and out router cache efficiency is implemented in the REST API.

Incoming efficiency:
in_efficiency = 100 – (rt_cache_in_slow_tot/rt_cache_in_hit)*100
Outgoing efficiency:
Out_efficiency = 100 – (rt_cache_out_slow_tot/rt_cache_out_hit)*100

Evaluating Routing Cache from a Mobile Application

Figure 7. Mobile App – Evaluating the Routing Cache

The routing cache can be viewed by commands “route -Cn” or “netstat -Cnre”. The whitelist_rt.local file has all the networks that are trusted on the cloud level which can

basically be considered as Whitelisted IP addresses. Those networks not listed may or may not become a blacklisted address, but this APP would alert the system administrator to evaluate their presence (typically via a script).The initial evaluation of routing cache is done by comparing source ip address from output of “route -Cn” with the IPs that are listed in whitelist_rt.local. Since, the data is not structured we use the same strategy as above to parse the data. The PHP script exposes a REST API which returns the data in JSON format, which allows us to work with objects.

URL: http://<server>/cache.php
Method: GET
URL Params: <none>
Header: Authorization: Basic + Base64(encrypt(username):encrypt(password))
Response: [{"src":"199.17.59.245","dst":"199.17.59.195","gateway":"199.17.59.195","flags":"-","metric":"0","ref":"1","use":"1058","iface":"eth0","blacklisted":0},{"src":"110.77.138.151","dst":"199.17.59.245","gateway":"199.17.59.245","flags":"l","metric":"0","ref":"0","use":"2","iface":"lo","blacklisted":1}]

The script below provides the logic and implementation strategy for evaluating the routing cache:

```
function parse_ip($ip)
{
        $ret = 1;
        $data = shell_exec('cat whitelist_rt.local');
        $data1 = explode("\n",$data);
        foreach ($data1 as $value){
        if(strlen($value)>1)
        $ips[] =preg_replace("/\.{2,}/",".",str_replace(".0",".",preg_replace("/[^0-9.]/","",$value)));
        }
        foreach ($ips as $addr) {
           if (strpos($ip, $addr) === 0) {
              $ret =0;
           }
        }
        return $ret;
}
```

Figure 8: PHP Script for Evaluating the Routing Cache

The above parse_ip function takes only a single argument such as the IP address and then compares it with the IPs stored in ‘whitelist_rt.local’. The comparison is done by checking the net portion of whitelisted IP address against the source IP address in the router cache. In PHP the strops (string operations) returns the first occurrence of the substring. If the IP address is not a substring then it is listed as blacklisted for this phase. It returns a ‘1’ if the IP is blacklisted and ‘0’ if not.

Authentication Logs

The authentication logs are useful for evaluating user login patterns and determining when sudo commands are invoked. This “auth log” file can be accessed at /var/log/auth.log. In addition this log is useful for identifying malicious activities. Besides the convenience displaying information on a mobile application is better approach because the auth.log data is not structured. Of course, it is easy to impose structure within the mobile app display. The strategy for parsing the data is similar to the examples above which use delimiters like space, periods, pipes and so forth. The server also uses a REST API which returns a JSON array.

URL: http://<server>/auth_log.php
Method: GET
URL Params: <none>
Header: Authorization: Basic + Base64(encrypt(username):encrypt(password))
Response: [{"timestamp":"Feb 1 11:01:02","log":"php1 CRON[5453] pam_unix(cron session) session closed for user smmsp"},{"timestamp":"Feb 1 11:09:01","log":"php1 CRON[5473] pam_unix(cron session) session opened for user root by (uid=0)"}]

The response is structured into basically timestamp and log.

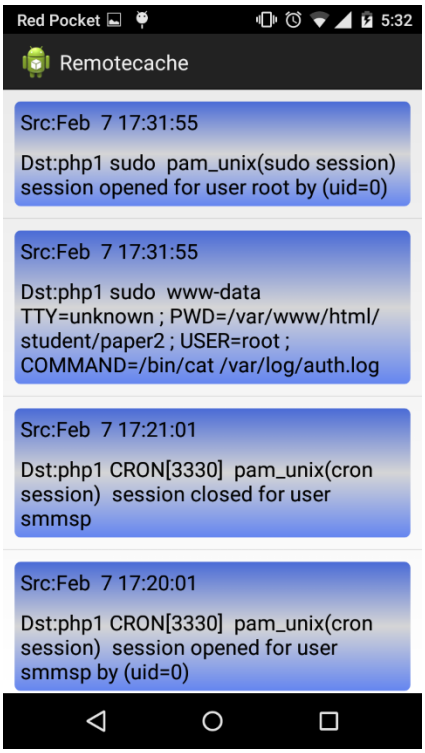

Figure 9. Imposing Structure for Authentication Logs on the Mobile App

Fail2Ban Logs

The Fail2Ban logs are created by Fail2Ban intrusion prevention framework. These logs can be found on /var/logs/Fail2Ban.log. Also, these logs are unstructured. So, to make this log meaningful and secure, we have created a script which copies of records to a MySQL database. To do this in Fail2Ban framework, we need to create an action that will trigger a script which writes data to MySQL whenever a ban or unbanned event has occurred. This configuration file is located at /etc/Fail2Ban/action.d. For the project herein, the configuration file below was created.

```
student@php1:/etc/Fail2Ban/action.d$ cat qshield.conf
[Definition]
actionstart =
actionstop =
actioncheck =
actionban = iptables -I INPUT -j DROP -s <ip>
        sudo php /var/www/html/student/paper2/ban.php <ip>
actionunban = iptables -D INPUT -j DROP -s <ip>
        sudo php /var/www/html/student/paper2/unban.php <ip>
```

Figure 10. Configuration File for Fail2Ban

The above configuration will call ban.php and unban.php whenever ban and unban of IPs occur respectively. Both the scripts evaluate one argument: ip-address.

Lastly, we transfer the information in "qshield.conf" into "jail.local" which is the configuration file for the Fail2Ban framework. This configuration file has jails for various protocols like http,ftp,ssh etc. Since this paper is concerned about ssh, we have created a jail only to ban ssh. Below is a sample the configuration file.

```
[ssh]

enabled  = true
port     = ssh
filter   = sshd
logpath  = /var/log/auth.log
maxretry = 3
action = qshield
```

These logs are integrated into the mobile application by using a REST API. In this case, the log files didn't need to be parsed because they are already stored on the database. So, a simple "get" query could be used to retrieve the data.

```
URL: http://<server>/Fail2Ban.php
Method: GET
URL Params: <none>
Header: Authorization: Basic + Base64(encrypt(username):encrypt(password))
Response:  [{"id":"16","status":"ban","ip":"199.17.59.234","timestamp":"2016-02-06
14:05:53","OrgName":"Minnesota State Colleges and Universities","Address":"30 7th Street East, Suite
350","City":"St.
Paul","StateProv":"MN","PostalCode":"55101","Country":"US"},{"id":"17","status":"ban","ip":"199.17.59.234","ti
```

mestamp":"2016-02-06 14:15:53","OrgName":"Minnesota State Colleges and Universities","Address":"30 7th Street East, Suite 350","City":"St. Paul","StateProv":"MN","PostalCode":"55101","Country":"US"}]

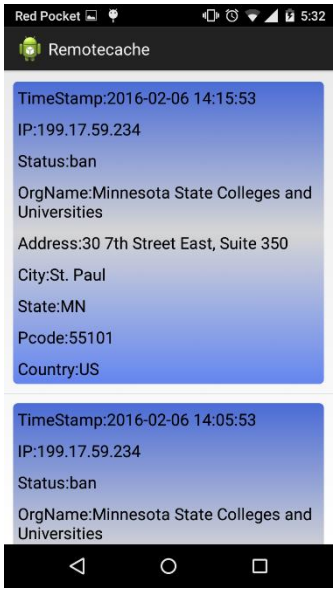

Figure 11. Configuration File Integrated into the Mobile App

Push notification for Fail2Ban Logs. The push notification is a message which is delivered by the server to the mobile application automatically without any need for a request from the mobile application. In the implementation herein, the push notification is used to notify the user about the bans and unbans of the IP address. This is also useful in identifying false positives.

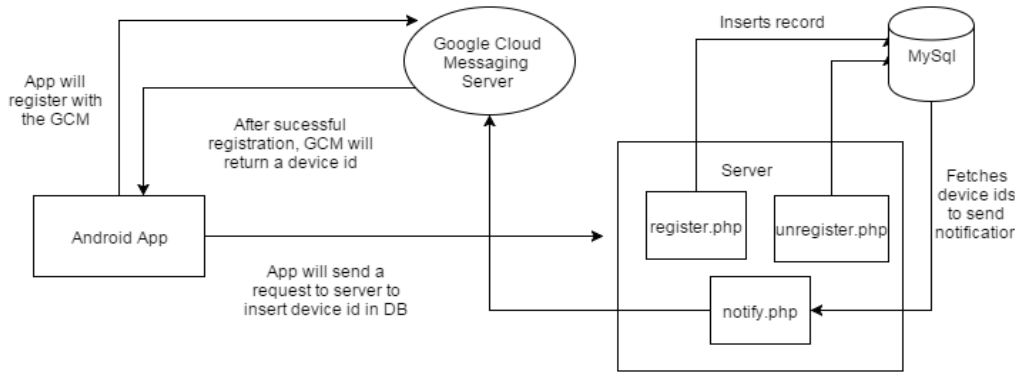

Figure 12. Push Notification Architecture

The mobile device will register with the Google Cloud Messaging server (GCM) upon successful login to the application. If the registration is successful, the GCM will return a device id which is a unique id to identify the device and application within the cloud. To send a notification to a mobile application, this device ID is imperative. To facilitate future development and in the spirit of event logging this device ID is being saved in a MySQL database. This process is accomplished when the registration API is called. To send notifications, a script called notify.php was created which in turn has a function called "notify" containing the arguments message and the IP-address that are captured from Fail2Ban.The ban.php and unban.php files which are discussed above call this "notify" function along with the arguments message and IP-address. Below is the format of a request to the GCM server.

