Research on Android Application Package
Stealth Download Hijacking

Xiao Fu\textsuperscript{1*}, Zhijian Wang\textsuperscript{1}, Yong Chen\textsuperscript{2}, Feng Ye\textsuperscript{1}, Zhu Zhu\textsuperscript{1} and Zhang Yihua\textsuperscript{1}
\textsuperscript{1}College of Computer and Information, Hohai University, 211100 Nanjing, China
\textsuperscript{2}Nanjing Longyuan Micro Electronic Technology Co. Ltd., 211106 Nanjing, China

ABSTRACT

Nowadays during the distributing and downloading of Android application packages, it is always be vulnerable to download hijacking attacks. Traffic analysis could be used by sites to detect if they are under this kind of regular download hijacking attacks. Unlike the regular ones, the stealth download hijacking attacks cannot be discovered by using such a method. By studying in an actual case, this paper presents a vulnerability of android application package download hijacking, which can be exploited to implement a stealth download hijacking by deploying bypass devices. And the victim sites can hardly notice it by using current methods. The cause, influence and mechanism of the exploit are discussed in this paper, and we also strive to give a solution for it.

RESEARCH BACKGROUND

At present days Android has become the most popular operation system used by mobile intelligent device in China. Since the end of 2016, the number of smart phone users had been more than 13 billion \cite{1} and more than 5.22 billion smart phones in 2017 were sold in the country \cite{2}. More than 83.2\% of the equipment had installed Android as their operation systems \cite{3}.

Meanwhile, for obvious reasons Android users within China cannot get access to Android Play market to download applications. Therefore third-party application markets and publishers’ sites have substituted the role of main channel to distribute android applications. Owing to the low level of information security management, most of the markets and sites are vulnerable to malicious entities and have become priorities of cyber-attacks. Among those attacks, download hijacking is one of the commonest means.

Into the Hijacking

In downloading hijacking, attackers often deploy bypass devices in the network nodes between the downloaders and publishers, listening to the communication traffic data. Once captured a specific download request for targeted resources, bypass devices would pretend as the real publishers and response a masqueraded message, redirecting the downloaders to a locator designated by the attackers.

Download hijacking belongs to the typical MITM (Man-In-The-Middle) attacks, which also exit in various forms ranging from session hijacking, browser hijacking to drive-by download. In session hijacking, attacker intercepts and takes over a
legitimately established session between a user and a host. The user-host relationship can apply to access of any authenticated resource, such as a web server, Telnet session, or other TCP-based connection [4]. Nowadays session hijacking could be easily detected by using SSO (Single Sign On) technology. Once session theft by attackers, the victim could instantly noticed and take steps to generate new sessions.

In browser hijacking, the user is not infected with regular malware but, while connected to a malicious or compromised website, front end languages such as JavaScript allow the user’s browser to perform malicious activities; in fact, attackers usually operate within the scope of actions that a browser is expected to execute. This kind of hijacking could be detected by deploying a propose detector installed into users’ web browsers which was trained with different instances of “normal” and “hijacked” browser behavior [5].

Drive-by download is more common than the hijackings above. When an Internet user visits a malicious web page, a malicious web server delivers a HTML document including malicious content to the user's computer system. The malicious content then exploits vulnerabilities on the visitor's computer system, which include vulnerabilities in web browsers, plug-ins, and operating systems. The exploitation leads to executing malicious code provided by attackers and the installation of malware on the visitor's computer systems. Detection of drive-by download attacks is an active area of research. Some methods of detection involve anomaly detection, which tracks for state changes on a user’s computer system while the user visits a webpage. This involves monitoring the user’s computer system for anomalous changes when a web page is rendered [6].

