UNIVERSITY OF TORONTO

MUNK SCHOOL OF GLOBAL AFFAIRS

Join the Global Conversation

Appendix A: Technical Analysis

Are the Kids Alright? Digital Risks to Minors from South Korea’s Smart Sheriff Application

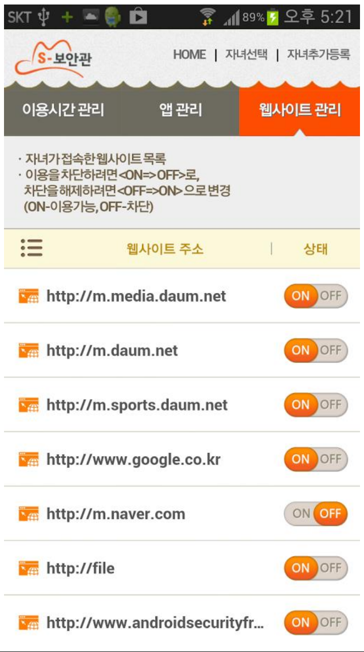
Document Version 1.0
20 September 2015

Smart Sherriff and S-Dream are mobile applications that allow parents to control their children's mobile phone usage and monitor their messages, respectively. The applications were released by the Korean Mobile Internet Business Association (MOIBA),[1] a consortium of mobile telecommunication providers and phone manufacturers, Smart Sheriff was officially launched for Android in June 2012[2] with an iOS version created soon after (the iOS app has not been updated since 2013 and its usability is reported to be limited because of platform restrictions).[3]

Smart Sheriff Functionality

Smart Sheriff allows parents to remotely monitor and administer applications that minors are able to access on their mobile device, and to schedule the times of day that the phone can be used. Descriptions of Smart Sheriff on app stores claim it can filter websites that minors can access. However, these functions have apparently been disabled since May 18, 2015. MOIBA indicated that the reason for disabling this functionality was concern over infringement on children's privacy However, as we note later in issue 1.3, "Disclosure of User Traffic Records in Cleartext," while parents cannot monitor or control their children's Internet access, cleartext user-traffic information continued to be sent to MOIBA servers in a vulnerable manner.

[1] https://www.moiba.or.kr/ [in Korean].
[2] First link here: http://ss.moiba.or.kr/customer/bbs/list.do [in Korean].
[3] http://translate.google.com/translate?hl=en&sl=ko&u=https://itunes.apple.com/us/app/seumateu-boangwan/id689953031%3Fmt%3D8&prev=search.


SKT 89% 오후 5:21

S-보안관

HOME | 자녀선택 | 자녀추가등록

이용시간 관리 | 앱 관리 | 웹사이트 관리

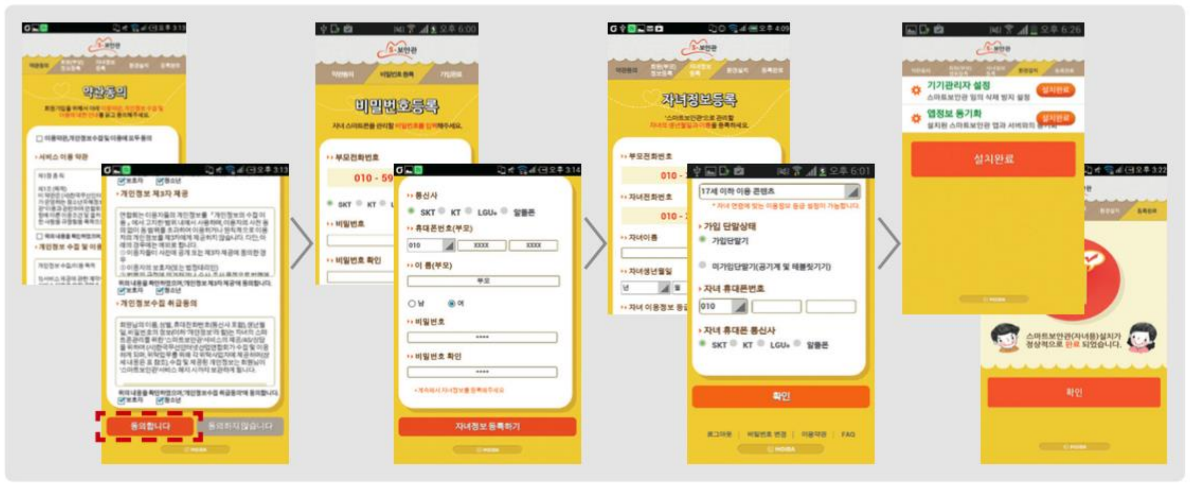
· 자녀가 접속한 웹사이트 목록
· 이용을 차단하려면 <ON=>OFF>로,
차단을 해제하려면 <OFF=>ON>으로 변경
(ON-이용가능, OFF-차단)

웹사이트 주소	상태
http://m.media.daum.net	[✓ ON] OFF
http://m.daum.net	[✓ ON] OFF
http://m.sports.daum.net	[✓ ON] OFF
http://www.google.co.kr	[✓ ON] OFF
http://m.naver.com	ON [✓ OFF]
http://file	[✓ ON] OFF
http://www.androidsecurityfr...	[✓ ON] OFF

Screenshot of removed filtering functions from Google Play Store.

Once installed on the child’s phone, Smart Sheriff requires information from a newly registered user on:

- phone numbers for parent and child;
- the child’s gender and date of birth;
- name of child; and
- PIN code for the account’s administration.


Installation Process from Smart Sheriff Guidebook

After registration, Smart Sheriff routinely transmits information on the usage and configuration of the child’s phone to the back-end server, including:

- manufacturer, model, and operating system version for the device;
- applications installed on the phone and their amount of usage; and
- websites visited.

Additionally, Smart Sheriff requests “device administration” privileges to the phone that allow it to set the device’s security policies for preventing removal of the application.[4]

It appears that the only authorized way to remove the app is through a function on the Web interface for deleting accounts. Activating the removal process from the MOIBA site pushes a command to the child’s phone to unlock access and uninstall Smart Sheriff.

During removal, Smart Sheriff performs a request to a unique API endpoint (//main/deviceRelieve), passing the device identifier as the only parameter. This request appears to be solely for tracking who uninstalls the Smart Sheriff, since the response is empty and the application does not appear to check the result. The Smart Sheriff database is kept in the sandboxed data directory and is reused if the application is reinstalled, but would not be accessible unless the user has root access to the device.

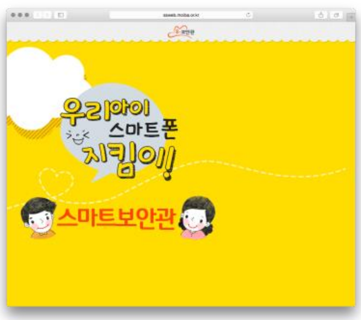
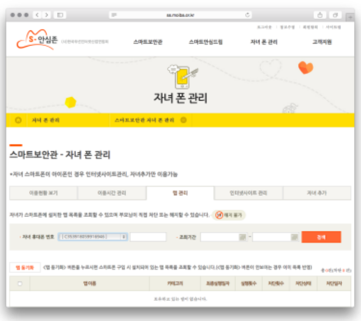
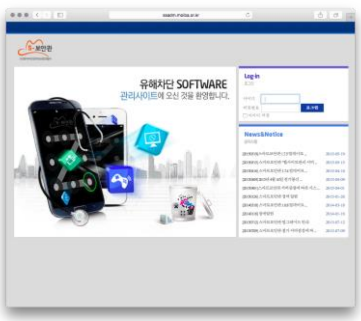
[4] http://developer.android.com/guide/topics/admin/device-admin.html

Smart Sheriff User Interface

Parents can control the usage of applications on a child's device through two interfaces, the application itself and a website hosted by MOIBA.[5] Both require the parent's phone number and the PIN code set during registration before the parent is granted administration privileges. All user interactions and service communications, whether through the mobile application or the desktop site, occur within a Web session to the same server.[6]

Smart Sheriff itself is simple in functionality. On both interfaces, the parent can review the information collected from the child's device and control what applications can be used. The application also enables parents to completely disable use of the device for certain times of the day over the week. The child's interaction with the application is limited to the warning messages that are triggered when s/he attempts to use prohibited applications or access the phone during restricted times.

An additional internal MOIBA administration website is exposed to the public but it requires username and password credentials for access.

		 유해차단 SOFTWARE 관리사이트에 오신 것을 환영합니다. Log in News&Notice
Application Interface	Web Interface	Internal Administration

Technical Methodology

Installation of Smart Sheriff is restricted in the Google Play Store to users connecting from South Korea. A genuine copy of the application package (APK) for version 1.7.5, released 15 May

[5] Since disclosure, MOIBA has split Smart Sheriff into two versions, a "lite" child application and a parental administration application.
[6] As noted in Lack of Transport Security in Communications with Smart Sheriff Infrastructure, this site lacked encryption for authentication and communications and exposed the application's users to compromise. Additionally, MOIBA's internal administration site (ssadm.moiba.or.kr) has not appeared to use HTTPS for login or administration.

2015 on the Play Store, was found online through alternative channels. Subsequent versions analyzed were obtained directly through the Google Play store.
With access to the APK, we were able to decompile the Java DEX package for evaluation and to install the application locally to capture traffic between Smart Sheriff and its remote APIs.

Versions of Smart Sheriff that were investigated during the course of this report

Version	**SHA-1 Sum**	**Comment**
1.7.5	734b98b40a9bcba5f4182aa006c9d9f2b0d880db	Original version investigated.
1.7.7	72abf9571aff3a92de8598477891a569fae9f553	Latest version released - some fixes included.

As we note later, Smart Sheriff fails to protect user information in transit or stored on the device, which provides easier access to service information for documenting systemic flaws. Through access to the decompiled bytecode and user traffic, we were able to catalogue the calls made to Smart Sheriff in the course of administration and use of the application.