URL:https://android.googleapis.com/gcm/send
Method: POST
URL Params: <none>
Header: Authorization: key=YOUR_API_KEY, Content-Type: application/json
Parameters: registration_ids = <array of device ids>, data = <array of key value pair>
Response:
{"multicast_id":6123852703457412158,"success":3,"failure":0,"canonical_ids":2,"results":[{"message_id":"0:1454882086684937%39ad5476f9fd7ecd"},{"registration_id":"APA91bGD3SNy6XfBr0o8x9d6Eh_TPSKRVjDfO8IHmmNqIfkXZjxaQKZqrr-DQywAo7rJypbNcXhnEznAjsluVBuw78Ow6QgxPYVesIFg79ZHt55_0FNmXBuXhq3bvo26TArTbKho2ras","message_id":"0:1454882086684941%39ad5476f9fd7ecd"}]}

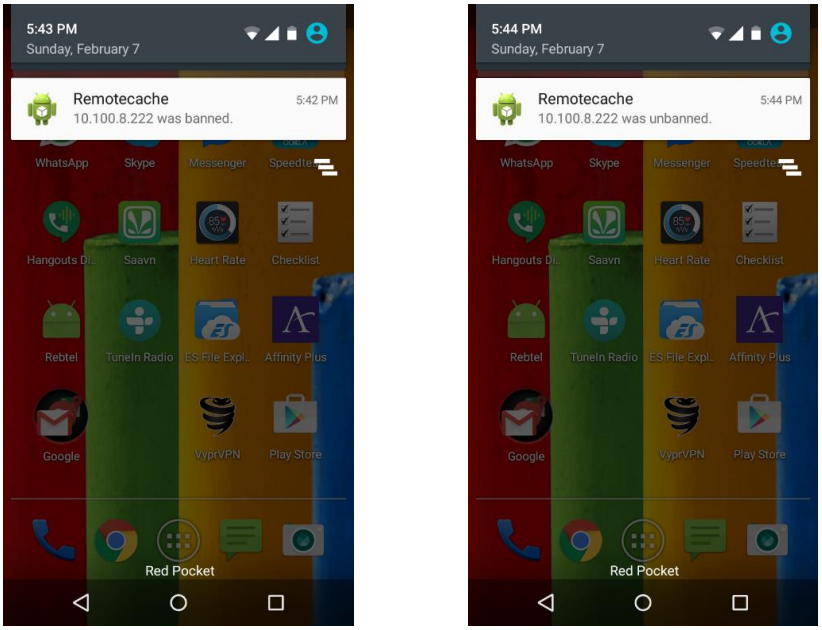

Figure 13. Push Notifications for Fail2Ban Logs

Inserting records to database. The idea behind this strategy is to secure the logs. It is similar to taking backups of Fail2Ban logs. This strategy will be more efficient if the database is on some other remote server. But, in our design, for the sake of convenience, we have used same host as where Fail2Ban is installed. This database can be used to create the REST API that can be consumed on the mobile application. To make this data more meaningful, we have included the location, host organization, etc. of the IP-address. To achieve this we have used "dig" command. Below is the function that takes the one argument IP-address and returns the array of details.

```
function getLoc($ip){
        $data = shell_exec("whois ".$ip." | grep
'OrgNam\|City\|Address\|StateProv\|PostalCode\|Country'");
        $data1 =  explode("\n",$data);
        foreach($data1 as $value){
        $data2 = explode(":",$value);
        if(sizeof($data2)>=2)
        $output[$data2[0]] = trim($data2[1]);
}

return $output;
}
```

Figure 14. Inserting Records To Database

Mitigating VM Denial of Service from a Mobile Application

Identifying sub-targets. Denial of Service attacks (DoS) are relatively easy to launch and if the target is not prepared, can be very effective. Therefore, it is critical that a potential target site devises a well thought out and timely strategy to combat such attacks. DoS have been around some time and have become easier to launch with the advent of the Internet (Hafner & Lyon, 1996). The Internet, in effect, gave the hacking world the access needed to launch DoS attacks remotely with no need to gain physical access to the computer room of the target. There are certainly many classi cations of attacks that could originate from the network and foundations of which are described by Cheswick and Bellovin (1994). However, all attacks will become instantiated in the operating system under one or more process identi cation numbers (PID). Properly identifying such attacks requires a sound monitoring strategy (Kargl, Mair, Schlott, & Weber, 2001). Monitoring the PID provides a common element that

can tie together the different system resources that might be affected by a DoS attack. Commonly, these resources fall into four categories: CPU, memory, storage, and network.

Tying the sub-targets together with a PID. To illustrate this interrelationship, a temperature conversion service (which converts Fahrenheit to Centigrade) has been instantiated on a VM within a cloud. The Linux netstat (network statistics) command output given reveals that it is running on network port 18002 and available from all networks (:::) assigned to that host and has been given PID 3224.

```
dguster@eros:~$ netstat -apeen | grep java
tcp6       0      0 :::18002                :::*                    LISTEN      1004536945 26213822    3224/java
```

The next command, ps (display processes) as seen below provides us with the memory (MEM) and CPU usage, in both cases they are well below 1% which indicates that this process in its present state is not causing either a memory or CPU DoS.

```
dguster@eros:~$ ps -aux | grep 3224
USER       PID %CPU %MEM    VSZ   RSS  TTY      STAT START   TIME COMMAND
dguster   3224  0.3  0.4 2245236 17656 pts/3   Sl+  14:03   0:02 java TempServer
```

However, in a cloud architecture using virtualization and symmetric multiprocessing, the CPU resources may be distributed across many physical devices. The second ps –ALF command below shows us that the original PID 3224 has been broken down into 14 “light weight processes” and are assigned to four different processors (PSR) numbered 0-3. While the original process 3224 retains the same PID number for the light weight process ID, each subsequent light weight process receives a new and different LWP number. However, because these are assigned hierarchically. if one were to kill the root level process, in this case 3224, the whole process stack would be removed. This fact will become important later in the paper where mitigating a DoS attack as quickly as possible becomes imperative. A quick side note

about this type of architecture and DoS attack is that the ability to multi-thread tasks and bring multiple processors into the fray when a DoS attack occurs makes it much more difficult for a DoS attack to be successful. Further, it lengthens the time a system administrator has to kill the offending process(s) and mitigate the attack.

```
dguster@eros:~$ ps -ALF | grep 3224
UID        PID  PPID  LWP  C NLWP    SZ   RSS PSR STIME TTY          TIME CMD
dguster   3224  3120  3224  0   14 561309 17672  3 14:03 pts/3    00:00:00 java TempServer
dguster   3224  3120  3225  0   14 561309 17672  0 14:03 pts/3    00:00:00 java TempServer
dguster   3224  3120  3228  0   14 561309 17672  1 14:03 pts/3    00:00:00 java TempServer
dguster   3224  3120  3229  0   14 561309 17672  2 14:03 pts/3    00:00:00 java TempServer
dguster   3224  3120  3230  0   14 561309 17672  3 14:03 pts/3    00:00:00 java TempServer
dguster   3224  3120  3231  0   14 561309 17672  0 14:03 pts/3    00:00:00 java TempServer
dguster   3224  3120  3232  0   14 561309 17672  0 14:03 pts/3    00:00:00 java TempServer
dguster   3224  3120  3233  0   14 561309 17672  2 14:03 pts/3    00:00:00 java TempServer
dguster   3224  3120  3234  0   14 561309 17672  3 14:03 pts/3    00:00:00 java TempServer
dguster   3224  3120  3235  0   14 561309 17672  1 14:03 pts/3    00:00:00 java TempServer
dguster   3224  3120  3236  0   14 561309 17672  1 14:03 pts/3    00:00:00 java TempServer
dguster   3224  3120  3237  0   14 561309 17672  2 14:03 pts/3    00:00:00 java TempServer
dguster   3224  3120  3238  0   14 561309 17672  1 14:03 pts/3    00:00:00 java TempServer
dguster   3224  3120  3239  0   14 561309 17672  1 14:03 pts/3    00:00:02 java TempServer
```

Last, the lsof command (list open files) provides a way of linking the process to storage resources. Below we are able to see respectively: the path to a directory on secondary storage, a library loaded from secondary storage into memory, a temporary file system workspace in memory but writable as a file to secondary storage, a UNIX socket which allows the java class to be linked to the operating system and finally a IP version 6 network connection linked to port 18002. A bottom up approach would require that each of these elements linked to PID 3224 be deleted independently which would more than likely not be quick enough to mitigate a modern DoS attack. Therefore, an effective method would be to kill the root process, in this case PID 3224, and all sub-processes (sometimes referred to as children) would then be eliminated as well.

```
dguster@eros:~$ lsof -p 3224
COMMAND  PID    USER   FD   TYPE             DEVICE SIZE/OFF     NODE NAME
java    3224 dguster  cwd    DIR              0,21    24576  786911 /rhome/dguster/javaclass
```

```
java    3224 dguster    2u   CHR             136,3      0t0       6 /dev/pts/3
java    3224 dguster  mem    REG               8,1   149280  715938 /lib/x86_64-linux-gnu/ld 2.15.so
java    3224 dguster  mem    REG               8,1    32768  525582 /tmp/hsperfdata_dguster/3224
java    3224 dguster   12u  unix 0x0000000000000000      0t0 26213820 socket
java    3224 dguster   13u  IPv6          26213822      0t0       TCP *:18002 (LISTEN)
```

Killing the Primary and Secondary PIDs with a Kill Command

Below we see that the process 3224 is killed. Then, a search using the ps –aux "display process command" reveals only the search argument using the grep filter. Further, a list of open files query returns nothing. So therefore 3224 and its children are no longer instantiated on the VM.

```
dguster@eros:~$ kill 3224
dguster@eros:~$ ps -aux | grep 3224
dguster   6051  0.0  0.0   6500   624 pts/0    S+   15:22   0:00 grep --color=auto 3224
dguster@eros:~$ lsof -p 3224
```

Identifying a strategy for killing PIDs from a mobile device. The primary goals in adapting a mobile device to support the remote mitigation of DoS attacks are typically related to identifying and quickly eliminating rogue PIDs, the ease of use, and, of course, added security. If any one of these goals are not met, then the solution would provide limited value. In terms of identifying rogue processes there are certain parameters that can be monitored and a simple example appears below. A java class is writing a large file into memory and is taking up 98.5% of the available memory available for this type of application. Therefore, if this mobile management strategy is to be effective this obvious violation has to be identified and the associated process (PID 12023) killed in a timely manner. Writing a script or scripts to identify such a violation and sending an alert to the mobile device of a system admin would be a major step in the success of this type of project.

```
dguster@eros:~$ ps -aux | grep java

dguster  12023 98.5 21.0 3271356 850956 pts/0  Sl+  09:52   0:22 java MemoryMappedFileInJava2
```

In terms of killing the process, once it is identified the problem will then be greatly affected by end-to-end transmission delay. Once the command is executed and transmitted to the host the delay is relatively minimal, as can be seen in the output below. The task itself takes ~ 3.65 milliseconds to run and the elapsed time from command to response is ~ 7.11 milliseconds.

```
dguster@eros:~$ perf stat -B kill -9 12161

 Performance counter stats for 'kill -9 12161':

       3.651294 task-clock                #    0.514 CPUs utilized
              1 context-switches          #    0.000 M/sec
              1 CPU-migrations            #    0.000 M/sec
            183 page-faults               #    0.050 M/sec
   <not counted> cycles
              0 stalled-cycles-frontend   #    0.00% frontend cycles idle
              0 stalled-cycles-backend    #    0.00% backend  cycles idle
              0 instructions              #    0.00  insns per cycle
              0 branches                  #    0.000 M/sec                  [19.34%]
   <not counted> branch-misses

     0.007110059 seconds time elapsed

dguster@eros:~$ ps -aux | grep java
dguster  12184  0.0  0.0   6500   624 pts/3    S+   10:02   0:00 grep --color=auto java
```