Meet the Android

To Android application package download hijacking, it seems like one combination of all the attacks above: Normally application publishers (usually third-party application markets or sites of developer) distribute the download links of Android application packages in the form of URL (Uniform Resource Locator) via HTTP (Hyper Text Transfer Protocol). Users could get the application packages by using web browsers or the other download managers to request the URLs. If one of those HTTP requests, whether in POST or GET method, has been sniffed and interrupted by a bypass device, a masqueraded HTTP 302 (Temporarily Moved) response would be sent to the user, and the request would be redirected to an URL designated by the attackers, sending another request. The second request is often destined to a malicious application package. In the application layer, HTTP session was hijacked. In the download method, the users’ browser was hijacked. And in terms of results, a malicious application was drive-by downloaded. So we can say, android application package download hijacking is an all-in-one hijacking involved with many kinds of hijackings. This made Android application package download hijacking much more comprehensive and harder to detect than the previous hijackings.

The attack method above could replace the Android application packages users tried to download with any data designated by the attackers. But due to the resources in the real URLs never have been downloaded, the publishers could easily detect if they are under download hijacking attack by using web traffic analysis. Comparing to UV (Unique Visitor), an abnormal decrement of PV (Page View) is a significant sign of been attacked by download hijacking.
To distinct from the attack studied in this manuscript, we call the download hijacking mentioned above Regular Download Hijacking as shown in Figure 1. Seemed rather different from Regular Download Hijacking, Stealth Download Hijacking which would be introduced below could barely be detected by publishers in any means of web traffic analysis. Unless reported and alerted by the users, this kind of attack is almost invisible to the publishers. Formerly, Stealth Download Hijacking was deemed to be a part of ACW (Advanced Cyber Weapon), being utilized by APT (Advanced Persistent Threat) to scheme and perform cyber-attacks against high value targets, which was hard-to-reach ordinary users. In the early of 2017 we have detected Stealth Download Hijacking was been used in a massive cyber-attack against ordinary civilian telecom users.

**DETECTION OF ATTACK**

In early 2017, multiple complaints by optical broadband network users of a specific ISP (Internet Service Provider) from Jiangsu province came into notice of us. According to the complains, the users claimed that Android application packages they intended to download from sites or application markets had always been replaced by irrelevant application packages such as “360 Mobile Assistant”, “PP Assistant”, “Wandoujia Market”, etc. After complained to the ISP custom service, those users were told that their smart phones had been infected by virus or malwares, and the ISP was totally innocent.

Those interconnected events have successfully attracted our attention. After connected with several developers whose Android applications packages were reported to be replaced and approached several web traffic analysis, we found that the number of PV and UV from the ISP of Jiangsu Province showed no evidence of any abnormal lower. In another word, there was no significant signs of been attacked by download hijacking. This event was no match with the scheme of any known attacks so we decide to detect this attack through an experimental evaluation.
Simulation Experimental Evaluation

In order to obviate interferences, the experiment environment was deployed on a completely new created virtual machine in VMware. The operation system was Microsoft Windows 7 and web browser was Internet Explorer 11, the network interface was PPPoE (Point to Point Protocol over Ethernet) through optical broadband network. According to the users’ report, we used browser to directly download the android application package “CIBN Network TV” on the site “www.91vst.com”. The link was located to a CDN (Content delivery network) peer whose IPv4 address was 58.222.18.2, located in Taizhou city, Jiangsu province.

We have tried three times to download the link and all the requested files was responded from two IPv4 address, 115.231.86.9 and 115.231.86.10, both of which were located in Yiwu city of Zhejiang province. Which was much more inexplicable, all the contents of downloaded files were not same ranging from “360 Mobile Assistant”, “PP Assistant” to “Wandoujia Market”. None of them was the application package we attempted to download. Then we used curl win64 built version to download the same resource and got the same result. Then we connected to a VPN (Virtual Private Network) server and tried another three more times to download the same link. All the files was responded from 58.222.18.2, and all the content was “CIBN Network TV”. Based on the results, we have determined that there was an existing download hijacking attack between the ISP network and the CDN peer.