Smart Sheriff only supports users with a South Korean phone number and does not provide service to individuals outside of the country. While S-Dream appears to require a valid phone number available to the application through device APIs,[7] Smart Sheriff requests that the user enter a phone number and then blindly trusts the registering user's information without secondary validation of ownership. Therefore, we were able to create an account associated with an accepted phone number, with the consent of the phone number owner, outside of South Korea

Although a Smart Sheriff application for iOS appears in the iTunes store, it is a version (v1.0.0) that has not been updated since 19 September 2013. Given the lack of attention paid to this version, and the clear failures of the recently maintained Android application, we did not attempt to evaluate flaws in the iOS version. However, screenshots from the iTunes store indicate that the iOS version was built through a similar approach and would therefore have the same issues.

Notifications and Responsible Disclosure

On 3 August 2015, Citizen Lab notified MOIBA of the issues identified in the two security audits. Following established standards for vulnerability disclosure, we set a publication deadline for a minimum of 45 days after our initial disclosure of vulnerabilities to the vendor.[8]

On 5 August a MOIBA representative replied and provided an initial timeline for addressing fifteen of the vulnerabilities. On 6 August MOIBA released an updated version of the application

[7] This function can also be bypassed since no additional verification of the number occurs.
[8] See, for example, "Vulnerability Disclosure Policy," http://www.cert.org/vulnerability-analysis/vul-disclosure.cfm.

(v1.7.6) that supported HTTPS.[9] An additional update (v1.7.7) released on 25 August claimed to address additional vulnerabilities. [10]

By the most recent timeline provided to the Citizen Lab by MOIBA, 20 September 2015 patches should be in place for twenty of the issues identified, with sixteen published. Two further patches are scheduled shortly after the publication of this report. However, we has not fully verified whether all patches have been implemented, and MOIBA has not fully of the manner in which the vulnerabilities were addressed.

On 4 September MOIBA was notified of this report's intended publication date and was sent a copy for review to ensure that no personally identifying information was inadvertently disclosed. As of the date of publication, we have not received any further correspondence from MOIBA.

Smart Sherriff: Tracked Issues

The following sections list vulnerabilities and implementation issues identified during our audit of Smart Sheriff. The issues are grouped by themes rather than degree of severity and impact. This assessment is the merged product of two independent security audits: one by researchers who collaborated at the 2015 Citizen Lab Summer Institute (CLSI) held at the Munk School of Global Affairs, University of Toronto[11] and another by the auditing firm Cure53.[12] Cure53's security audit was performed under an ongoing contract from the Open Technology Fund.[13] Where Cure53 identified the issue, we provide a reference to their security audit report.

Lack of Transport Security in Communications with Smart Sheriff Infrastructure

[9] https://ss.moiba.or.kr/customer/bbs/info.do?BBS_BOARD_CODE=Notice&BBS_POST_CODE=2949&pop=Y&NOWNUM=3 [in Korean].

[10] https://ss.moiba.or.kr/customer/bbs/info.do?BBS_BOARD_CODE=Notice&BBS_POST_CODE=2984&pop=Y&NOWNUM=1 [in Korean].

[11] The Citizen Lab at the Munk School of Global Affairs, University of Toronto, is an interdisciplinary laboratory that explores the intersection of information and communications technology (ICT), global security, and human rights. For more information on the Citizen Lab see https://citizenlab.org. The Citizen Lab Summer Institute is an annual research workshop organized by the Citizen Lab (see https://citizenlab.org/summerinstitute/index.html). In this document we identify the individuals who collaborated on the security audit of Smart Sheriff at CLSI 2015 as "CLSI participants."

[12] Cure53 is a Berlin-based security company specializing in thorough and manual penetration tests and code audits covering Web applications, cryptographic implementations, and other soft- and hardware. For more information on Cure53 see https://cure53.de

[13] For more information on the Open Technology Fund see https://www.opentechfund.org

Issue 1.1: No Transport Security in Smart Sheriff Communications (Severity: High)

Smart Sheriff maintains and updates itself through interactions with an API located at api.moiba.or.kr through JSON-encoded POST requests over a keep-alive HTTP connection (primarily to the endpoint */MessageRequest*).[14] While this server supports TLS (see SSL Misconfiguration on MOIBA Resources) and appears to properly respond to API calls over HTTPS, currently all API calls are hardcoded in the application to make requests in the clear. The only encrypted communications between the application and the remote Smart Sheriff service are those that use Google's Cloud Messaging (GCM) infrastructure. GCM is used as a push messaging service to coordinate changes made in the application's administrative interfaces (see Remote Code Execution via Man-in-the-Middle in the Application WebView) and the local application's database. Outside of GCM, we could not identify a single API request using transport encryption, and a number of searches on the decompiled application yielded no results for HTTPS addresses.

As a result of this failure to encrypt data, the parameters necessary for authentication, registration, and coordination with the Smart Sheriff services are transmitted to a remote API in cleartext exposed to anyone listening on the network. These contain users' Personally Identifiable Information (PII), including the children's and parents' names, dates of birth, mobile device hardware and system information, gender, and telephone numbers. Smart Sheriff also subsequently transmits in the clear the unique account identifier and passcode set by the parent to limit access, as well as session cookies, which are used for authenticating the client to the service. This behaviour is especially concerning in a mobile application, given that mobile devices generally favor WiFi over mobile data usage, and are therefore exposed to eavesdropping by other users of wireless networks.

While the authentication flaws enumerated later in this report present more expedient access to accounts, failure to encrypt communications provides for the easy interception and impersonation of communications between Smart Sheriff and local network users. These transactions provide an intermediary with the ability to control the device by forging commands to restrict usage or to disable the device. Through a privileged network position, such as directing the device to a false version of the API through DNS, an attacker could disrupt the use of the device, falsify application queries on the sensitivity of content, or curtail parental controls.

Smart Sheriff offers a Web interface for the administration of accounts in a normal browser environment on a site hosted on the same server as the API (at an alternative domain, ss.moiba.or.kr). This site also lacks encryption for authentication and communications, and further exposes the application's users to compromise. Additionally, MOIBA's internal administration site (ssadm.moiba.or.kr) does not appear to use HTTPS for login or administration.

[14] Cure53, "Pentest-Report Smart Sheriff 07.2015," p. 4 and CLSI participants.

Issue 1.2: Remote Code Execution via Man-in-the-Middle in the Application Interface —Insecure WebView Use (Severity: High)

Smart Sheriff provides the main user interface for administering the application through an Android WebView that communicates with a remote Web server (ssweb.moiba.or.kr).[15] A simple full-text search on the decompiled application indicates that the lack of transport encryption between the application and remote server could allow an attacker to get control over a phone running Smart Sheriff because of its insecure usage of Android's JavaScript Interfaces for WebViews.

Affected Code

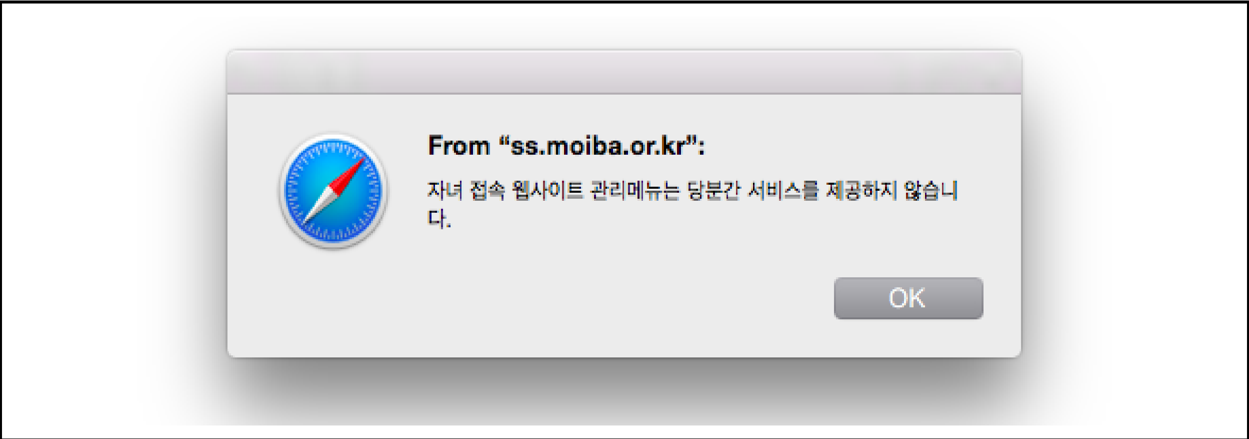
```
Object obj1 = “http://ssweb.moiba.or.kr/pushAlarm”;

_L6:

  WebView webview = (WebView)findViewById(0x7f070000);

  webview.setWebViewClient(new l(this));

  webview.getSettings().setJavaScriptEnabled(true);

  webview.getSettings().setSavePassword(false);

  webview.getSettings().setSaveFormData(false);

  webview.setWebChromeClient(new kr.co.wigsys.sheriff.ui.f(this));

  webview.postUrl(((String) (obj1)), ((String) (obj)).getBytes());
```

Given that all network traffic between the application and the API uses unencrypted HTTP, a man-in-the-middle attack can be easily executed by any attacker who manages to lure a victim into a malicious WiFi network or otherwise has the ability to intercept the user's traffic. Consequently, this allows an attacker to insert arbitrary code into the session and can therefore act as a simple remote code execution vector.

Issue 1.3: Disclosure of User Traffic Records in Cleartext (Severity: High)

[15] Cure53, “Pentest-Report Smart Sheriff 07.2015,” p. 3.

Smart Sheriff ships with the ability to monitor and filter access to Web content, although use of this functionality is not currently available for parents.[16] Despite this feature not being operational, the application still sends records of all Web traffic from the child's device to the Smart Sheriff service. Rather than tunnel traffic to an intermediary for traffic inspection, Smart Sheriff establishes a monitoring service on Android to read the browser history as it is recorded to report and match it against a block list. For every page accessed, an API request is made to the Smart Sheriff API containing the requested domain, page, and URL parameters regardless of whether this Web request was performed to an HTTPS site.