The challenge then will be to protect the system while a true decision about the offending rogue process is being made by the system admin. Once again, this decision will take time, although the mobile app approach should minimize some of the end-to-end delay associated with logging in remotely via a virtual terminal program or a browser there still will be some degree of asynchronous human think time. Therefore, a strategy needs to be pursued to have the potentially rogue process suspended by the identifying script and if determined to be a rogue process, then killed by the sys admin via the mobile app. In cases in which the process is determined not to be a rogue process then it could be restarted from the same APP. While suspending a process stops it from using CPU resources, the memory content remains

in place but cannot grow. The LINUX command series in Figure 8 shows how a given process can be temporarily disabled and restarted using the kill command. A PID (14439) from a potentially rogue process is identified and disabled with the signal stop switch via the kill command. When the process is then displayed its status flag changes from S (suspended waiting for resources) to T (stopped). In the last part of the results below it is then changed back to S (if indeed it is not a rogue process), but of course it could also be killed if it truly is a rogue process. This type of logic will need to be incorporated into the design of the mobile app to minimize the effect of false positives.

```
dguster@eros:~$ ps -aux | grep java
dguster  14439 38.6 25.7 3271356 1043744 pts/0 Sl+  12:45   0:11 java MemoryMappedFileInJava2
dguster  14455  0.0  0.0   6500   624 pts/3    S+   12:46   0:00 grep --color=auto java

dguster@eros:~$ kill -SIGTSTP 14439

dguster@eros:~$ ps -aux | grep java
dguster  14439  8.8 25.7 3271356 1043744 pts/0 Tl   12:45   0:12 java MemoryMappedFileInJava2
dguster  14463  0.0  0.0   6500   624 pts/3    S+   12:47   0:00 grep --color=auto java

dguster@eros:~$ kill -SIGCONT 14439

dguster@eros:~$ ps -aux | grep java
dguster  14439  7.3 25.7 3271356 1043744 pts/0 Sl   12:45   0:12 java MemoryMappedFileInJava2
dguster  14490  0.0  0.0   6500   620 pts/3    S+   12:48   0:00 grep --color=auto java
```

Proactively identifying a rogue process. In order to effectively manage DoS processes it is important to devise a proactive rather than a reactive policy. As state earlier there are four main categories of denial of service: CPU, memory, disk and the network. For the sake of simplicity, we will focus on the CPU. There are many ways of using the Linux tools to identify a process that is overusing CPU resources, but in most cases they are a variant of the ps command. To illustrate this process, we picked the method from How to Find Which Process is Causing High CPU Usage (Newpaint, 2015), and the results appear below:

```
dguster@eros:~/javaclass$ ps -eo pcpu,pid,user,args | sort -k1 -r | head -10

%CPU   PID USER     COMMAND
82.8  3748 dguster  java MemoryMappedFileInJava2
12.0  3764 root     [flush-0:21]
 0.1  1395 root     /usr/sbin/vmtoolsd
 0.0 31281 root     [kworker/0:0]
 0.0 30824 root     /usr/lib/policykit-1/polkitd --no-debug
 0.0 30757 root     /usr/sbin/console-kit-daemon --no-daemon
 0.0 30697 syslog   rsyslogd -c5
 0.0 30344 root     /usr/sbin/sshd -D
 0.0 29030 root     [kworker/3:2]
```

The resulting list displays the 10 processes that are using the most memory. The first entry in the list is really problematic in that it is using 82.8% of available memory. Note that the Linux operating system has processes in place to help safeguard against a denial of service. For example, in the list above, the flush process is freeing memory by writing dirty memory pages out to disk.

Server side. The host in this case is using a popular release of Linux called Ubuntu. The services on this host are provided by an Apache Web Server. This service facilitates the communication with the mobile application. Also, on this host a MySQL database has been configured to provide secondary storage to store the mobile device IDs. Of course a modular approach has been used to create the code. Below are all the components of the server which are used to implement this application:

cProc.sh: This bash script is used to restart the stopped process

register.php: This is the rest API exposed to mobile app used to store the device id of the Android Phone in the MySql DB.

This API accepts the GET Method and takes ‘id’ as an argument.

unregister.php: This is the rest API exposed to the mobile app used to delete the device id of the Android Phone in the MySql DB.

This API accepts the GET Method and takes ‘id’ as an argument.

db_config.php: This file has the configuration used to connect to the MySQL DB. This is not an API and it is used internally by other files which need to connect to DB.

cronjob.sh: This is a bash script which basically acts a cronjob which calls the process.sh every five seconds.

MCrypt.php: This is a library which uses AES-128 internally for encrypting and decrypting data. This is not exposed as API and it is only used internally with other files.

startProcess.php: This is a Rest API which is displayed to the user by the mobile application. This is used to start the process. It will internally call the ‘cProc.sh’ internally.

This API accepts the GET Method and takes the ‘pid’ as an argument.

stopProcess.php: This is a Rest API which is displayed to the user by the mobile application. This is used to stop the process. It will internally call the ‘sProc.sh’ internally.

This API accepts the GET Method and takes the ‘pid’ as an argument.

sProc.sh: This bash script is used to stop the running process.

killProcess.php: This is a Rest API which is displayed to the user by the mobile application. This is used to kill the process. It will internally call the ‘kProc.sh’ internally.

kProc.sh: This bash script is used to kill the running process.

process.sh: This is a bash script which is used to identify the process with a high CPU usage. This will also call the ‘kill.sh’ to suspend the process.

kill.sh: This is a bash script which actually suspends the process and calls the ‘notify.php’.

auth.php: This is not an API and it is the authentication library which contains the code for HTTP Basic Authentication along with AES-128 encryption. The credentials are transmitted as encrypted data.

notify.php: This is API calls the GCM (Goggle Cloud Messaging) to send the push notification. It gets the device ids from the MySQL DB.

Client side. In our design, the client is an Android Application. This Application has two screens i.e. the Login Screen and the Administration screen which appear below as screenshots in Figure 2.

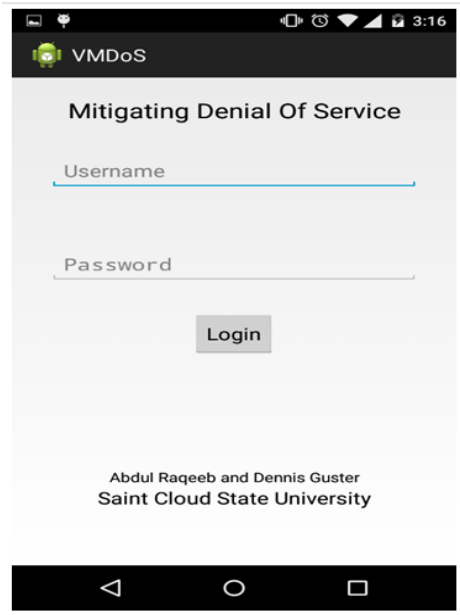

Figure 15. Login Screen

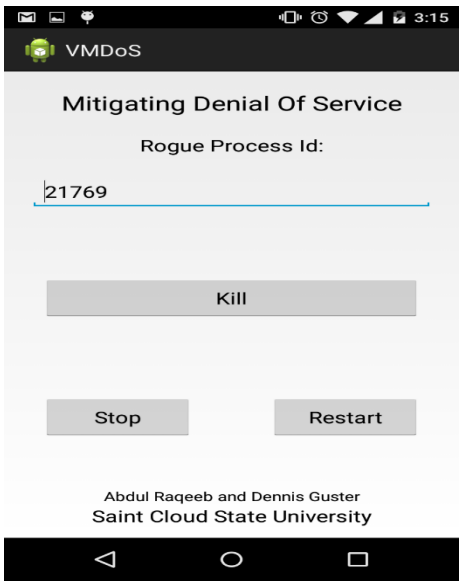

Figure 16. Administration Screen

The Login Screen is simply used to authenticate the application with the server. The login screen will be shown only once,. i.e., when the user installs the application. Therefore, when the user opens the APP for the second time it will not show the login screen again, instead the administration screen would then be displayed. Also, when the user logs in successfully for the first time the credentials are encrypted using AES-128 and then stored on the mobile device to support future communications.

Push Notifications for Rouge Process

Figure 17 illustrates how the push notification is used within this application. The push notification, also called the server push notification, is the delivery of information from a software application to a computing device, without a specific request from the client. In our design, push notifications are used to alert the user about the existence of a rogue process.

After a successful login, the mobile device will register with the GCM, upon a successful registration, the GCM will then return a device id to the mobile application. This ID will be used to uniquely identify the device within the cloud. The mobile application will then send a request to the server along with this device id and then store it in the database (at this point the register.php api is called).

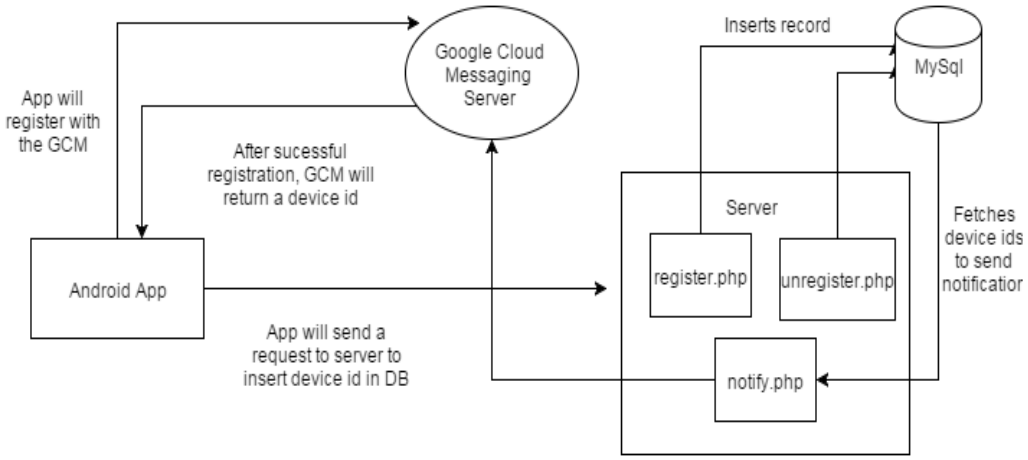

Figure 17. Architecture of Push Notification of Rouge Process

Next, when the server finds any rogue process it can, it will then send the push notification using the device id (again the notify.php api is called to notify the registered devices). Of course the object Notify.php has the code needed to send the request to the GCM. Below is an example of a request to the GCM:

GCM API : https://android.googleapis.com/gcm/send
Method : POST
Parameters:
registration_ids : is the array of device ids
data : is the array of key value pair (Ex: “message” =>”Rogue process found”)
Header : Authorization: key=”This key can be found on the Google API Console

Below, in Figure 18, is an example of the Android mobile device screenshot after the rogue process has been killed.