Network Traffic Analysis

To determine the method in which attackers performed the hijacking between users and sites, we installed Wireshark and WinPcap, in order to execute a network traffic analysis. When the link was clicked in browsers, a HTTP request would have been sent to the site 58.222.18.2 by the browser. We call it Request0 for convenience. The HTTP header of Request0 was shown in TABLE I:

| Method: | GET |
| URL: | /vst/apk/updateDownload/Z2J9KVYTBjcFLyYQewTX.apk HTTP/1.1|
| Accept: | text/html, application/xhtml+xml, */* |
| Referer: | http://www.91vst.com/ |

Then the web browser received a HTTP response from 58.222.18.2. We call it Response1 and the HTTP header of it was shown in TABLE II:

| Status Code: | HTTP/1.1 302 Moved Temporarily |
| Content-Type: | text/html |
| Location: | http://115.231.86.10:9780/L06/W_D_J.apk |
| Accept: | text/html, application/xhtml+xml, */* |
| Referer: | http://www.91vst.com/ |
In the header it can be seen that Response1 was a 302 Temporarily Moved response, the content of which was to inform the web browser the requested resource on the link was temporarily removed to an URL “http://115.231.86.10:9780/L06/W_D_J.apk”. The TTL (Time-To-Live) value of Response1 was 113, and the ordinary TTL of responses from 58.222.18.2 was 56. According to that, it could been determined Response1 was a masqueraded response sent by the bypass device deployed by attackers, not from the real site 58.222.18.2. Once Response1 was received by the web browser, a new request would be sent by web browser to 115.231.86.10, we call it Request1. The HTTP header of Request1 was shown in TABLE III:

| Method: | GET |
|---------|-----|
| URL:    | /L06/W_D_J.apk HTTP/1.1
| Accept: | text/html, application/xhtml+xml, */*
| Referer: | http://www.91vst.com/

When Request1 was responded by 115.231.86.10, an application package of Wandoujia Market would been downloaded by the web browser. So far the method was almost the same as Regular Download Hijacking attack: sending a masqueraded HTTP 302 (Temporarily Moved) response to redirect the request to 115.231.86.10 and to deceive web browser into downloading designated packages. The weird part was, in this case all the users had not downloaded their requested packages on the site 58.222.18.2, but the traffic logs on CND peer 58.222.18.2 showed that all the users had downloaded the packages successfully. That means if the users did not execute a network traffic analysis, this hijacking would never been detect by traffic analysis on site. The method of this hijacking had remarkable characteristics of Stealth Download Hijacking, which differed from all the known download hijacking attackers we have detected before.

For this reason, we expanded our searching range of traffic analysis from HTTP traffics to all TCP/IP (Transmission Control Protocol/Internet Protocol) traffics.

The TCP/IP header of Request0 was shown in TABLE IV:

| Internet Protocol Version 4 |
|----------------------------|
| Src: | 192.168.0.110 |
| Dst | 58.222.18.2 |

| Transmission Control Protocol |
|-------------------------------|
| Src Port: | 3771 |
| Dst Port: | 80 |
| Seq: | 1 |
| Ack: | 1 |
| Len: | 344 |
The TCP/IP header of Response1 was shown in TABLE V:

| Internet Protocol Version 4 |
|----------------------------|
| Src: 58.222.18.2           |
| Dst: 192.168.0.110         |

| Transmission Control Protocol |
|-----------------------------|
| Src Port: 80                |
| Dst Port: 3771              |
| Seq: 1                      |
| Ack: 1                      |
| Len: 129                   |

The TCP/IP header of Request1 was shown in TABLE VI:

| Internet Protocol Version 4 |
|----------------------------|
| Src: 192.168.0.110         |
| Dst: 115.231.86.10         |

| Transmission Control Protocol |
|-----------------------------|
| Src Port: 3772              |
| Dst Port: 9780              |
| Seq: 1                      |
| Ack: 1                      |
| Len: 306                   |

It can be clear that all the Seq (Sequence Number) and Ack (Acknowledgment Number) value in TCP/IP header of Request0, Response1 and Request1 were 1, which means in the processes of sending above requests and responses, a new session had been created in each time, none of the requests and responses were in a same TCP session connection [7]. Therefore, both of the browser and site had correctly handled the requests and response from the other, due to the no-connection and no-state features of HTTP [8]. The stateless of HTTP allows each response from sites does no need to be semantically related in context with the last request from browser. This feature was designed to free the connection between browsers and sites in order to conserve resources during requests and responses.