From "ss.moiba.or.kr":

자녀 접속 웹사이트 관리메뉴는 당분간 서비스를 제공하지 않습니다.

Notification on Smart Sheriff's Web interface that the filtering and monitor service is not currently available.

In response to these calls, Smart Sheriff returns a score for each site, determining whether the site should be filtered. If the site is deemed offensive, then the user is forwarded to a blocked site page. While this query includes a property "`ENCRYPT_URL`," which is an obfuscated copy of the requested URL, it concurrently also sends the same URL in clear text. Although some information would already be available to intercepting parties, Smart Sheriff circumvents part of the protective features of HTTPS, thus undermining the security of third-party websites the user has visited by simultaneously exposing the traffic to the network and sending a complete browsing record to MOIBA.

[16] CLSI participants.

Host	`api.moiba.or.kr`
Resource	`POST /MessageRequest`
Sent	{
	“DIRECTORY”:”*Page Requested*”,
	“PORT”:”80”,
	“ENCRYPT_URL”:”*Obfuscated URL*”,
	“MOBILE”:”*Device Identifier*”,
	“action”:”CLT_BLCK_CHKURLBLOCKINFO”,
	“DEVICE_ID”:”*Device Identifier*”,
	“URL”:”*Domain*”,
	“PARAMETER”:”*Passed URL Parameters*”
	}
Returns	{
Urlencoded	“BLCK_ACT_DIVN”:”1”,
	“B_PATH”:”/”,
	“ENCRYPT_URL”:”*Encrypted URL of Request*”,
	“B_FILE”:”/”,
	“B_PORT”:”80”,
	“BAD_BLCK_GRADE”:”*Site Grade*”
	}

Parental Controls Can Be Bypassed or Removed

Issue 2.1: Possible Filter or Schedule Restriction Bypass via Unsafe URL Check (Severity: Low)

Smart Sheriff uses the WebView method shouldOverrideUrlLoading() to determine whether a URL should be loaded or blocked.[17] This method is implemented in a vulnerable way because it checks the URL string for certain values using the string method contains(), which is a simple search for the presence of the string in the address.

Affected Code (decompiled source)

```
function shouldOverrideUrlLoading:

s.startsWith("market://") || s.startsWith("tel:") ||
s.startsWith("http") && ! s.contains("ssweb.moiba.or.kr")
```

To prevent parent-set restrictions from disabling access to the application interface, this method ensures that "ssweb.moiba.co.kr" is not blocked by any filtering rules. This means that *any* URL, even if blacklisted, can be requested, as long as the string "ssweb.moiba.co.kr" is attached to the address (e.g., "*blockedsite.com/*?ssweb.moiba.co.kr," which would not be likely to interfere with the usability of the site requested). Then, the contains() method call will return a *true* and the URL will be considered whitelisted.

[17] Cure53, "Pentest-Report Smart Sheriff 07.2015," p. 4.

Issue 2.2: Smart Sheriff API Discloses Parent Password (Severity: High)

If an attacker is able to guess, enumerate, or intercept the device identifier of a phone with Smart Sheriff installed, they can then make API queries on its behalf, granting control over the account and providing for the disclosure of user information (for further details see Insufficient Protection for Account Access and Failures to Authenticate API Queries).[18] Among these disclosures, queries can be made to the Smart Sheriff API that offers account and password retrieval to disclose the PIN code associated with the account.

Host	api.moiba.or.kr
Resource	POST /MessageRequest
Sent	request={“action”:”CLT_MBR_GETCLIENTMEMBERINFO”,”MOBILE”:”*Device Identifier*”}
Example	request={“action”:”CLT_MBR_GETCLIENTMEMBERINFO”,”MOBILE”:”]5Z\\WSVAB5]”}

Returns
 	
Urlencoded
	{
	“CHILD_GRADE_TYPE”:”*Child Grade*”,
	“CHILD_BIR_YMD”:”*Date of Birth*”,
	“MEMBER_YN”:”*Membership Status*”,
	“CHILD_BLCK_GRADE”:”*Level of Blocking*”,
	“PASSWORD”:”*Parental Pin (Encoded)*”,
	“PARENT_MOBILE”:”*Parental Phone Number(Encoded)*”,
	“REGISTRATION_ID”:”*Registration ID*”,
	“DIVN”:”*(Parent or Child)*”
	}

The response consists of URL-encoded JSON data containing a “PASSWORD” and “PARENT_MOBILE” field. The interesting data are “encrypted” with a simple XOR obfuscation mechanism and can be extracted from the Android app. (See further details on this

[18] Cure53, “Pentest-Report Smart Sheriff 07.2015,” p. 7 and CLSI participants.

obfuscation mechanism in Insufficient Protection for Account Access and Failures to Authenticate API Queries).

The application can be removed by entering the PIN code assigned during the initial registration process. This PIN appears to be constrained to four characters, which is substantially limited compared to industry practice. As we note in Smart Sheriff Does Not Appear to Monitor or Rate Limit Sensitive API Requests, a brute-force attack would be unlikely to be prevented by the API. However, while a PIN length of four characters is insufficient to protect any information of a sensitive nature, the insecure API request takes the burden of brute forcing 10,000 numbers off the attacker and exposes the passwords directly.

During removal, Smart Sheriff performs a request to a unique API endpoint (*//main/deviceRelieve*), passing the device identifier as the only parameter. This request appears to be solely for the purpose of tracking who uninstalls Smart Sheriff, since the response is empty and the application does not appear to check the result. The Smart Sheriff database is kept in the sandboxed data directory and is reused if the application is reinstalled, but would not be accessible unless the user has root.

Insufficient Protection for Account Access and Failures to Authenticate API Queries

Issue 3.1: Identification to Smart Sheriff API Is Based on Predictable Identifiers (Severity: High)

The primary mechanism for identification across Smart Sheriff APIs is a string variable named "`MOBILE`," "`MOBIL E_NUMVAL`," "`PARENT_MOBILE`," or "`CHILD_MOBILE`," which we refer to as a *device identifier*.[19] The device identifier is primarily derived from the phone number associated with the device the application is installed on or, if the phone number is not available, a unique hardware identifier such as the phone's IMSI.

When used for authentication, this value is XOR obfuscated. The key for this operation can either be easily reverse-engineered or extracted from the decompiled sources of the app, allowing an attacker to decrypt any of the protected data or target specific users. Since the number of phone numbers in South Korea is finite, predictable, and relatively limited, an attacker could potentially enumerate over all assigned South Korean phone numbers to determine whether an account is associated with any given number. Alternatively, in the case of targeted intrusion, the attacker can in most cases simply determine an individual's device identifier from their phone number. Once the device identifier is obtained, takeover of the account is trivial.

String "Crypto" (XOR) in Python

[19] Cure53, "Pentest-Report Smart Sheriff 07.2015," p. 6 and CLSI participants.

```
import sys

SS_XOR_KEY =
bytearray([109,0,111,105,98,97,103,116,119,0,105,103,115,121,115,116,101,0,
109,115,102,105,103,104,116,0,105,110,103,104,104,104,107,0,107,107,107,111
,107])

def xor_strings(s,t): return ““.join(chr(ord(a)^b) for a,b in zip(s,t))

sys.stdout.write(xor_strings(sys.argv[1], SS_XOR_KEY))
```

The key used to obfuscate the device identifier is represented by the string "‘`moibagtwigsystemsfightinghhhkkkkok`." Even if the key is not known to an attacker, the cryptography in use by the app can be fully bypassed with the use of a simple and well-known plaintext attack. All the XOR-encoded information can be easily decoded and allow any attacker to access sensitive information or perform API calls to leak additional data.

Issue 3.2: API Queries Do Not Require Authentication (Severity: High)

Smart Sheriff does not require cookies, such as a session ID, or any form of second-factor authentication to perform a broad number of operations on behalf of the user through its API.[20] To exacerbate this, other parameters that could be used to validate the user to Smart Sheriff, such as children's names, can be omitted or set with false data for most API calls. Parameters such as *DEVICE_ID* appear to be nothing more than duplicating the obfuscated device identifier discussed earlier.

The Smart Sheriff application interface is a WebView wrapper for a site located on the public Internet. Authentication between the application and its interface is based on the obfuscated device identifier and a session identifier is maintained through a simple cookie. While interception of this cleartext transaction should be sufficient for hijacking, more straightforward forgery can occur by predicting the device identifier to create a new session cookie to gain access to the same application interface as the user of the account would see on their mobile device.

[20] Cure53 and CLSI participants. "Pentest-Report Smart Sheriff 07.2015," p. 6 contains further examples of the user data that are disclosed by unauthenticated queries.

Action	Authentication from the Phone to the Application Interface
Host	`ssweb.moiba.or.kr`
Resource	`POST /main/login`
Requires	`MOBILE=`*`Device Identifier`*
Sent	Valid Session Cookie

Issue 3.3: Arbitrary Users Can Claim Child Accounts to Monitor Activities and Modify Protection Settings (Severity: High)

Smart Sheriff does not ensure the number claimed by a user in the process of registration is actually associated with the device or under the user's control.[21]

Typically, mobile services that authenticate based on a phone number require a second factor prior to registration or login, such as providing a code sent by SMS to the requested phone number. Registration for Smart Sheriff can occur during the first use of the application and from the Web interface. The on-device registration API provides for the most straightforward means for a third party to associate a number with an account, since it does not appear to be necessary to be logged in or in possession of a valid number to claim the number to a parent. Through registering a child account to a parent, the child's activity can be monitored and the services offered by Smart Sheriff can be curtailed, including deleting the application from the child's phone through a page provided in the Web interface.