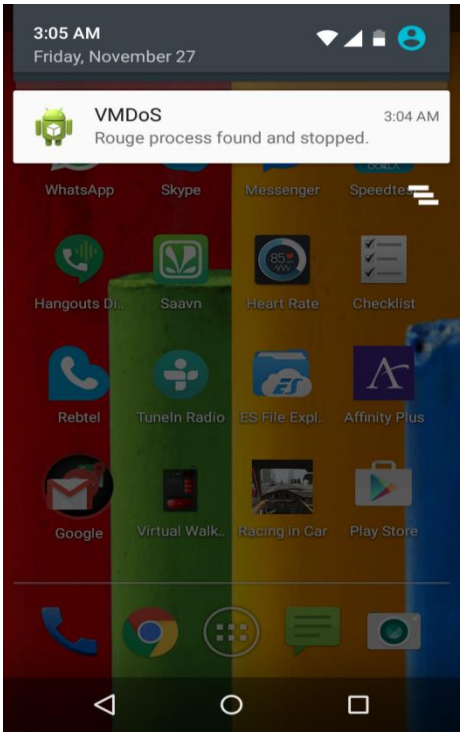

Figure 18. Rouge Process Killed

Encryption

Because this project attempted to use object oriented programming whenever possible a well-respected library [9] was used for encryption on both the mobile app and the server (GitHub, 2014). This library makes use of AES 128. When the mobile application sends a request to the server the data is encrypted before being transmitted. For example, one of the parameters to be passed would be the process ID which could be considered to be sensitive data. So, under this scenario, every process ID is encrypted and then sent over the REST API, of course then the, REST API is capable of decrypting the data.

The Initial Vector and Key both are initially shared between mobile app and the server. Below is a sample implementation of the Encryption on the mobile app.

```
String credentials = MCrypt.bytesToHex(mcrypt.encrypt("raqeeb")) + ":"+
MCrypt.bytesToHex(mcrypt.encrypt("superman2"));
```

The decryption on the server side is accomplished with the logic below:

```
$mcrypt = new MCrypt();
$username =$mcrypt->decrypt($user);
$password = $mcrypt->decrypt($pass);
```

Authentication for Accessing Mobile Application

In this application, authentication is done by the HTTP Basic Auth routine in conjunction with the AES-128 encryption class. The main file for handling the authentication and login API on the server is “auth.php.” Every request from the mobile app is then authenticated using that HTTP Basic Auth routine. Following is the example of adding the Auth header to the mobile application. Below is the sample java code for a request on client side:

```
credentials = MCrypt.bytesToHex(mcrypt.encrypt("raqeeb")) + ":"                +
MCrypt.bytesToHex(mcrypt.encrypt("superman2"));

String credBase64 = Base64.encodeToString(
credentials.getBytes(), Base64.DEFAULT).replace("\n","");
HttpClient httpclient = new DefaultHttpClient();
HttpGet httpGet = new HttpGet(params[0]);
httpGet.setHeader("Authorization", "Basic " + credBase64);
HttpResponse response = httpclient.execute(httpGet);
```

The above credentials are stored in the persistent memory of mobile device once the user has successfully logged in. This facilitates efficiency when evaluating future requests to the server. Figure 19 shows the sample PHP code used to handle the authentication requests.

```
function pc_validate($user,$pass) {


	$mcrypt = new MCrypt();
	$username =$mcrypt->decrypt($user);
	$password = $mcrypt->decrypt($pass);
	$pass_md5 = md5($password);

	$query = "select * from users where username = '$username' and password
= '$pass_md5'";

	$result=mysql_query($query);
	$count=mysql_num_rows($result);

	if($count==1)
	{
		return true;
	}
	else
	{
	 return false;
	}
}
```

Figure 19. PHP Code Used for Authentication Requests

To achieve better security, the username and password have also been encrypted using the AES-128 class before being transmitted to the server. In turn, the server will handle this request by decrypting and following the rest of the authentication process. A much needed future addition to this project would be implementing an administration control panel for controlling user accounts within the mobile application. Keeping this in mind the users are being stored in a MYSQL table called ‘users’. To ensure added security the passwords are stored as a MD5 hash.

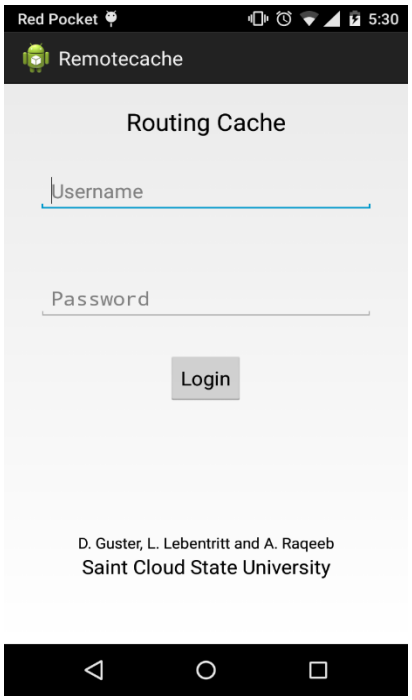

Figure 20. Authentication Process for Mobile App

Chapter IV: Conclusion and Future Work

The results generally indicate that an indirect measurement strategy such as using the router cache can detect attacks that may not trip an intrusion detection system. This is due in part to trying to minimize false positives within intrusion detection system and the fact that hacker have learned to utilize less intense attacks to minimize the probability of getting caught and to minimize their foot print from a forensics point of view. The data evaluated revealed that scenario. There were typically about 10 attacks per hour that did not trip the intrusion detection system, but were caught by the router cache. However, there were also several events per hour where the IP address was not on the whitelist and hence was mistakenly identified as an attack. Future research by the authors needs to address how the whitelist can be expanded beyond the list of current trusted domains. In many cases, outside address are related to a service provider such as Charter and Charter related client addresses in the authors' service area could be added to the white list. Also, use of reoccurring client addresses if used ethically could be added as well.

Using the router cache for this purpose places an additional burden on that cache in the sense it will get read more often. So it will be important to make sure the cache is well tuned. On the authors' system the virtual machine tested was getting an excellent hit ratio of about 95% in each direction. This is a good indicator of the health of the configuration and hopefully can be maintained when the router cache is also used to identify intrusions.

Last, for this technique to be viable, a quick and convenient interface is required. It makes sense to develop this interface as a mobile app. Equipment rooms are seldom manned 24/7 anymore, so assuming that a systems administrator carried a smart phone at all times,

intrusion could be identified and dealt with remotely. Certainly, this remote management would require that sensitive information travel the airways. Therefore, the design presented herein attempted to use encryption mechanisms to ensure secure transmission of data. Further, it was clear that this security tool could be just one of many that could be managed via a mobile app. In fact this is the second tool developed by the authors, see Guster et al. (2015). To accommodate such a mission, creating the MYSQL data base as a depository of security related data will allow the sharing of such data across multiple security tools deployed as mobile apps.

Hackers are consistently use a dynamic strategy; in other words, they are constantly devising new attack techniques and are aware of the importance of minimizing their attack fingerprint. Indirect methods such as this can be effective because adding to the routing cache table can be viewed as a subsequent event to a primary event such as attacking the secure shell daemon. This is an unconventional strategy and a hacker would really need to think outside the box to thwart its effectiveness. Ultimately, this makes it more difficult for a hacker to: go undetected, cover their tracks and ultimately succeed.

References

Benvenuti, C. (2006). *Understanding LINUX network internals*. Sebastopol, CA: O'Reilly Media, Inc.

Bernat, V. (2011a). *Tuning Linux IPv4 route cache*. Retrieved from http://vincent.bernat.im/en/blog/2011-ipv4-route-cache-linux.html.

Bernat, V. (2011b). *Tuning Linux IPv4 route cache*. Retrieved from: https://github.com/vincentbernat/www.luffy.cx/blob/master/content/en/blog/2011-ipv4-route-cache-linux.html

Cheswick, W. R., & Bellovin, S. M. (1994). *Firewalls and internet security: Repelling the wily hacker*. Boston, MA: Pearson.

Cho, Y., Qu, G., & Wu, Y. (2012). Insider threats against trust mechanism with watchdog and defending approaches in wireless sensor networks. *Proceedings IEEE CS Security and Privacy Workshops*, SPW 2012), 134-141. doi:10.1109/SPW.2012.32

Clinton, L. (2013). *The evolution of the cyber threat and public policy*. BrightTalk. Retrieved from: https://www.brighttalk.com/community/it-security/webcast/8883/67297

Dev, L. (2014). *The five most common cyberattack myths-revealed*. Retrieved from: http://www.cybereason.com/five_most_common_cyberattacks_myths_revealed/.

Geier, E. (2015). *10 Android apps for Linux Server Admins*. Retrieved from http://www.linuxplanet.com/linuxplanet/reviews/7301/2.

Guster, D., Abdul, R., & Rice, E. (2015). Mitigating virtual machine denial of service attacks from mobile apps. *Journal of Network and Information Security*, *3*(2), 21-31.

GitHub. (2014). *Android-PHP-Encrypt-Decrypt*. Retrieved from https://github.com/serpro/Android-PHP-Encrypt-Decrypt/

Hafner, K., & Lyon, M. (1996). Where wizards stay up late: The origins of the internet. New York, NY: Simon and Schuster.

Jichkar, M. A., & Chandak, D. B. (2014). *An implementation on detection of trusted service provider in mobile ad-hoc networks*. doi:10.14445/22315381/IJETT-V11P213

Kargl, F., Maier, J., Schlott, S., & Weber, M. (2001). *Protecting web servers from distributed denial of service attacks*. Retrieved from https://gixtools.net/ddos-mitigation/

Meyer, D., & Partan, A. (2003). *S-BGP/soBGP Panel: What do we really need and how do we architect a compromise to get it?* NANOG 0306.

Munsell, A. (2014). *Setup hack and HHVM on digital ocean*. Retrieved from https://www.andrewmunsell.com/blog/setup-hack-and-hhvm-on-digital-ocean/

Newspaint. (2013). *How to diagnose high Sys CPU on Linus*. Retrieved from https://newspaint.wordpress.com/2013/07/24/how-to-diagnose-high-sys-cpu-on-linux/

Nguyen, B. (2004). *LINUX file system hierarchy*. Retrieved from http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html

Saha, S., Lukyanenko, A., & Ylä-Jääski, A. (2015). Efficient cache availability management in Information-Centric Networks. *Computer Networks, 84*, 32-45. doi:10.1016/j.comnet.2015.04.005.

Shuo, C., Jun, X., Kalbarczyk, Z., & Iyer, R. K. (2006). Security vulnerabilities: From analysis to detection and masking techniques. *Proceedings of the IEEE*, 94(2), 407-418. doi:10.1109/JPROC.2005.862473.

Snader, R., & Borisov, N. (2008), A tune-up for Tor: Improving security and performance in the Tor network. *Proceedings of the Network and Distributed Security Symposium - NDSS'08, Internet Society*.