For example, in the TCP/IP header of Request0, Seq was 1, Len (Segment Length) was 344. So if in the same TCP session, in the TCP/IP header of Response1 which assuming to be the response to Request0, Ack should be 345, calculated by adding Seq and Len of Request0 (1+344). In fact Response1 did not observed strictly to this rule, but the web browser still handled it as the response to Request0.

By searching in TCP/IP traffic, one response had a Seq of 1 and an Ack of 345 has been found between Request0 and Response1. We call it Response0 and the TCP/IP header of it was as below:

According to the information above, it can be seen clearly Response0 only had a header, and the segment length was 0 as shown in TABLE VII which means Response0 had an empty content of body.
Between Request1 and Response1 in timeline we found a response that had the same Seq and Ack as Response0. We call it Response2 and the TCP/IP header of it was shown as TABLE VIII:

### TABLE VIII. TCP/IP HEADER OF RESPONSE2.

| Internet Protocol Version 4 |  |  |
|-----------------------------|-----------------|-----------------|
| Src:                        | 58.222.18.2     | 58.222.18.2     |
| Dst:                        | 192.168.0.110   | 192.168.0.110   |

| Transmission Control Protocol |  |  |
|-------------------------------|-----------------|-----------------|
| Src Port:                      | 80              | 80              |
| Dst Port:                      | 3771             | 3771             |
| Seq:                           | 1               | 1               |
| Ack:                           | 345             | 345             |
| Len:                           | 1412            | 1412            |

Because the TCP/IP headers of Response0 and Response2 had the same Seq and Ack, the TCP stack of experiment environment discarded Response2, which was arrived later than Response0 in timeline, as a TCP Out-of-Order error response. The browser just discarded Response2 and did not approach it. So that was the reason why traffic analysis on site failed to detect the hijacking: all the requested resources had been sent by site and arrived at browser, but the browser ignored them as TCP errors.

**Research on Attack Method**

By analyzing the content of Response2, it came to us that Response2 was the true application package responded from 58.222.18.2 as shown in Figure 2.

![Content of message response2](image-url)
To tell which one, Response0 or Response2, was the true response, is not very comprehensive. By transferring the hex of body content in Response2 into text, it can be seen that Response2 contained a header of HTTP response with a content-type of Android package, whose length was 26615091 bytes. We searched all the packages in timeline and had found a sequence of HTTP Continuation packages following up with Response2. By adding them up we got a copy of Android package which we intended to download. Meanwhile, Response0 only had an empty body content, though it had the same Seq and Ack with Response2.

Till now the attack mechanism of hijacking is clear. The site 58.222.18.2 had properly sent a full copy of Android package requested in Request0 to browser, by streaming it into a HTTP response. The HTTP response was divided into multiple packages when passed through gateways in order to be within the limitations of MTU (Maximum Transmission Unit). The very first package, in another word Response2, contained the header information which was necessary for browser to decide how to approach it. Bypass device deceived the browser into discarding Response2 by sending a masqueraded Response0, which occupied the Req and Ack of Response2. Once Response2 was discarded by browser, all the other following up packages arrived at browser would have become invalid as shown in Figure 3, just like cutting off the head of a snake.

Figure 3. HTTP continuation following up with response2.

For the above-mentioned facts, the temporal relations of each requests and responses were shown in Figure 4.