Action	Register Account from Device's API	
Host	`ssweb.moiba.or.kr`	
Resource	`POST /member/memberRegisterProc`	

Sent
 	
Urlencoded
	`ACCOUNT_GBN=login&`
	
	`IN_DEVICE_ID=&`
	
	`SMRT_PHN_OS=&`
		`PRVT_INFO_COL_AGREE_YN=Y&`
		
		`FLAG_YN_C=Y&`
		
		`FLAG_YN_V=Y&`

[21] CLSI participants.

	MOIBILE_DEVICE_VENDER=&
	
	MOIBILE_DEVICE_MODEL=&
	
	MOIBILE_ANDROID_VER=&
	
	MOIBILE_ANDROID_RELEASE=&
	
	REGISTRATION_ID=&
	
	CHILD_MOBILE=*Child Number*&
	
	OS_TYPE=&
	
	APP_TYPE=&
	
	PARENT_TELECOM_CD=&
	
	PARENT_NAME=&
	
	PARENT_SEX=&
	
	PARENT_MOBILE=*Parent Number*&
	
	SERVICE_USE_AGREE_YN=Y&
	
	PRVT_INFO_USE_AGREE_YN=Y&
	
	PRVT_INFO_OFFER_AGREE_YN=Y&
		PACH_GBN=&
		
		TELECOM_CD=&
		
		NAME=&
		
		BIR_YMD1=&
		
		BIR_YMD2=&
		
		BIR_YMD3=&
		
		SEX=&
		
		reqnum=&
		
		CHILD_NAME=&
		
		CHILD_SEX=&
		
		BIR_Y=&
		
		BIR_M=&
		
		BIR_D=&
		
		CHILD_BLCK_GRADE=

Action	Register Child Phone from the Web Interface’s API
Host	`ss.moiba.or.kr`
Resource	`POST /childPhone/insertAddChild.do`
Requires	Valid Session Cookie for Real Account

Sent
 	
Urlencoded
	CHILD_MOBILE=*Child Number*&
	
	CHILD_NAME=&
	
	CHILD_NCKNM=&

	BIR_YMD=&
	CHILD_SEX=&
	TEL_CD=&
	CHILD_MOBILE1=*Child Number[1-3]*&
	CHILD_MOBILE2=*Child Number [4-8]*&
	CHILD_MOBILE3=*Child Number [8-12]*

Issue 3.4: Smart Sheriff Does Not Appear to Monitor or Rate Limit Sensitive API Requests (Severity: Medium)

To understand the popularity of Smart Sheriff's services, we attempted to enumerate potentially valid phone numbers through the application's API to determine whether an account was associated with the number.[22] After enumeration of nearly one million potential phone numbers, scanning stopped to limit further data collection. At no point were these queries subject to restriction from the Smart Sheriff service, despite how these requests should have been highly noticeable.[23] This means that brute-force attempts on passwords and numbers would be feasible against the application, even if great protections for account access were put into place.

Issue 3.5: Denial and Comprise of Smart Sheriff Service Based on Claims of Mobile Numbers (Severity: High)

Once a device identifier has been claimed by a "parent," it does not appear to be immediately possible to regain access to that number through the Smart Sheriff application or Web interface. The service trusts claims made on numbers without additional validation, which creates the opportunity for a malicious actor to deny access to the service through preemptively associating phone numbers to non-existent accounts. (For methods see Arbitrary Users Can Claim Child Accounts to Monitor Activities and Modify Protection Settings).

While the Web interface's enrolment page checks whether a number is taken before attempting to register, the availability-check and registration actions are performed by two independent API calls. If the Web interface's registration API is called directly for an already-claimed number, it blindly creates a new account association, whether or not the already-claimed number has an existing account. This allows for the additional association of an account to a parental

[22] Cure53 and CLSI participants.
[23] These data were subsequently destroyed because of concerns over the sensitivity of user information and the ease of collection.

administrator without validation by the child unilaterally, providing a further vector of compromise.

Combined with the predictable identification schema of Smart Sheriff and the service's failure to rate limit requests on account service, this vulnerability provides a vector for massively disrupting the service through denying further registrations and compromising accounts to delete the application from users' phones. Thus, the blind trust model could be used as a means to permanently disrupt all functions of the Smart Sheriff service.

Issue 3.6: Smart Sheriff API Leaks Account Information (Severity: Medium)

As noted in Smart Sheriff API Discloses Parental Password, the Smart Sheriff API offers an action labeled `CLT_MBR_GETCLIENTMEMBERINFO`, which takes an obfuscated device identifier and exposes personal and administrative information.[24] These APIs disclose a substantial amount of private data on the user based solely on the device identifier, for example, the parent's phone number, associated children, and dates of birth. Since the device identifier is easily obtainable and predictable, any API-provided information is therefore subject to simple disclosure and enumeration. Given the simplicity of this approach, we do not attempt to detail every piece of data leaked by various APIs. We expect that any information collected by Smart Sheriff can be obtained by a third party on the basis of the device identifier.

[24] Cure53, "Pentest-Report Smart Sheriff 07.2015," p. 8 and 11 and CLSI participants.

Issue 3.7: Improper Authentication of API Requests Allow for False Incident Reports (Severity: Low)

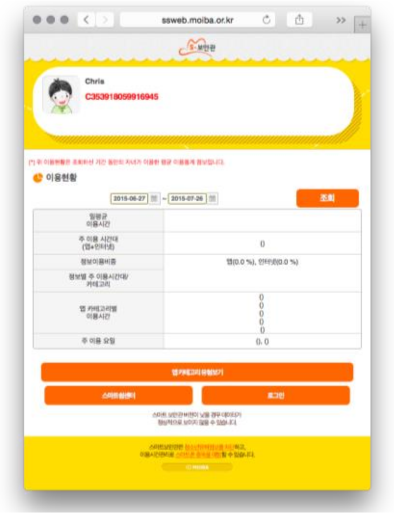
When a child installs (and uses) additional applications or visits Web pages, these actions are reported to the parent through their copy of the application or the Web interface. [25] Since API requests are not properly authenticated, a third party can easily forge reports of prohibited activity once they have access to the device identifier. A parent attempting to discern whether the reported activity was actually produced by the child's behaviour would have no means of knowing whether the claim was valid.

Issue 3.8: Smart Sheriff Application Interface Leaks Account Information Without Authentication

Smart Sheriff does not consistently authenticate requests and API calls using the obfuscated device identifier or the derivable session cookie.[26] In numerous instances, information is retrievable simply based on passing a phone number to a specific URL. In the case of the remotely hosted application interface, the *selfUseStatus* page requires only the plain device identifier (the unencrypted phone number or hardware identifier used to register the child's account) and then discloses the name, age, and usage statistics of the associated user.

Address	`http://ssweb.moiba.or.kr/main/selfUseStatus?MOBILE_NUMVAL=`*`Child Phone Number`*

[25] Cure53, "Pentest-Report Smart Sheriff 07.2015," p. 11 and CLSI participants.
[26] Cure53, "Pentest-Report Smart Sheriff 07.2015," p. 10.



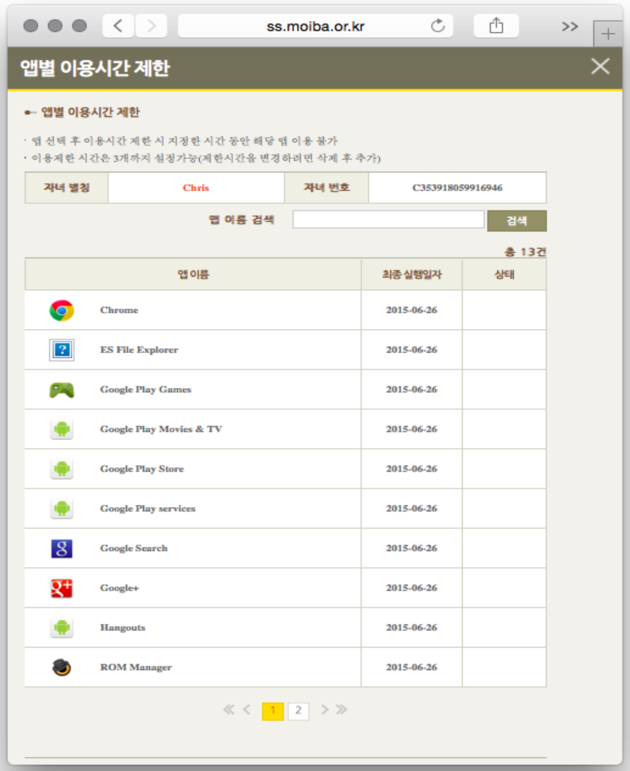
일평균 이용시간	
주 이용 시간대 (앱+인터넷)	0
정보이용비중	앱(0.0 %), 인터넷(0.0 %)
정보별 주 이용시간대/ 카테고리	
앱 카테고리별 이용시간	0 0 0 0 0
주 이용 요일	0, 0

Issue 3.9: Smart Sheriff Web Interface Allows Device Control and Discloses Personal Information Without Authentication (Severity: High)

Similar to the application interface, multiple information and administration pages for the Web interface do not consistently authenticate credentials and rely solely on the device identifier passed as a URL parameter, even where they may also require the child's name.[27] (The CHILD_NAME parameter also creates issues in Reflected Cross-site Scripting (XSS) in Web Interface Pages). These pages display the user's installed applications and control the schedules for when a child can use their phone. In the latter case, this vulnerability allows any individual with the proper device identifier of a child to be able to completely disable the use of their device.

Action	`Display User's Installed Application`
Address	`http://ss.moiba.or.kr/popup/popupApp.do?DAY=N&CHILD_MOBILE=`*`Child Phone Number`*`&CHILD_NAME=`*`Arbitrary String`*

[27] CLSI participants.