Symantec. (n.d.) *Advanced Persistent Threat (APT): The uninvited guest*. Retrieved from http://www.symantec.com/theme.jsp?themeid=apt-infographic-1

Teller, T. (2012). The Biggest Cybersecurity Threats of 2013. Forbes. Retrieved from http://www.forbes.com/sites/ciocentral/2012/12/05/the-biggest-cybersecurity-threats-of-2013-2/

Zhang, Y., Mao, Z., & Wang, J.). (2007). A firewall for routers: Protecting against routing misbehavior. *Proceedings of the International Conference on Dependable Systems and Networks*, (Proceedings - 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2007), 20-29. doi:10.1109/DSN.2007.7

Appendix

Application Name: Router Cache Android Application

Login.java

```
package com.bcrl.rcache;


import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;


import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONException;
import org.json.JSONObject;



import com.bcrl.rcache.MainActivity.WebRequest;
import com.bcrl.rcache.R;


import android.app.Activity;
import android.content.Intent;
```

```
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.os.AsyncTask;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.util.Base64;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class Login extends Activity {

        Button loginBtn;
        String SERVER_URL = "http://10.31.7.11/student/paper2/";
        WebReq_Return returnInterface;
        MCrypt mcrypt;
        EditText uEt, pEt;
        String username, password;
        SharedPreferences settings;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login);
        settings = PreferenceManager
                        .getDefaultSharedPreferences(getApplicationContext());
        String user = settings.getString("username", null);
        String pass = settings.getString("password", null);

        if(user != null && pass != null){
                Intent intent = new Intent(Login.this,
                                MainActivity.class);
                startActivity(intent);
                finish();
        }

        final Editor edit = settings.edit();
        loginBtn = (Button) findViewById(R.id.button1);
        mcrypt = new MCrypt();
        uEt = (EditText) findViewById(R.id.editText1);
        pEt = (EditText) findViewById(R.id.editText2);
```

```
loginBtn.setOnClickListener(new OnClickListener() {

        @Override
        public void onClick(View v) {
                createLogin();

        }
});

returnInterface = new WebReq_Return() {
        @Override
        public void setResult(String result) {
                try {
                        JSONObject jo = new JSONObject(result);
                        String code = jo.getString("code");
                        if (Integer.parseInt(code) == 200) {
                                runOnUiThread(new Runnable() {
                                        public void run() {
                                                edit.putString("username",
username);
```

```
                edit.putString("password",
password);
                edit.commit();
                Intent intent = new
Intent(Login.this,
        MainActivity.class);
                startActivity(intent);
                finish();
            }
        });
    } else {
        runOnUiThread(new Runnable() {
            public void run() {
        Toast.makeText(getApplicationContext(),
                "Invalid
credentials",
        Toast.LENGTH_SHORT).show();
            }
        });
```

```
                        }
                    } catch (JSONException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
            };

        }

        public void createLogin() {
            try {
                username = MCrypt.bytesToHex(mcrypt.encrypt(uEt.getText()
                        .toString()));
                password = MCrypt.bytesToHex(mcrypt.encrypt(pEt.getText()
                        .toString()));
                Log.d("url", SERVER_URL + "auth.php?user=" + username +
"&pass="
                        + password);
                WebRequest wr = new WebRequest();
                wr.execute(SERVER_URL + "auth.php?user=" + username +
"&pass="
```

```
                + password);


        } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}

public class WebRequest extends AsyncTask<String, Void, Void> {


        @Override
        protected Void doInBackground(String... params) {
                // TODO Auto-generated method stub
                InputStream isr = null;
                String result = "";
                long start, end;
                start = System.currentTimeMillis();
                try {


                        // create HttpClient
                        HttpClient httpclient = new DefaultHttpClient();
```

```
                HttpGet httpGet = new HttpGet(params[0]);
                HttpResponse response = httpclient.execute(httpGet);

                // receive response as inputStream
                isr = response.getEntity().getContent();

                // convert inputstream to String
                BufferedReader reader = new BufferedReader(
                                new InputStreamReader(isr, "iso-8859-
1"), 8);
                StringBuilder sb = new StringBuilder();
                String line = null;
                while ((line = reader.readLine()) != null) {
                        sb.append(line + "\n");
                }

                isr.close();
                result = sb.toString();

                Log.d("output", result);
                end = System.currentTimeMillis();
            } catch (Exception ex) {
```

```
                    Log.d("exception", ex.toString());
                    end = System.currentTimeMillis();
                }
                returnInterface.setResult(result);
                return null;
            }

        }

        public interface WebReq_Return {
            void setResult(String date);
        }
}
```

MainActivity.java

```
package com.bcrl.rcache;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
```

```
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONException;
import org.json.JSONObject;

import com.google.android.gcm.GCMRegistrar;
import com.bcrl.rcache.R;

import android.app.Activity;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Base64;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
```

```
public class MainActivity extends Activity {

    EditText pidET;
    String SERVER_URL = "http://10.31.7.11/student/paper2/";
    Button perBtn, authBtn, Fail2BanBtn, cacheBtn;
    TextView in_eff, out_eff;
    WebReq_Return returnInterface;
    MCrypt mcrypt;
    String credentials, username, password;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        GCMRegistrar.checkDevice(this);
        GCMRegistrar.checkManifest(this);
        final String regId = GCMRegistrar.getRegistrationId(this);
        GCMRegistrar.register(getApplicationContext(),
                GCMIntentService.SENDER_ID);

        perBtn = (Button) findViewById(R.id.button3);
```

```
Fail2BanBtn = (Button) findViewById(R.id.button2);
authBtn = (Button) findViewById(R.id.button1);
cacheBtn = (Button) findViewById(R.id.button4);
in_eff = (TextView) findViewById(R.id.textView4);
out_eff = (TextView) findViewById(R.id.TextView01);

mcrypt = new MCrypt();

perBtn.setOnClickListener(new OnClickListener() {

	@Override
	public void onClick(View v) {
		fetchPerformance();

	}
});

Fail2BanBtn.setOnClickListener(new OnClickListener() {

	@Override
	public void onClick(View v) {
		Intent intent = new Intent(MainActivity.this,
```

```
                                Fail2Ban.class);
                intent.putExtra("type", "fail");
                startActivity(intent);
        }
});

authBtn.setOnClickListener(new OnClickListener() {

        @Override
        public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
                                Fail2Ban.class);
                intent.putExtra("type", "auth");
                startActivity(intent);

        }
});
cacheBtn.setOnClickListener(new OnClickListener() {

        @Override
        public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
```

```
                                        Fail2Ban.class);
                    intent.putExtra("type", "cache");
                    startActivity(intent);

                }
            });

            returnInterface = new WebReq_Return() {
                @Override
                public void setResult(final String result) {
                    runOnUiThread(new Runnable() {
                        public void run() {

                                        Toast.makeText(getApplicationContext(),
result,

        Toast.LENGTH_LONG).show();

                        }
                    });

                }
```

```
        };

        if (getIntent().getExtras() != null) {
            String message = getIntent().getExtras().getString("name");
        }
    }

    public void fetchPerformance() {
        try {

            credentials = MCrypt.bytesToHex(mcrypt.encrypt("raqeeb")) +
":"
                    +
MCrypt.bytesToHex(mcrypt.encrypt("superman2"));
            Log.d("url", SERVER_URL + "performance.php");
            WebRequest wr = new WebRequest();
            wr.execute(SERVER_URL + "performance.php");
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

```
}

public class WebRequest extends AsyncTask<String, Void, String> {

    @Override
    protected void onPostExecute(String result) {
        try {
            JSONObject jo = new JSONObject(result);
            in_eff.setText("In efficiency:" +
jo.getString("in_efficency")
                    + "%");
            out_eff.setText("Out efficiency:"
                    + jo.getString("out_efficency") + "%");
        } catch (JSONException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    };

    @Override
    protected String doInBackground(String... params) {
        // TODO Auto-generated method stub
```

```
                InputStream isr = null;
                String result = "";
                long start, end;
                start = System.currentTimeMillis();
                try {
                        String credBase64 = Base64.encodeToString(
                                        credentials.getBytes(),
Base64.DEFAULT).replace("\n",
                                        "");
                        // create HttpClient
                        HttpClient httpclient = new DefaultHttpClient();

                        HttpGet httpGet = new HttpGet(params[0]);
                        httpGet.setHeader("Authorization", "Basic " +
credBase64);
                        HttpResponse response = httpclient.execute(httpGet);

                        // receive response as inputStream
                        isr = response.getEntity().getContent();

                        // convert inputstream to String
                        BufferedReader reader = new BufferedReader(
```

```
                                        new InputStreamReader(isr, "iso-8859-
1"), 8);
                        StringBuilder sb = new StringBuilder();
                        String line = null;
                        while ((line = reader.readLine()) != null) {
                                sb.append(line + "\n");
                        }

                        isr.close();
                        result = sb.toString();

                        Log.d("output", result);
                        end = System.currentTimeMillis();
                } catch (Exception ex) {
                        Log.d("exception", ex.toString());
                        end = System.currentTimeMillis();
                }
                String resp = "Response Time" + (end - start) + " millis";
                returnInterface.setResult("CMD exe time -" + resp);
                return result;
        }
```

```
        }

}
```

GCMIntentService.java

```
package com.bcrl.rcache;

import java.util.Timer;
import java.util.TimerTask;

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.os.PowerManager;
import android.util.Log;
```

```
import com.google.android.gcm.GCMBaseIntentService;
import com.bcrl.rcache.R;

public class GCMIntentService extends GCMBaseIntentService {

	private static final String TAG = "GCM Tutorial::Service";
	private static final String SUrl = "http://10.31.7.11/student/paper2/";

	// Use your PROJECT ID from Google API into SENDER_ID
	public static final String SENDER_ID = "1037879414870";

	public GCMIntentService() {
		super(SENDER_ID);
	}

	@Override
	protected void onRegistered(Context context, String registrationId) {

		Log.i(TAG, "onRegistered: registrationId=" + registrationId);

		HttpClient mClient = new DefaultHttpClient();
```

```
                HttpGet get = new HttpGet(SUrl + "/register.php?id=" + registrationId);
                try {
                        mClient.execute(get);
                        HttpResponse res = mClient.execute(get);

                } catch (Exception e) {
                        // TODO: handle exception
                }

        }

        @Override
        protected void onUnregistered(Context context, String registrationId) {

                Log.i(TAG, "onUnregistered: registrationId=" + registrationId);
                HttpClient mClient = new DefaultHttpClient();

                HttpGet get = new HttpGet(SUrl + "/unregister.php?id=" +
registrationId);
                try {
                        mClient.execute(get);
```

```
                HttpResponse res = mClient.execute(get);

        } catch (Exception e) {
                // TODO: handle exception
        }

}

@Override
protected void onMessage(Context context, Intent data) {
        String message;
        String message1;
        String message2;

        // Message from PHP server
        message = data.getStringExtra("message");
        message1 = data.getStringExtra("message1");
        message2 = data.getStringExtra("message2");

        // Starts the activity on notification click
        int icon = R.drawable.ic_launcher;
        long when = System.currentTimeMillis();
```

```
                // NotificationManager notificationManager = (NotificationManager)
                context.getSystemService(Context.NOTIFICATION_SERVICE);
                Notification notification = new Notification(icon, message, when);

                String title = context.getString(R.string.app_name);