Figure 4. Temporal relations of each message in stealth download hijacking.
During an abnormal downloading, since Request0 sent by the users’ web browser was responded by the site 58.222.18.2, Response2 would have been received and the downloading process completed correctly.

But during a Stealth Downloading Hijacking attack, once the bypass device captured Request0, a masqueraded Response0 and Response1 would have been sent as from 58.222.18.2. The Len of Response0 was 0 and the Seq of its TCP/IP header was 1, Ack was the value Len of Request0 adds Seq of Request0. The purpose of sending masqueraded Response0 was to take up the Ack value of Response2 which was the true response from 58.222.18.2, so that Response2 would have been discarded by web browser as a TCP Out-of-Order error. Since 58.222.18.2 sent Response2 as a response, the logs on 58.222.18.2 would have been recorded as Request0 was responded correctly and a successful downloading was completed. Response1 was a temporarily moved response, its purpose was to redirect users’ request to an URL designated by the attackers which was on the site 115.231.86.10. Together Response0 and Response1 accomplished a Stealth Download Hijacking. The greatest harm of Stealth Download Hijacking is, both of users and sites paid for their bandwidth, none of them could download or distribute the real application package. Especially for the sites, all the servers’ logs and status showed increasing downloads and perfect networks streams, without any interrupted downloads or PV decrement.

**IMPACT OF VULNERABILITY**

Attacker exploited the vulnerability in the no-connection and no-state features of HTTP by taking up Ack value in TCP/IP header of abnormal response, redirecting the request to a designated URL without interrupting true respond from the site. This vulnerability exists in HTTP protocol, theoretically impacting all the sites using HTTP to publish resources. Experimental and observational evidence concerned that only the HTTP requests in GET method with an URL contained “.apk” string had been hijacked. It means that the initial targets of hijacking possibly were the Android devices because the MIME (Multipurpose Internet Mail Extensions) type of resources that had an extension name of “.apk” was “application/vnd.android.package-archive” [9].

But in the experiment, all the operation systems ranging from Windows to Ubuntu had been impacted. We believe it was because the attacker had failed to identify the User-Agent of users’ web browsers correctly [10]. The attacker was intended to hijack the download requests of Android devices but all the browsers with different User-Agent values had been infected, possibly owing to the poor technique of the attacker. iOS and macOS devices were not the primal targets of this attack because their users were not permitted to download and install application packages from any other third-party markets and untrusted resources. Downloading packages from Apple Store was not impacted by the vulnerability.

Within Jiangsu province almost all the android application package published and distributed via HTTP has been impacted by the vulnerability, including third-party markets and developer sites. Some of the applications having an in-application updating function (also called Hot Swapping [11]) impacted by the vulnerability, because HTTP requests were sent in updating and sniffed by bypass device. Most popular Android applications were affected by this attack within Jiangsu province, ranging from online videos, comic markets to electronic business platforms, including CIBN Network TV,
Comic Home, Suning Appliance, and the list of victims is still increasing. We have informed all the confirmed victims and suggested them to update to HTTPS (Hyper Text Transfer Protocol over Secure Socket Layer), in order to defend this attack.

According to statistics at the end of 2016 the number of broadband network users within Jiangsu province was over 28 million [12] and among it the number of optical users were more than 10 million [13]. Based on this data the victim number of this attack would have been no less than 10 million and possibly up to 28 million. According to the average daily downloading frequency of 5 times per user, the attacker could obtain more than 50 million downloads of designated resources, which would generate a 250 thousands RMB income per day according to 5 RMB for every 10 thousands downloads which was the average promotional price of android application packages.

In any case, the owner of 115.231.86.9 and 115.231.86.10 benefited most from this attack by stealing download traffics from users and sites. According to WHOIS records on APNIC (Asia-Pacific Network Information Center) web site, these two addresses were owned by a shell company in Zhejiang province called Yiwu Huachen Network which has a suspected benefit transportation with senior managers in China Telecom. However it was hard to believe a shell company in Zhejiang province could have an ability to deploy bypass deceives into backbone network of Jiangsu province. We informed the ISP custom service and shared all the evidences collected with them, requesting to establish an inner audit. Our appeal went unanswered.