앱별 이용시간 제한

앱별 이용시간 제한

- 앱 선택 후 이용시간 제한 시 지정한 시간 동안 해당 앱 이용 불가
- 이용제한 시간은 3개까지 설정가능(제한시간을 변경하려면 삭제 후 추가)

자녀 별칭	Chris	자녀 번호	C353918059916946

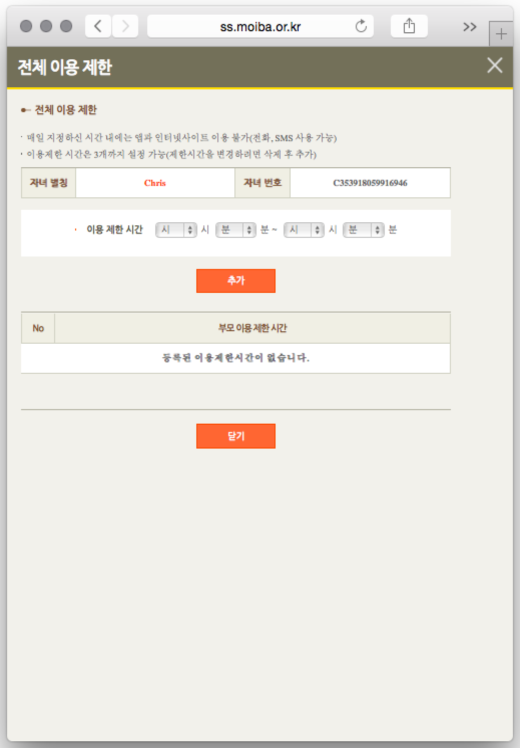
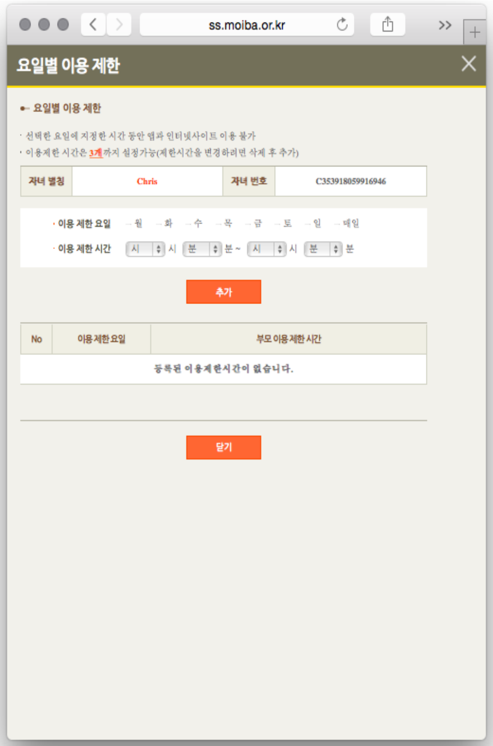
앱 이름 검색 ______ 검색

총 13건

앱 이름	최종 실행일자	상태
Chrome	2015-06-26	
ES File Explorer	2015-06-26	
Google Play Games	2015-06-26	
Google Play Movies & TV	2015-06-26	
Google Play Store	2015-06-26	
Google Play services	2015-06-26	
Google Search	2015-06-26	
Google+	2015-06-26	
Hangouts	2015-06-26	
ROM Manager	2015-06-26	

« < 1 2 > »

Action	Administer Usage Schedule
Address	`http://ss.moiba.or.kr/popup/popupDay.do?DAY=N&CHILD_MOBILE=`*`Child Phone Number`*`&CHILD_NAME=`*`Arbitrary String`*

	`http://ss.moiba.or.kr/popup/popupAll.do?DAY=N&CHILD_MOBILE=`*`Child Phone Number`*`&CHILD_NAME=`*`Arbitrary String`*
	

ss.moiba.or.kr

전체 이용 제한

전체 이용 제한

· 매일 지정하신 시간 내에는 앱과 인터넷사이트 이용 불가(전화, SMS 사용 가능)

· 이용제한 시간은 3개까지 설정 가능(제한시간을 변경하려면 삭제 후 추가)

자녀 별칭	Chris	자녀 번호	C353918059916946

· 이용 제한 시간 시 시 분 분 ~ 시 시 분 분

No	부모 이용 제한 시간
등록된 이용제한시간이 없습니다.	

ss.moiba.or.kr

요일별 이용 제한

요일별 이용 제한

· 선택한 요일에 지정한 시간 동안 앱과 인터넷사이트 이용 불가

· 이용제한 시간은 3개까지 설정가능(제한시간을 변경하려면 삭제 후 추가)

자녀 별칭	Chris	자녀 번호	C353918059916946

· 이용 제한 요일 월 화 수 목 금 토 일 매일

· 이용 제한 시간 시 시 분 분 ~ 시 시 분 분

No	이용제한요일	부모 이용제한 시간
등록된 이용제한시간이 없습니다.		

Issue 3.10: Smart Sheriff Web Interface Allows Account Access and Discloses Personal Information Through Unauthenticated Web Interface API Queries (Severity: High)

The Web interface uses AJAX requests to communicate with back-end APIs for access to information and to administer accounts.[28] In several cases, these APIs appear to blindly accept obtainable parameters, such as the device identifier or phone number, for retrieving information and modifying accounts. Furthermore, the Web interface's queries are often not checked against expectations of what accounts a user is expected to have access to (i.e., that parents should be able to modify the information of their own children only), and may not even check whether any user is logged in. This lack of verification could lead to the mass compromise of user accounts or removal of all Smart Sheriff accounts.

These operations are performed through POST requests of values to endpoints under the `/ajax` URL, primarily identified based on telephone numbers or device identifiers. For example, APIs that control the change of login credentials do not check whether the requester is a parent of that child or the owner of the account. The same is true for account registration and scheduling times

[28] CLSI participants.

when children can use the mobile device. The Web interface appears to pass a large number of parameters to each API method, whether they are used or not. The account deletion endpoint `ajax/ajaxDeleteAgree.do`, which triggers the child's installation of an application to unlock administrative control and delete itself from the phone, mostly passes the same variables.

While our audit examined only a limited number of queries, it is clear that an attacker who has enumerated the phone numbers could potentially change the PINs of parental accounts, remotely disable devices, and even disclose personal information for all Smart Sheriff users.

Action	Changes User Information, Including Password.
Host	`ss.moiba.or.kr`
Resource	`POST /ajax/ajaxUpdateMember.do`

Sent
 	
Urlencoded
	`PARENT_MOBILE=`*`Parent Phone Number`*`&`
	
	`CHILD_MOBILE=`*`Child Phone Number`*`&`
	
	`CHILD_BIR_YMD=`*`Child Date of Birth`*`&`
	
	`DELETE_LIST=`*`Child Numbers to Delete from Parent's Account`*`&`
	
	`NAME=`*`Name`*`&`
	
	`PASS=`*`Parental Pincode`*`&`
	
	`PASS_CNFM=`*`Parental Pincode`*`&`
	
	`EMAIL=`*`Email Address`*`&`
	
	`CHILD_NAME=`*`Child Name`*`&`
	
	`CHILD_NCKNM=`*`Child Name`*`&`
	
	`BIR_YMD=`*`Date of Birth`*`&`
	
	`CHILD_SEX=`*`Child Gender`*`&`
	
	`CHILD_BLCK_GRADE=`*`Child Blocking Level`*

Returns	*`HTTP 201`*

Inadequate Protection for Locally Stored Data

Issue 4.1: Unsafe Mobile App Data Storage on SD Card (Severity: Low)

Smart Sheriff defeats the built-in protections provided by the Android operating system by saving sensitive data in plaintext on the SD card, which other applications can write to or read from.[29]

Issue 4.2: Lack of Storage Protections on the Mobile Application (Severity: Low)

Smart Sheriff does not implement any form of cryptographic protection on the mobile internal storage.[30] All data are stored in unencrypted plaintext and can be accessed by anyone with access to the phone.

Issue 4.3: Multiple Insecure Concatenations on Mobile App (Severity: Low)

Smart Sheriff does not follow security best practices and performs a large number of string concatenations for generating SQL queries and intents.[31] In addition to this, there is no apparent input validation preceding these concatenations. Below are examples found during the security audit.

Affected Code	`source/src/kr/co/wigsys/sheriff/b/a.java`

```
obj6 = ((PackageManager) (obj3)).queryIntentActivities(((Intent) (obj4)),
8192);

obj7 = ((PackageManager) (obj5)).queryIntentActivities(((Intent) (obj)),
8192);

kr.co.wigsys.sheriff.d.a.c(kr.co.wigsys.sheriff.d.c.b(), (new

StringBuilder("[] App = [").append(s).append("] stop db update ret_value =
[ ]").toString());
```

[29] Cure53, "Pentest-Report Smart Sheriff 07.2015," p. 8.
[30] Ibid., p. 10.
[31] Ibid., p. 15.

It appears there is no sanitization of queries against the SQLite stored locally on the phone. Since Smart Sheriff synchronizes this information against the remote service, it may be possible to perform a SQL injection again against this database to disclose private information, such as the administrative PIN, or to delete parent-set schedules or limitations. While most changes from the parent are pushed to the device through GCM, some information is still synchronized in the clear. In the latter case, this could provide for a man-in-the-middle attack. Even where GCM is used, it is unlikely that parameters entered into the Web interface are sanitized before being pushed to the device, creating another vector for injection.

Failures to Sanitize Input Data

Issue 5.1: Reflected Cross-site Scripting (XSS) in Application Interface Pages (Severity: Medium)

The member registration form on the application interface fails to sanitize user-input prior to rendering it on the HTML page.[32] This could be leveraged by an attacker to execute JavaScript in the security context of the ssweb.moiba.or.kr domain and provide a means to impersonate application users and interfere with the application.