                Intent notificationIntent = new Intent(context, MainActivity.class);
                notificationIntent.putExtra("name", message);
                notificationIntent.putExtra("pid", message1);
                notificationIntent.putExtra("link", message2);

                // set intent so it does not start a new activity
                notificationIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP
                                | Intent.FLAG_ACTIVITY_SINGLE_TOP);
                PendingIntent intent = PendingIntent.getActivity(context, 0,
                                notificationIntent,
PendingIntent.FLAG_UPDATE_CURRENT);
                notification.setLatestEventInfo(context, title, message, intent);
                notification.flags |= Notification.FLAG_AUTO_CANCEL;

                // Play default notification sound
                notification.defaults |= Notification.DEFAULT_SOUND;
```

```
                    // Vibrate if vibrate is enabled
                    notification.defaults |= Notification.DEFAULT_VIBRATE;
                    // notificationManager.notify(0, notification);
                    NotificationManager manager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
                    manager.notify(R.string.app_name, notification);

                    {
                        // Wake Android Device when notification received
                        PowerManager pm = (PowerManager) context

        .getSystemService(Context.POWER_SERVICE);
                        final PowerManager.WakeLock mWakelock =
pm.newWakeLock(
                                PowerManager.FULL_WAKE_LOCK
                                        |
PowerManager.ACQUIRE_CAUSES_WAKEUP, "GCM_PUSH");
                        mWakelock.acquire();

                        // Timer before putting Android Device to sleep mode.
                        Timer timer = new Timer();
```

```
                TimerTask task = new TimerTask() {
                    public void run() {
                        mWakelock.release();
                    }
                };
                timer.schedule(task, 5000);

            }

        }

        @Override
        protected void onError(Context arg0, String errorId) {

            Log.e(TAG, "onError: errorId=" + errorId);
        }

}
```

MCrypt.java

```
/*
 *
 */
```

```
package com.bcrl.rcache;

import java.security.NoSuchAlgorithmException;

import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class MCrypt {

    static char[] HEX_CHARS = {'0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f'};

    private String iv = "fedcba9876543210";//Dummy iv (CHANGE IT!)
    private IvParameterSpec ivspec;
    private SecretKeySpec keyspec;
    private Cipher cipher;

    private String SecretKey = "0123456789abcdef";//Dummy secretKey (CHANGE
IT!)

    public MCrypt()
```

```
{
        ivspec = new IvParameterSpec(iv.getBytes());

        keyspec = new SecretKeySpec(SecretKey.getBytes(), "AES");

        try {
                cipher = Cipher.getInstance("AES/CBC/NoPadding");
        } catch (NoSuchAlgorithmException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (NoSuchPaddingException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}

public byte[] encrypt(String text) throws Exception
{
        if(text == null || text.length() == 0)
                throw new Exception("Empty string");

        byte[] encrypted = null;
```

```
        try {
            cipher.init(Cipher.ENCRYPT_MODE, keyspec, ivspec);

            encrypted = cipher.doFinal(padString(text).getBytes());
        } catch (Exception e)
        {
            throw new Exception("[encrypt] " + e.getMessage());
        }

        return encrypted;
    }

    public byte[] decrypt(String code) throws Exception
    {
        if(code == null || code.length() == 0)
            throw new Exception("Empty string");

        byte[] decrypted = null;

        try {
            cipher.init(Cipher.DECRYPT_MODE, keyspec, ivspec);
```

```
            decrypted = cipher.doFinal(hexToBytes(code));
            //Remove trailing zeroes
            if( decrypted.length > 0)
            {
               int trim = 0;
               for( int i = decrypted.length - 1; i >= 0; i-- ) if( decrypted[i] == 0 )
trim++;

               if( trim > 0 )
               {
                  byte[] newArray = new byte[decrypted.length - trim];
                  System.arraycopy(decrypted, 0, newArray, 0, decrypted.length -
trim);
                  decrypted = newArray;
               }
            }
         } catch (Exception e)
         {
            throw new Exception("[decrypt] " + e.getMessage());
         }
         return decrypted;
```

```
}


public static String bytesToHex(byte[] buf)
{
   char[] chars = new char[2 * buf.length];
   for (int i = 0; i < buf.length; ++i)
   {
      chars[2 * i] = HEX_CHARS[(buf[i] & 0xF0) >>> 4];
      chars[2 * i + 1] = HEX_CHARS[buf[i] & 0x0F];
   }
   return new String(chars);
}


public static byte[] hexToBytes(String str) {
      if (str==null) {
            return null;
      } else if (str.length() < 2) {
            return null;
      } else {
            int len = str.length() / 2;
```

```
            byte[] buffer = new byte[len];
            for (int i=0; i<len; i++) {
                buffer[i] = (byte) Integer.parseInt(str.substring(i*2,i*2+2),16);
            }
            return buffer;
        }
}
```

```
private static String padString(String source)
{
  char paddingChar = 0;
  int size = 16;
  int x = source.length() % size;
  int padLength = size - x;

  for (int i = 0; i < padLength; i++)
  {
        source += paddingChar;
  }
```

```
        return source;
      }
}
```

FObject.java

```
package com.bcrl.rcache;

public class FObject {

        public String status;
        public String ip;
        public String timestamp;
        public String orgname;
        public String address;
        public String city;
        public String state;
        public String pcode;
        public String country;

}
```

Fail2Ban.java

```
package com.bcrl.rcache;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.List;

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import com.bcrl.rcache.MainActivity.WebRequest;

import android.app.Activity;
import android.os.AsyncTask;
```

```
import android.os.Bundle;
import android.util.Base64;
import android.util.Log;
import android.widget.ListView;

public class Fail2Ban extends Activity {
	String credentials;
	MCrypt mcrypt;
	String SERVER_URL = "http://10.31.7.11/student/paper2/";
	List<FObject> data;
	List<CObject> cdata;
	List<AObject> adata;
	ListView lv;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setContentView(R.layout.Fail2Ban);
		lv = (ListView) findViewById(R.id.listview);
		mcrypt = new MCrypt();
		data = new ArrayList<FObject>();
```

```
        cdata = new ArrayList<CObject>();
        adata = new ArrayList<AObject>();

        if (getIntent().getExtras() != null) {
            if (getIntent().getExtras().getString("type").equals("fail")) {
                callFail2Ban();
            }

            if (getIntent().getExtras().getString("type").equals("auth")) {
                callAuth();
            }

            if (getIntent().getExtras().getString("type").equals("cache")) {
                callCache();
            }
        }

}

public void callFail2Ban() {
        try {
            WebRequest wr = new WebRequest();
```

```
                credentials = MCrypt.bytesToHex(mcrypt.encrypt("raqeeb")) +
":"
                                +
MCrypt.bytesToHex(mcrypt.encrypt("superman2"));
                wr.execute(SERVER_URL + "/Fail2Ban.php");
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

        public void callAuth() {
            try {
                WebRequest wr = new WebRequest();
                credentials = MCrypt.bytesToHex(mcrypt.encrypt("raqeeb")) +
":"
                                +
MCrypt.bytesToHex(mcrypt.encrypt("superman2"));
                wr.execute(SERVER_URL + "/auth_log.php");
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
```

```
                }
        }

        public void callCache() {
                try {
                        WebRequest wr = new WebRequest();
                        credentials = MCrypt.bytesToHex(mcrypt.encrypt("raqeeb")) +
":"
                                        +
MCrypt.bytesToHex(mcrypt.encrypt("superman2"));

    wr.execute("http://studentweb.bcrl.stcloudstate.edu/~raqueeb.abdul/cache.php");
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }

        public void parseFail2Ban(JSONArray ja) {
                for (int i = ja.length() - 1; i >= 0; i--) {
                        try {
                                JSONObject job = ja.getJSONObject(i);
```

```
                    FObject fa = new FObject();
                    fa.status = job.getString("status");
                    fa.ip = job.getString("ip");
                    fa.timestamp = job.getString("timestamp");
                    fa.orgname = job.getString("OrgName");
                    fa.address = job.getString("Address");
                    fa.city = job.getString("City");
                    fa.state = job.getString("StateProv");
                    fa.pcode = job.getString("PostalCode");
                    fa.country = job.getString("Country");
                    data.add(fa);
                } catch (JSONException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
            Log.d("test", "test");
            Fadapter fa = new Fadapter(getApplicationContext(), data);
            lv.setAdapter(fa);

}
```

```
public void parseAuth(JSONArray ja) {

	ja = reverseJSON(ja);
	for (int i = 1; i <= 20; i++) {
		try {
			JSONObject job = ja.getJSONObject(i);
			AObject fa = new AObject();
			fa.log = job.getString("log");
			fa.timestamp = job.getString("timestamp");
			adata.add(fa);

		} catch (JSONException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
	AAdapter ca = new AAdapter(getApplicationContext(), adata);
	lv.setAdapter(ca);
}

public void pareCache(JSONArray ja) {
	for (int i = 0; i < ja.length(); i++) {
```

```
        try {

                JSONObject job = ja.getJSONObject(i);
                if (job.getString("blacklisted").equals("1")) {
                        CObject fa = new CObject();
                        fa.blacklist = job.getString("blacklisted");
                        fa.src = job.getString("src");
                        fa.dst = job.getString("dst");
                        fa.flags = job.getString("flags");
                        fa.metric = job.getString("metric");
                        fa.ref = job.getString("ref");
                        fa.gateway = job.getString("gateway");
                        fa.user = job.getString("use");
                        fa.iface = job.getString("iface");
                        cdata.add(fa);
                }
        } catch (JSONException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}
CAdapter ca = new CAdapter(getApplicationContext(), cdata);
```

```
            lv.setAdapter(ca);
        }

        public class WebRequest extends AsyncTask<String, Void, String> {

            @Override
            protected void onPostExecute(String result) {
                try {
                    JSONArray ja = new JSONArray(result);
                    if
(getIntent().getExtras().getString("type").equals("fail")) {
                        parseFail2Ban(ja);
                    }

                    if
(getIntent().getExtras().getString("type").equals("auth")) {
                        parseAuth(ja);
                    }

                    if
(getIntent().getExtras().getString("type").equals("cache")) {
                        pareCache(ja);
```

```
                }

            } catch (JSONException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        };

        @Override
        protected String doInBackground(String... params) {
            // TODO Auto-generated method stub
            InputStream isr = null;
            String result = "";
            long start, end;
            start = System.currentTimeMillis();
            try {
                String credBase64 = Base64.encodeToString(
                        credentials.getBytes(),
Base64.DEFAULT).replace("\n",
                        "");
                // create HttpClient
                HttpClient httpclient = new DefaultHttpClient();
```

```
HttpGet httpGet = new HttpGet(params[0]);
httpGet.setHeader("Authorization", "Basic " +
credBase64);
HttpResponse response = httpclient.execute(httpGet);

// receive response as inputStream
isr = response.getEntity().getContent();