Action	`curl -s --data ‘OS_TYPE=A&CHILD_MOBILE=<script>alert(1)</script>‘ ‘http://ssweb.moiba.or.kr/member/pmemberRegisterPwdForm’`

Response	`...`
	
	`<td class=“telnum”>`
	
	`<p> <script>alert(1)</script> </p>`
	
	`...`

Issue 5.2: Reflected Cross-site Scripting (XSS) in Web Interface Pages (Severity: Medium)

The Web interface page for installed applications, discussed in the context of user information disclosure at Smart Sheriff Web Interface Allows Account Access and Discloses Personal Information Without Authentication includes parameters that are embedded in the page without sanitization.[33] The CHILD_NAME parameter is decorative — it’s used for passing the child’s

[32] Ibid., p. 8.
[33] CLSI participants.

name to the page but it is not matched against the actual child name for authentication. This can be used by an attacker to launch malicious Javascript using the credentials of a logged-in user.

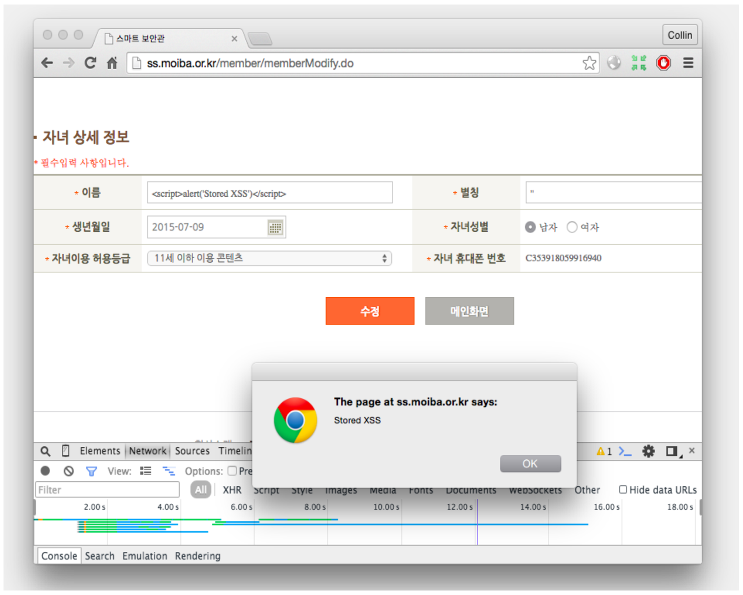
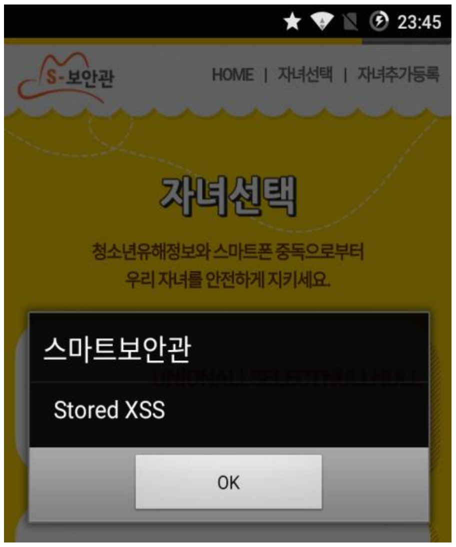
Action	`curl -s --data ‘`
	
	`http://ss.moiba.or.kr/popup/popupApp.do?DAY=N&CHILD_MOBILE=`*`Child Mobile`*`&CHILD_NAME=<script>alert(1)</script>‘`

Response	`...`
	
	`<th>자녀 별칭</th>`
	
	`<td><span id=“childName”><script>alert(1)</script></span></td>`
	
	`...`

Issue 5.3: Stored Cross-site Scripting (XSS) from Web Interface Pages (Severity: Medium)

The Web interface pages and APIs discussed in Smart Sheriff Web Interface Allows Account Access and Discloses Personal Information Through Unauthenticated Web Interface API Queries do not sanitize input during entries of information, or within the back-end API.[34] As a result, malicious Javascript can be inserted into browsing sessions through modification of information, such as children’s name. Failures of the Smart Sheriff service to validate user data changes, or account associations made through the API, increase the risk of this vulnerability being leveraged against end users, especially when they can be done en masse against its entire user base.

[34] CLSI participants.

	
Stored XSS through changing the child name achieved against an example parent account through an unauthenticated Web interface API.	

ss.moiba.or.kr/member/memberModify.do

자녀 상세 정보

* 필수입력 사항입니다.

* 이름	<script>alert('Stored XSS')</script>	* 별칭	"
* 생년월일	2015-07-09	* 자녀성별	[✓ 남자] 여자
* 자녀이용 허용등급	11세 이하 이용 콘텐츠	* 자녀 휴대폰 번호	C353918059916940

The page at ss.moiba.or.kr says:

Stored XSS

청소년유해정보와 스마트폰 중독으로부터
우리 자녀를 안전하게 지키세요.

스마트보안관

Stored XSS

Other Issues and Misconfigurations

Other noteworthy findings did not immediately lead to an exploit but they might aid attackers in achieving malicious compromise in the future. Most of these reported issues are vulnerable code snippets that did not provide an easy way to be called, or that were not explored further because of the numerous issues that could be readily exploited.

Issue 6.1: Smart Sheriff Test-Pages and Other Found Resources Leaks Application Internals (Severity: Low)

The sites and services that Smart Sheriff communicates with contain pointers and URLs exposing debug pages, test applications, and similar resources that should never be published on a production system.[35] These pages deliver significant internal information that could later be used to exfiltrate data, enumerate private infrastructure information, and carry out further attacks.

Information Leakage on ssweb.moiba.or.kr:

- http://ssweb.moiba.or.kr/index_.jsp
- http://ssweb.moiba.or.kr/html/filelist.html

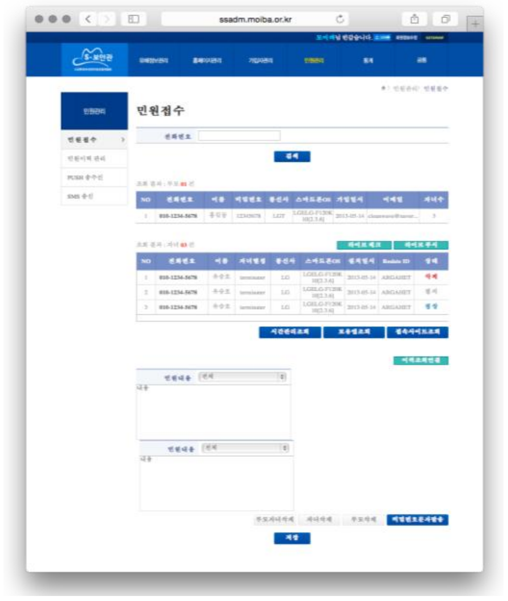
[35] Cure53, “Pentest-Report Smart Sheriff 07.2015,” p. 5.

Information Leakage on ssadm.moiba.or.kr when retrieved without Javascript:

Contents of `curl -i ‘http://ssadm.moiba.or.kr/’`

```
<!DOCTYPE html PUBLIC “-//W3C//DTD HTML 4.01 Transitional//EN”
“http://www.w3.org/TR/html4/loose.dtd”>

<html>

<head>

<meta http-equiv=“Content-Type” content=“text/html; charset=utf-8”>

<script type=“text/javascript”>

//window.onload = function(){

  document.location.href = “/index”;

//}

</script>

</head>

<body>

<ul>

<li><a href=‘/index’>관리자메인</a></li>

<li><a href=‘/subMain’>서브메인메인</a></li>

<li><a href=‘/harm/app/list’>유해정보관리</a>

      <ul>

      <li><a href=“/harm/app/appList”>앱관리</a></li>

      <li><a href=“/harm/site/list”>사이트관리</a></li>
      <li><a href="/harm/accept/acceptList_app">앱/사이트 접수 관리</a></li>

      </ul>

</li>

<li><a href='/member/admin/memberAdm'>가입자관리</a>

<li><a href='/minwon/minwonList'>민원관리</a>

<li><a href='/home/report/list'>홈페이지</a></li>

</ul>

<p>

<a href='/html/filelist.html'>디자인</a><br/><br/>

<a href='/minwon/minwonPushTest'>Push TEST</a><br/><br/>

<a href='/minwon/livePushTest'>Live Push TEST</a><br/>

<a href="minwon/logPushTest">log Push Test</a></br>

</p>

</body>
```

Among the highlighted parts the URL `http://ssadm.moiba.or.kr/html/filelist.html` is particularly interesting. It is available without any authentication and reveals a large amount of further information, including some of URLs listed below.

http://ssadm.moiba.or.kr/html/petition/petition_list.html http://ssadm.moiba.or.kr/html/petition/petition_history.html http://ssadm.moiba.or.kr/html/petition/petition_push.html http://ssadm.moiba.or.kr/html/petition/petition_sms.html http://ssweb.moiba.or.kr/html/filelist.html http://ssweb.moiba.or.kr/html/childmag/childinfo.html http://ssadm.moiba.or.kr/html/homepage/board_list.html# http://ssadm.moiba.or.kr/html/member/appversion.html# http://ssadm.moiba.or.kr/html/member/institution.html http://ssadm.moiba.or.kr/html/harmfulinfo/acceptlist.html# http://ssadm.moiba.or.kr/html/cleanwave_main_in.html#	

ssadm.moiba.or.kr

민원관리

민원접수

NO	전화번호	이름	비밀번호	통신사	스마트폰OS	가입일시	이메일	자녀수
1	010-1234-5678	홍길동	12345678	LGT	LGE LG-F120K 10[2.3.6]	2013-05-14	cleanwave@naver...	3

NO	전화번호	이름	자녀별칭	통신사	스마트폰OS	등록일시	[illegible]	상태
1	010-1234-5678	[illegible]	terminator	LG	LGE LG-F120K 10[2.3.6]	2013-05-14	[illegible]	삭제
2	010-1234-5678	[illegible]	terminator	LG	LGE LG-F120K 10[2.3.6]	2013-05-14	[illegible]	[illegible]
3	010-1234-5678	[illegible]	terminator	LG	LGE LG-F120K 10[2.3.6]	2013-05-14	[illegible]	정상

These addresses appear to demonstrate at least an initial version of the administrative interfaces for the Smart Sheriff application that is used to manage user accounts, restrict access to sites, and display a log of user interactions for all users. Given the authentication issues noted previously, these pages may leak information on vulnerable administrative APIs.

Issue 6.2: SSL Misconfiguration on MOIBA Resources (Severity: Medium)

Smart Sheriff's back-end server (IP address `211.110.12.203`) is within the published IPv4 range `211.110.8.0 - 211.110.15.255 (/21)` assigned to:

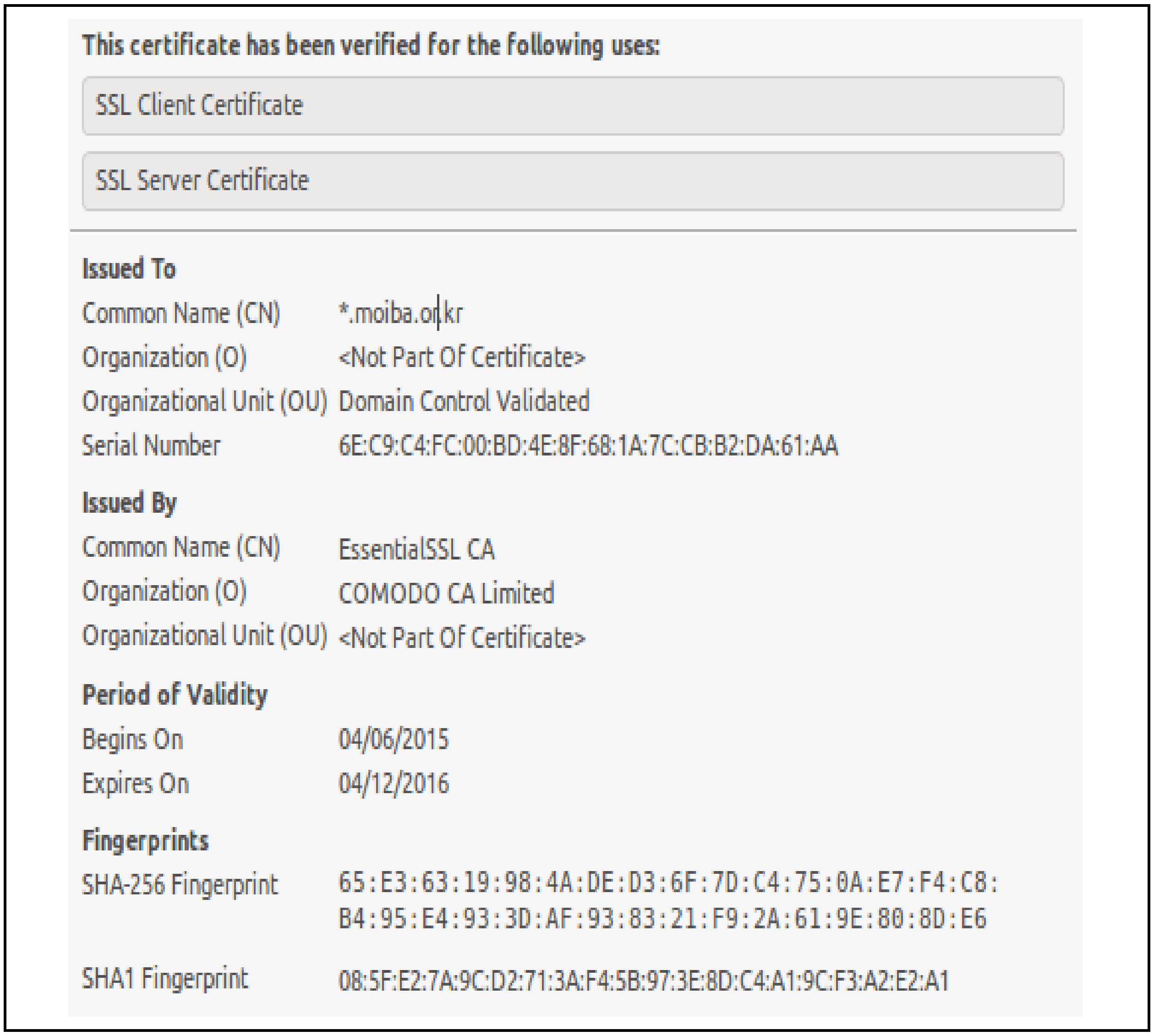
```
Network Name: HANANET-INFRA

Organization Name: SK Broadband Co Ltd

Address: 267, Seoul Jung-gu Toegye-ro

Zip Code: 100-711
```

MOIBA serves a number of APIs, application interfaces, and sites from the same host located at address 211.110.12.203 (at least the domains `api.moiba.or.kr`, `ssweb.moiba.or.kr`, `ssadm.moiba.or.kr`, `ss.moiba.or.kr`, and `sd.moiba.or.kr` are associated with this IP).[36] Host names registered to the moiba.or.kr domain according to DNS records for this range include:

```
211.110.12.196   www, info, m, mis, mt

211.110.12.203   sd, ss

211.110.15.5     ns1

211.110.15.6     ns2
```

The SSL certificate for the server was issued by Comodo using the wildcard `*.moiba.or.kr.`

[36] Ibid., p. 13 and CLSI participants.



This certificate has been verified for the following uses:

SSL Client Certificate

SSL Server Certificate

Issued To

Common Name (CN)	*.moiba.or.kr
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	Domain Control Validated
Serial Number	6E:C9:C4:FC:00:BD:4E:8F:68:1A:7C:CB:B2:DA:61:AA

Issued By

Common Name (CN)	EssentialSSL CA
Organization (O)	COMODO CA Limited
Organizational Unit (OU)	<Not Part Of Certificate>

Period of Validity

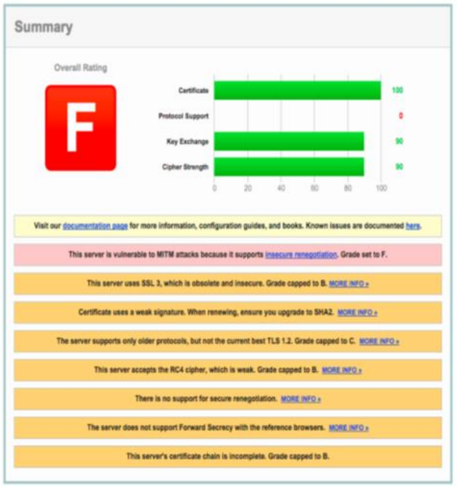
Begins On	04/06/2015
Expires On	04/12/2016

Fingerprints

SHA-256 Fingerprint	65:E3:63:19:98:4A:DE:D3:6F:7D:C4:75:0A:E7:F4:C8: B4:95:E4:93:3D:AF:93:83:21:F9:2A:61:9E:80:8D:E6
SHA1 Fingerprint	08:5F:E2:7A:9C:D2:71:3A:F4:5B:97:3E:8D:C4:A1:9C:F3:A2:E2:A1

While `211.110.12.203` does support SSL and appears to properly handle the same requests over HTTPS, it is not properly configured to support modern TLS ciphersuites and has not discontinued support for features or protocols that are known to allow compromise of communications.[37]

[37] https://www.ssllabs.com/ssltest/analyze.html?d=api.moiba.or.kr&hideResults=on

SSL-Labs Test Report (api.moiba.or.kr)	
● Prone to MiTM attacks via insecure renegotiation ● SSL 3 support ● Weak (SHA1) certificate signature ● The server solely supports old protocols like SSLv3 and TLS 1.0 ● The insecure RC4 cipher is supported ● Secure renegotiation is not supported ● Forward Secrecy is not supported ● The server certificate chain is incomplete	 Summary Overall Rating F Certificate 100 Protocol Support 0 Key Exchange 90 Cipher Strength 90 Visit our documentation page for more information, configuration guides, and books. Known issues are documented here. This server is vulnerable to MITM attacks because it supports insecure renegotiation. Grade set to F. This server uses SSL 3, which is obsolete and insecure. Grade capped to B. MORE INFO » Certificate uses a weak signature. When renewing, ensure you upgrade to SHA2. MORE INFO » The server supports only older protocols, but not the current best TLS 1.2. Grade capped to C. MORE INFO » This server accepts the RC4 cipher, which is weak. Grade capped to B. MORE INFO » There is no support for secure renegotiation. MORE INFO » The server does not support Forward Secrecy with the reference browsers. MORE INFO » This server's certificate chain is incomplete. Grade capped to B.

Issue 6.3: Resources Out of Date, Potentially Vulnerable (Severity: Medium)

The `211.110.12.203` default page publishes the Apache Tomcat version 6.0.29 to the browser (e.g., a 405 page) showing software released in 2010.[38] Version 6.0.29 is known to have thirty-five vulnerabilities including the following:

- **CVE-2014-0230 – CVSS 7.8.** Apache Tomcat 6.x before 6.0.44, 7.x before 7.0.55, and 8.x before 8.0.9 does not properly handle cases where an HTTP response occurs before finishing reading an entire request body. Thus, it allows remote attackers to cause a denial of service (memory consumption) via a series of aborted upload attempts.

- **CVE-2011-3190 – CVSS 7.5.** Certain AJP protocol connector implementations in Apache Tomcat 7.0.0 through 7.0.20, 6.0.0 through 6.0.33, 5.5.0 through 5.5.33, and possibly other versions, allow remote attackers to spoof AJP requests, bypass authentication, and obtain sensitive information by causing the connector to interpret a request body as a new request.

- **CVE-2013-2067 – CVSS 6.8.** The form authentication feature in Apache Tomcat 6.0.21 through 6.0.36 and 7.x before 7.0.33 does not properly handle the relationships between authentication requirements and sessions. This allows remote attackers to inject a request into a session by sending this request during a completion of the login form. This is a variant of a session fixation attack.

[38] Cure53, "Pentest-Report Smart Sheriff 07.2015," p. 14 and CLSI participants.