// convert inputstream to String
BufferedReader reader = new BufferedReader(
                new InputStreamReader(isr, "iso-8859-
1"), 8);
StringBuilder sb = new StringBuilder();
String line = null;
while ((line = reader.readLine()) != null) {
        sb.append(line + "\n");
}

isr.close();
result = sb.toString();
```

```
                Log.d("output", result);
                end = System.currentTimeMillis();
            } catch (Exception ex) {
                Log.d("exception", ex.toString());
                end = System.currentTimeMillis();
            }

            return result;
        }

}

public JSONArray reverseJSON(JSONArray ja) {
        JSONArray newJsonArray = new JSONArray();
        for (int i = ja.length() - 1; i >= 0; i--) {
            try {
                newJsonArray.put(ja.get(i));
            } catch (JSONException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
```

```
                return newJsonArray;


        }
}
```

FAdapter.java

```
package com.bcrl.rcache;


import java.util.List;


import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.LinearLayout;
import android.widget.ListAdapter;
import android.widget.TextView;


public class Fadapter extends BaseAdapter {
        Context context;
        List<FObject> data;
        private static LayoutInflater inflater = null;
```

```
public Fadapter(Context c, List<FObject> listArray) {
        this.context = c;
        this.data = listArray;
        inflater = (LayoutInflater) c
                        .getSystemService("layout_inflater");

}

@Override
public int getCount() {
        // TODO Auto-generated method stub
        return this.data.size();
}

@Override
public Object getItem(int paramInt) {
        // TODO Auto-generated method stub
        return this.data.get(paramInt);
}

@Override
```

```
        public long getItemId(int paramInt) {
            // TODO Auto-generated method stub
            return paramInt;
        }

        @Override
        public View getView(int paramInt, View paramView, ViewGroup
paramViewGroup) {
            TextView
ip,timestamp,status,orgname,address,city,state,pcode,country;
            LinearLayout messageView;
            View localView = paramView;
            if (localView == null) {
                localView = inflater.inflate(R.layout.fadapter, null);
            }
            ip = (TextView) localView.findViewById(R.id.ip);
            timestamp = (TextView) localView.findViewById(R.id.timestamp);
            status = (TextView) localView.findViewById(R.id.status);
            orgname = (TextView) localView.findViewById(R.id.orgname);
            address = (TextView) localView.findViewById(R.id.address);
            city = (TextView) localView.findViewById(R.id.city);
            state = (TextView) localView.findViewById(R.id.state);
```

```
                pcode = (TextView) localView.findViewById(R.id.pcode);
                country = (TextView) localView.findViewById(R.id.country);


                timestamp.setText("TimeStamp:"+data.get(paramInt).timestamp);
                ip.setText("IP:"+data.get(paramInt).ip);
                status.setText("Status:"+data.get(paramInt).status);
                orgname.setText("OrgName:"+data.get(paramInt).orgname);
                address.setText("Address:"+data.get(paramInt).address);
                city.setText("City:"+data.get(paramInt).city);
                state.setText("State:"+data.get(paramInt).state);
                pcode.setText("Pcode:"+data.get(paramInt).pcode);
                country.setText("Country:"+data.get(paramInt).country);
                return localView;
        }

}
```

CObject.java

```
package com.bcrl.rcache;


public class CObject {
```

```
        public String src;
        public String dst;
        public String gateway;
        public String flags;
        public String metric;
        public String ref;
        public String user;
        public String iface;
        public String blacklist;

}
```

CAdapter.java

```
package com.bcrl.rcache;

import java.util.List;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
```

```
import android.widget.LinearLayout;
import android.widget.ListAdapter;
import android.widget.TextView;

public class CAdapter extends BaseAdapter {
	Context context;
	List<CObject> data;
	private static LayoutInflater inflater = null;

	public CAdapter(Context c, List<CObject> listArray) {
		this.context = c;
		this.data = listArray;
		inflater = (LayoutInflater) c.getSystemService("layout_inflater");

	}

	@Override
	public int getCount() {
		// TODO Auto-generated method stub
		return this.data.size();
	}
```

```
		@Override
		public Object getItem(int paramInt) {
			// TODO Auto-generated method stub
			return this.data.get(paramInt);
		}

		@Override
		public long getItemId(int paramInt) {
			// TODO Auto-generated method stub
			return paramInt;
		}

		@Override
		public View getView(int paramInt, View paramView, ViewGroup
paramViewGroup) {
			TextView ip, timestamp, status, orgname, address, city, state, pcode,
country;
			LinearLayout messageView;
			View localView = paramView;
			if (localView == null) {
				localView = inflater.inflate(R.layout.fadapter, null);
			}
```

```
                ip = (TextView) localView.findViewById(R.id.ip);
                timestamp = (TextView) localView.findViewById(R.id.timestamp);
                status = (TextView) localView.findViewById(R.id.status);
                orgname = (TextView) localView.findViewById(R.id.orgname);
                address = (TextView) localView.findViewById(R.id.address);
                city = (TextView) localView.findViewById(R.id.city);
                state = (TextView) localView.findViewById(R.id.state);
                pcode = (TextView) localView.findViewById(R.id.pcode);
                country = (TextView) localView.findViewById(R.id.country);
                timestamp.setText("Src:" + data.get(paramInt).src);
                ip.setText("Dst:" + data.get(paramInt).dst);
                status.setText("Gateway:" + data.get(paramInt).gateway);
                orgname.setText("Flags:" + data.get(paramInt).flags);
                address.setText("Iface:" + data.get(paramInt).iface);
                city.setText("Metric:" + data.get(paramInt).metric);
                state.setText("Use:" + data.get(paramInt).user);
                pcode.setText("Ref:" + data.get(paramInt).ref);
                country.setText("Blacklist:" + data.get(paramInt).blacklist);
                return localView;
        }
}
```

AObject.java

```
package com.bcrl.rcache;

public class AObject {


	public String timestamp;

	public String log;


}
```

AAdapter.java

```
package com.bcrl.rcache;


import java.util.List;


import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.BaseAdapter;

import android.widget.LinearLayout;

import android.widget.TextView;


public class AAdapter extends BaseAdapter {

	Context context;
```

```
List<AObject> data;
private static LayoutInflater inflater = null;

public AAdapter(Context c, List<AObject> listArray) {
        this.context = c;
        this.data = listArray;
        inflater = (LayoutInflater) c.getSystemService("layout_inflater");

}

@Override
public int getCount() {
        // TODO Auto-generated method stub
        return this.data.size();
}

@Override
public Object getItem(int paramInt) {
        // TODO Auto-generated method stub
        return this.data.get(paramInt);
}
```

```
		@Override
		public long getItemId(int paramInt) {
			// TODO Auto-generated method stub
			return paramInt;
		}

		@Override
		public View getView(int paramInt, View paramView, ViewGroup
paramViewGroup) {
			TextView log, timestamp;
			LinearLayout messageView;
			View localView = paramView;
			if (localView == null) {
				localView = inflater.inflate(R.layout.aadapter, null);
			}
			log = (TextView) localView.findViewById(R.id.log);
			timestamp = (TextView) localView.findViewById(R.id.timestamp);
			timestamp.setText("Src:" + data.get(paramInt).timestamp);
			log.setText("Dst:" + data.get(paramInt).log);

			return localView;
		}
```

```
}
```

Login.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal"
            android:layout_marginTop="20dp"
            android:text="Routing Cache"
            android:textAppearance="?android:attr/textAppearanceLarge" />
```

```
    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:ems="10"
        android:hint="Username" >

        <requestFocus />
    </EditText>

    <EditText
        android:id="@+id/editText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:ems="10"
        android:hint="Password"
        android:inputType="textPassword" />

</LinearLayout>
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:orientation="vertical" >

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Login"
        android:layout_gravity="center_horizontal" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:layout_weight="0.5" >

    <TextView
```

```
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:text="D. Guster, L. Lebentritt and A. Raqeeb"
        android:textColor="#000" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:text="Saint Cloud State University"
        android:textColor="#000"
        android:textSize="18sp" />

  </LinearLayout>
</LinearLayout>
```

Fail2Ban.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <ListView
        android:id="@+id/listview"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </ListView>

</LinearLayout>
```

FAdapter.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding = "10dp"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@drawable/gradient"
```

```
android:orientation="vertical" >

<TextView
   android:id="@+id/timestamp"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:padding="5dp"
   android:text="TextView"
   android:textSize="18sp"
   android:textColor="#000" />
<TextView
   android:id="@+id/ip"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:padding="5dp"
   android:text="TextView"
   android:textSize="18sp"
   android:textColor="#000" />
<TextView
   android:id="@+id/status"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
```

```
    android:padding="5dp"
    android:text="TextView"
    android:textSize="18sp"
    android:textColor="#000" />
<TextView
    android:id="@+id/orgname"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="5dp"
    android:text="TextView"
    android:textSize="18sp"
    android:textColor="#000" />
<TextView
    android:id="@+id/address"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="5dp"
    android:text="TextView"
    android:textSize="18sp"
    android:textColor="#000" />
<TextView
    android:id="@+id/city"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="5dp"
    android:text="TextView"
    android:textSize="18sp"
    android:textColor="#000" />
<TextView
    android:id="@+id/state"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="5dp"
    android:text="TextView"
    android:textSize="18sp"
    android:textColor="#000" />
<TextView
    android:id="@+id/pcode"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="5dp"
    android:text="TextView"
    android:textSize="18sp"
    android:textColor="#000" />
```

```
        <TextView
            android:id="@+id/country"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="TextView"
            android:textSize="18sp"
            android:textColor="#000" />

    </LinearLayout>

</LinearLayout>
```

Activity_mail.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:gravity="center_horizontal"
        android:text="Routing Cache"
        android:textAppearance="?android:attr/textAppearanceLarge" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:orientation="horizontal" >

    <Button
        android:id="@+id/button3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:layout_weight="1"
        android:text="Reload" />

    <Button
        android:id="@+id/button4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:layout_margin="30dp"
        android:text="Cache" />

</LinearLayout>
```

```
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/TextView01"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="Out Efficiency:" />

        <TextView
            android:id="@+id/textView4"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="In Efficiency:" />
    </LinearLayout>
</LinearLayout>
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:orientation="horizontal" >

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:layout_weight="1"
        android:text="Auth" />

    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:layout_weight="1"
        android:text="Fail2Ban" />
```

```
</LinearLayout>

<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_weight="0.5"
   android:orientation="vertical" >

   <TextView
      android:id="@+id/textView2"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:gravity="center_horizontal"
      android:text="D. Guster, L. Lebentritt and A. Raqeeb"
      android:textColor="#000" />

   <TextView
      android:id="@+id/textView3"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:gravity="center_horizontal"
      android:text="Saint Cloud State University"
```

```
        android:textColor="#000"
        android:textSize="18sp" />
    </LinearLayout>