Also this server runs Apache/2.0.65, released in 2013, which is no longer supported. The current version of Apache is 2.4.16 (two major versions newer than Apache/2.0.65). The server is configured with Tomcat Connector (mod_jk) version 1.2.37, released June 2012. The current version is 1.2.40.

Certain URLs may be processed by an even older / unsupported version (Apache/2.0.59) released in 2007.[39] The older version of Apache could be a forged header inserted by a load balancer, application firewall, or proxy. The Citrix Netscaler load balancer was detected with high probability. This also may introduce vulnerabilities such as a bypass using "HTTP Header Pollution" (CVE-2015-2841).

Apache/2.0.59, if in fact used, is known to have twenty-eight vulnerabilities including:

- **CVE-2011-3192 – CVSS 7.8.** The byte range filter in the Apache HTTP Server 1.3.x, 2.0.x through 2.0.64, and 2.2.x through 2.2.19 allows remote attackers to cause a denial of service (memory and CPU consumption) via a Range header that expresses multiple overlapping ranges. It was exploited in the wild in August 2011 and constitutes a vulnerability different from CVE-2007-0086.

- **CVE-2013-2249 – CVSS 7.5.** mod_session_dbd.c in the mod_session_dbd module in the Apache HTTP Server before 2.4.5 proceeds with save operations for a session without considering the dirty flag and the requirement for a new session ID, which signifies unspecified impact and remote attack vectors.
- **CVE-2009-1890 – CVSS 7.1.** When a reverse proxy is configured, the stream_reqbody_cl function in mod_proxy_http.c in the mod_proxy module in the Apache HTTP Server before 2.3.3 does not properly handle an amount of streamed data that exceeds the Content-Length value. This allows remote attackers to cause a denial of service (CPU consumption) via crafted requests.

- **CVE-2009-1891 – CVSS 7.1.** The mod_deflate module in Apache httpd 2.2.11 and earlier compresses large files until completion, even after the associated network connection is closed. It allows remote attackers to cause a denial of service (CPU consumption).

These out-of-date services put the Smart Sheriff infrastructure at an extremely high likelihood of compromise or error. Versions should be current and patch levels should be no more than days or weeks behind at the very most.

The Apache services also appear to be misconfigured, leaving default documents and test code that also can be used to enumerate and compromise infrastructure.

[39] Identified in the HTTP response header as "Apache/2.0.59 HP-UX_Apache-based_Web_Server (Unix) DAV/2 mod_jk/1.2.27-dev"

- HTTP TRACE is enabled, suggesting the host is vulnerable to XST.
- X-Frame-Options header is not included in HTTP response: Clickjacking possible.
- inodes are leaked via ETags: 0xW/5222 0x138987 5471000
- Cookie JSESSIONID is created without httponly flag, allowing access by JavaScript. A malicious script could transmit cookies to another site. A session cookie transmission would enable session hijacking.
- /test/ directory: RSS feed, test result {“rslt”:”ok”}
- /docs/ directory: Apache Tomcat 6.0
- Default files, such as example servlets, should not be kept on server.
- Javascript on the server also leaks gratuitous and personal details. For example /js/common.js repeatedly sends a comment to users that could be useful in phishing or revealing further issues:

```
DATE       VER   DEVELOPER

2008.12.01      1.0   PARK HAK JIN
```

Issue 6.4: Development Resources Exposed and Domains Possibly Open for Registration (Severity: Low)

Although the Korean Mobile Internet Business Association is the public distributor of Smart Sheriff, the primary developer appears to have been a company called at varying times “101GT” or “Wigsys.”[40] The mobile app reveals a number of URLs that reside on MOIBA hosts that should neither be exposed to the Internet nor included in the sources of a production app. This includes URLs from the following list, specifically URLs that indicate that their sole purpose is to offer test and debug features:

- `http://192.168.0.5:8083`
- `http://220.117.226.129`
- `http://220.117.226.129:8082`
- `http://220.117.226.129:9090/demo-gcm-server`
- `http://hikdev.cafe24.com/demo-gcm-server`

While some of these addresses are internal network references or currently inaccessible, it is assumed that the demo servers and debug software increase the attack surface. The application has embedded to it multiple references to development properties, including a site at the address hikdev.cafe24.com that has expired. Additionally, other domain names associated with the original developer have expired, and may provide resources or further private information on the service. Given the gravity of other documented issues and the scope of this audit, this path was not fully explored and may demonstrate further issues.

[40] Cure53, “Pentest-Report Smart Sheriff 07.2015,” p. 15 and CLSI participants.

Issue 6.5: Erroneous Queries Expose Internal Database Structure (Severity: Low)

As noted previously, the Web interface for Smart Sheriff fails to consistently perform basic validation of user input at the back end, allowing the submission of unsanitized or erroneous data to the back-end Oracle database.[41] Where non-null values are required for database operations, such as correlating information in separate tables, the query will incur a database error that is exposed to the user. This error includes query statements that disclose internal information, which could lead to further enumeration of vulnerabilities.

We did not attempt to enumerate all potential database errors in consideration of broader issues reported here as well as concerns about disruption of services. However, this both demonstrates the absence of basic back-end filtering of requests and runs contrary to the core practice of removing debug information on production systems.

Found Errors (Formatted for Readability)

```
org.springframework.dao.DataIntegrityViolationException:

### Error updating database. Cause:
java.sql.SQLIntegrityConstraintViolationException: ORA-01400: NULL 을
(“MOIBA”.”MTB_PARENT_INFO”.”PARENT_MOBILE”) 안에 삽입할 수 없습니다

### The error may involve parent.parentDao.mergeParentChildInfo-Inline

### The error occurred while setting parameters

### SQL: MERGE INTO MTB_PARENT_INFO

      USING DUAL ON (PARENT_MOBILE = ?, AND CHILD_MOBILE = ?)

      WHEN NOT MATCHED THEN

            INSERT (PARENT_MOBILE, CHILD_MOBILE, REG_ID, REG_DATE)
            VALUES(?, ?, ?, SYSDATE)

      WHEN MATCHED THEN

            UPDATE SET MOD_ID = ?, MOD_DATE = SYSDATE
```

[41] CLSI participants.

```
### Cause: java.sql.SQLIntegrityConstraintViolationException: ORA-01400:
NULL 을 (“MOIBA”.”MTB_PARENT_INFO”.”PARENT_MOBILE”) 안에 삽입할 수 없습니다

; SQL []; ORA-01400: NULL 을 (“MOIBA”.”MTB_PARENT_INFO”.”PARENT_MOBILE”)
안에 삽입할 수 없습니다

; nested exception is java.sql.SQLIntegrityConstraintViolationException:
ORA-01400: NULL 을 (“MOIBA”.”MTB_PARENT_INFO”.”PARENT_MOBILE”) 안에 삽입할 수
없습니다
```

```
org.springframework.dao.DataIntegrityViolationException:

### Error updating database. Cause:
java.sql.SQLIntegrityConstraintViolationException: ORA-01400: NULL 을
(“MOIBA”.”MTB_CHILD_INFO”.”CHILD_MOBILE”) 안에 삽입할 수 없습니다

### The error may involve child.childDao.mergeChildInfo-Inline

### The error occurred while setting parameters

### SQL: MERGE INTO MTB_CHILD_INFO

      USING DUAL ON (CHILD_MOBILE = ?)

      WHEN NOT MATCHED THEN

            INSERT (CHILD_MOBILE, CHILD_NAME, CHILD_NCKNM, CHILD_BIR_YMD,
            CHILD_SEX, MEMBER_STAT_CD, STAT_MOD_DATE, REG_ID, REG_DATE,
            REGISTRATION_ID, SMRT_PHN_OS, OS_TYPE, SERVICE_USE_AGREE_YN,
            PRVT_INFO_USE_AGREE_YN, PRVT_INFO_OFFER_AGREE_YN,
            PRVT_INFO_COL_AGREE_YN)

             VALUES(?, ?, ?, ?, ?, ‘01’, SYSDATE, ?, SYSDATE, ?,?, ?, ?, ?,
            ?, ?)

      WHEN MATCHED THEN

            UPDATE SET CHILD_NAME = ?, CHILD_NCKNM = ?, CHILD_BIR_YMD = ?,
            MEMBER_STAT_CD = DECODE(MEMBER_STAT_CD,’03’,’02’,’01’),
            STAT_MOD_DATE = SYSDATE, MOD_ID = ?, MOD_DATE = SYSDATE

### Cause: java.sql.SQLIntegrityConstraintViolationException: ORA-01400:
NULL 을 (“MOIBA”.”MTB_CHILD_INFO”.”CHILD_MOBILE”) 안에 삽입할 수 없습니다
```

```
; SQL []; ORA-01400: NULL 을 (“MOIBA”.”MTB_CHILD_INFO”.”CHILD_MOBILE”) 안에
삽입할 수 없습니다

; nested exception is java.sql.SQLIntegrityConstraintViolationException:
ORA-01400: NULL 을 (“MOIBA”.”MTB_CHILD_INFO”.”CHILD_MOBILE”) 안에 삽입할 수
없습니다
```

Note on S-Dream

Smart Sheriff’s complementary message-monitoring application, S-Dream, was not within the scope of our research because of its smaller installation base according to Google Play statistics.

However, on cursory inspection, we find similar failures to protect user data in transit that could be further leveraged for disclosure of information, account access, or misrepresentation of user behaviour through forged requests. S-Dream appears to report back messages that it deems to be troublesome in the clear. Additionally, S-Dream appears to share common infrastructure, and potentially code or APIs, with Smart Sheriff. The compromise of Smart Sheriff’s back end, through either unauthenticated or unencrypted calls to the site, as well as the out-of-date software stack, would similarly affect users of S-Dream.

Given these considerations and substantial findings on Smart Sheriff, we encourage further evaluation of the practices and implementation of S-Dream, concurrent to addressing the issues that we identify here.