</LinearLayout>
```

AAdapter.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@drawable/gradient"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/timestamp"
            android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:padding="5dp"
        android:text="TextView"
        android:textColor="#000"
        android:textSize="18sp" />

    <TextView
        android:id="@+id/log"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dp"
        android:text="TextView"
        android:textColor="#000"
        android:textSize="18sp" />
  </LinearLayout>

</LinearLayout>
```

cache.php

```
<?php
header('Content-Type: application/json');

$cmd = "netstat -Cnre";
```

```
$data = shell_exec($cmd);
$data1 = explode("\n",$data);
$data2 = array_slice($data1,2);
foreach($data2 as $value){
$output[] = array_values(array_filter(explode(" ",$value),'not_empty_string'));
}
for($i=0;$i<count($output)-1;$i++)
{
        $arrayOutput['src'] = $output[$i][0];
        $arrayOutput['dst'] = $output[$i][1];
        $arrayOutput['gateway'] = $output[$i][2];
        if(count($output[$i])==8){
        $arrayOutput['flags'] = $output[$i][3];
        $arrayOutput['metric'] = $output[$i][4];
        $arrayOutput['ref'] = $output[$i][5];
        $arrayOutput['use'] = $output[$i][6];
        $arrayOutput['iface'] = $output[$i][7];
        }else{
        $arrayOutput['flags'] = "-";
        $arrayOutput['metric'] = $output[$i][3];
        $arrayOutput['ref'] = $output[$i][4];
        $arrayOutput['use'] = $output[$i][5];
```

```
            $arrayOutput['iface'] = $output[$i][6];
            }
            $arrayOutput['blacklisted'] = parse_ip($output[$i][0]);
            $final_output[] = $arrayOutput;
        }

        echo json_encode($final_output);

        function not_empty_string($s) {
          return $s !== "";
        }

        function parse_ip($ip){

        $ret = 1;
        $data = shell_exec('cat whitelist_rt.local');
        $data1 = explode("\n",$data);

        foreach ($data1 as $value){
        if(strlen($value)>1)
        $ips[] =preg_replace("/\.{2,}/",".",str_replace(".0",".",preg_replace("/[^0-
9.]/","",$value)));
```

```
}

foreach ($ips as $addr) {
    if (strpos($ip, $addr) === 0) {
       $ret =0;
    }
}

return $ret;
}
?>
```

Performance.php

```
<?php
include 'auth.php';
//validateAuth();
$cmd = 'lnstat -s1 -i1 -f rt_cache';
$data = shell_exec($cmd);
$data1 = explode("\n", $data);
$output = $data1[sizeof($data1)-2];
$result = str_replace(" ", "", $output);
$data1 = explode("|", $result);
$rtcache_in = $data1[1];
```

```
$rtcache_in_tot = $data1[2];
$rtcache_out = $data1[8];
$rtcache_out_tot= $data1[9];

if($rtcache_in == 0){
        $in_efficency = "100";
}
else
{
        $in_efficency=100-($rtcache_in_tot/$rtcache_in)*100;
}
if($rtcache_out == 0){
        $out_efficency = "100";
}
else{
        $out_efficency=100-($rtcache_out_tot/$rtcache_out)*100;
}

$final["in_efficency"]=$in_efficency;
$final["out_efficency"]=$out_efficency;

echo json_encode($final);
```

```
?>
```

notify.php

```
<?php
function notify($message,$pid){
include 'db_config.php';
$con= mysql_connect($host,$username,$password);
$location = getLoc($pid);
if(!$con)
{
die('couldnot connect'.mysql_error());
}
mysql_select_db("results1",$con);
$result=mysql_query("select regid from gcmReg");
while($row=mysql_fetch_assoc($result))
{
        $output[]=$row['regid'];
}
$url = 'https://android.googleapis.com/gcm/send';
$fields = array(
            'registration_ids'  =>$output,
```

```
                    'data'            => array( "message" => $message,"message1" => $pid
,"message2" => "jntu"  ),
                    );

        $headers = array(
                    'Authorization: key=AIzaSyA-
XeA33yWCWA_KHwQAfxah02cb7hrHPNo' ,
                    'Content-Type: application/json'
                );

        // Open connection
        $ch = curl_init();

        // Set the url, number of POST vars, POST data
        curl_setopt( $ch, CURLOPT_URL, $url );

        curl_setopt( $ch, CURLOPT_POST, true );
        curl_setopt( $ch, CURLOPT_HTTPHEADER, $headers);
        curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);

        curl_setopt( $ch, CURLOPT_POSTFIELDS, json_encode( $fields ) );
```

```
        // Execute post
        $result = curl_exec($ch);

        // Close connection
        curl_close($ch);

        echo $result;
        //print_r($location);

        $stmt = "INSERT INTO `Fail2Ban`(`status`,
`ip`,`OrgName`,`Address`,`City`,`StateProv`,`PostalCode`,`Country`) VALUES
('ban','".$pid."','".$location["OrgName"]."','".$location["Address"]."','".$location["City"]."','".$
location["StateProv"]."','".$location["PostalCode"]."','".$location["Country"]."')";

        if (mysql_query($stmt, $con)) {

        } else {
            echo "Error: " . $stmt . "<br>" . mysql_error($con);
        }

        mysql_close($con);
```

```
        }


        function getLoc($ip){
                $data = shell_exec("whois ".$ip." | grep
'OrgNam\|City\|Address\|StateProv\|PostalCode\|Country'");
                $data1 =  explode("\n",$data);
                foreach($data1 as $value){
                $data2 = explode(":",$value);
                if(sizeof($data2)>=2)
                $output[$data2[0]] = trim($data2[1]);
        }


        return $output;
        }
?>
```

Fail2Ban.php

```
<?php
header('Content-Type: application/json');
include 'db_config.php';
include 'auth.php';
//validateAuth();
$con= mysql_connect($host,$username,$password);
```

```
if(!$con)
{
die('couldnot connect'.mysql_error());
}
mysql_select_db("results1",$con);
$queryStr = "select * from Fail2Ban;";
$result=mysql_query($queryStr);

while($row=mysql_fetch_assoc($result))
{
$output[]=$row;
}

echo json_encode($output);
?>
```

auth_log.php

```
<?php
header('Content-Type: application/json');
error_reporting( 0 );
include 'auth.php';
//validateAuth();
$cmd = "sudo cat /var/log/auth.log";
```

```
if(isset($_GET["query"])){
$cmd=$cmd."| grep ". $_GET["query"];
}

$data = shell_exec($cmd);
$data1 = explode("\n",$data);
$data2 = array_slice($data1,2);

foreach($data2 as $value){
$output_1 = explode(":",$value);
$timestamp_part = explode(" ",$output_1[2]);
$timestamp = $output_1[0].":".$output_1[1].":".$timestamp_part[0];

$log = $timestamp_part[1]." ".$timestamp_part[2]." ".$output_1[3]." ".$output_1[4];
if(sizeof($output_1) == 6){
$log = $log." ".$output_1[5];
}
$result["timestamp"]=$timestamp;
$result["log"]=$log;

$output[]=$result;
```

```
}

echo json_encode($output);
?>
```

unban.php

```
<?php
ini_set('display_errors', 1);
include 'notify.php';
$pid = $argv[1];
$message = $pid." was unbanned.";
notify($message,$pid);

?>
```

ban.php

```
<?php
ini_set('display_errors', 1);
include 'notify.php';
$pid = $argv[1];
$message = $pid." was banned.";
notify($message,$pid);
```

```
?>
```

auth.php

```
<?php
include 'MCrypt.php';
include 'db_config.php';
ini_set('display_errors', 1);
$con= mysql_connect($host,$username,$password);
if(!$con)
{
die('couldnot connect'.mysql_error());
}
mysql_select_db("results1",$con);

function validateAuth(){
if (! pc_validate($_SERVER['PHP_AUTH_USER'], $_SERVER['PHP_AUTH_PW'])) {
    header('WWW-Authenticate: Basic realm="My Website"');
    header('HTTP/1.0 401 Unauthorized');
    echo "You need to enter a valid username and password.";
    exit;
}
}
```

```
function pc_validate($user,$pass) {
        $mcrypt = new MCrypt();
        $username =$mcrypt->decrypt($user);
        $password = $mcrypt->decrypt($pass);
        $pass_md5 = md5($password);

        $query = "select * from users where username = '$username' and password =
'$pass_md5'";

        $result=mysql_query($query);
        $count=mysql_num_rows($result);

        if($count==1)
        {
                return true;
        }
        else
        {
         return false;
        }
}
```

```
if(!empty($_GET['user']))
login();

function login(){
	if(pc_validate($_GET['user'],$_GET['pass'])){
		$resultData['code'] = 200;
	}
	else{
		$resultData['code'] = 401;
	}

	echo json_encode($resultData);
}
?>
```

dbconfig.php

```
<?php
$host = "localhost:3306";
$username = "root";
$password = "Manhattan@123"
?>
```

Mcrypt.php

```
<?php
```

```
class MCrypt
{
    private $iv = 'fedcba9876543210'; #Same as in JAVA
    private $key = '0123456789abcdef'; #Same as in JAVA


    function __construct()
    {
    }

    /**
     * @param string $str
     * @param bool $isBinary whether to encrypt as binary or not. Default is: false
     * @return string Encrypted data
     */
    function encrypt($str, $isBinary = false)
    {
        $iv = $this->iv;
        $str = $isBinary ? $str : utf8_decode($str);

        $td = mcrypt_module_open('rijndael-128', ' ', 'cbc', $iv);
```

```
    mcrypt_generic_init($td, $this->key, $iv);
    $encrypted = mcrypt_generic($td, $str);

    mcrypt_generic_deinit($td);
    mcrypt_module_close($td);

    return $isBinary ? $encrypted : bin2hex($encrypted);
}

/**
 * @param string $code
 * @param bool $isBinary whether to decrypt as binary or not. Default is: false
 * @return string Decrypted data
 */
function decrypt($code, $isBinary = false)
{
    $code = $isBinary ? $code : $this->hex2bin($code);
    $iv = $this->iv;

    $td = mcrypt_module_open('rijndael-128', ' ', 'cbc', $iv);
```

```
        mcrypt_generic_init($td, $this->key, $iv);
        $decrypted = mdecrypt_generic($td, $code);

        mcrypt_generic_deinit($td);
        mcrypt_module_close($td);

        return $isBinary ? trim($decrypted) : utf8_encode(trim($decrypted));
    }

    protected function hex2bin($hexdata)
    {
        $bindata = '';

        for ($i = 0; $i < strlen($hexdata); $i += 2) {
            $bindata .= chr(hexdec(substr($hexdata, $i, 2)));
        }

        return $bindata;
    }

}
?>
```

unregister.php

```
<?php
include 'db_config.php';
$id = $_REQUEST['id'];
$con= mysql_connect($host,$username,$password);
if(!$con)
{
die('couldnot connect'.mysql_error());
}
mysql_select_db("results1",$con);
        $queryStr = "DELETE FROM gcmReg WHERE regid='$id';";
$result=mysql_query($queryStr);
mysql_close($con);
?>
```

register.php

```
<?php
include 'db_config.php';
$con= mysql_connect($host,$username,$password);
$id = $_REQUEST['id'];
if(!$con)
{
die('couldnot connect'.mysql_error());
```

```
}
mysql_select_db("results1",$con);
        $queryStr = "insert into gcmReg set regid='$id';";
$result=mysql_query($queryStr);
mysql_close($con);
?>
```