PREVENTION OF VULNERABILITY

As mentioned above, the attacker accomplished Stealth Download Hijacking attack by deploying bypass devices which sent masqueraded Response0 and Response1. So if the masqueraded responses could be filtered or intercepted, the users can be prevented from attacker by exploiting such a vulnerability.

Some of the victim users reported that they had firewall and IDS (Intrusion Detection Systems) deployed as perimeter protection. After several evaluations, we found that all the firewalls and IDSs were transparent to Response0 and Response1. Preset rules allowed Response0 and Response1 to pass through unimpeded. This is because these two packages contained no illegal characteristic for firewall or IDS to distinguish them.

The characteristic of Response0 was obvious. Response0 was masqueraded in order to occupy Seq and Ack values of Response2 by attacker. If we filter the TCP packages with a Len value of 0, Response0 could be intercepted easily and Response2 which was the true response could be downloaded by browser. Notice that not all the TCP packages with a Len of 0 is a masqueraded one. While a TCP session was establishing, three packages would have be sent by both sides. This process was widely known as Three Times Handshake, and all the three packages in handshake process had a Len of 0. But in handshake, none of Ack or Seq would be more than 1. So we set rule to filter Response0: stop all the TCP packages if their Len is 0 and any of Ack or Seq is more than 1. Be alert this rule would cause a false positive: when a TCP session was closing, packages with Len of 0 would also have been sent by both sides. Adding this rule to firewall or IDS would disable the ability of remote servers to release TCP sessions proactively, the computers behind firewall or IDS must wait the TCP sessions to time out.
Once Response0 was stopped, Response2 could be approached properly by browser, and the requested application package would be downloaded. But the prevention is not done yet, we met another problem: double response. When a link was clicked, two application packages would be downloaded, one of which is the requested application, and the other is a random one from 115.231.86.9 or 115.231.86.10. To deal with the problem, we need to filter Response1 too.

Response1 was masqueraded in order to redirect users’ download to the resources on 115.231.86.9 or 115.231.86.10. Filtering all the HTTP response containing a string of 115.231.86.9 or 115.231.86.10 could disable Response1 from misleading browsers to the attacker designated resource. Unlike Response0, adding the rule to filter Response1 would not cause any false positive or side effects: 115.231.86.9 and 115.231.86.10 were the attacker’s foot stand and golden goose, if no one downloads anything on 115.231.86.9 and 115.231.86.10, their theft could not steal any traffic from victims. If Response1 was stopped, double response would have been solved.

When both rules were set, Response0 and Response1 could be stopped temporality. In the user-ends or soft router with Linux installed, prevention can be made by configuring iptables/netfilter. And in the other ends or systems, third-party modules or application are needed to do such a prevention. We have tested these rules on several Linux based operation systems like Debian, Ubuntu and CentOS, all of them worked fine.

Application packages provided via HTTPS (Hyper Text Transfer Protocol over Secure Socket Layer) URLs have not been affected by the vulnerability. It was trusted that bypass device did not have the ability to identify the HTTPS requests encrypted by TLS (Transport Layer Security Protocol) or SSL (Secure Sockets Layer).

CONCLUSIONS

Through a case study, this paper presented vulnerability and exploit of android application package download hijacking. In this paper the principle, detection, analysis and prevention of the vulnerability were researched and discussed, expecting to provide first hand materials and data to network security researchers at home and abroad.

Comparing to previous Regular Download Hijacking, Stealth Download Hijacking is more concealed and socially harmful for it cannot be detected by network traffic analysis on the victim site. We believe this kind of attack was performed by some APT groups [14] with social resources and techniques, which is necessary to be awarded by ISP internal audit departments and law enforcements. We had spent months of time to evaluate, survey, analysis this case, to inform the victims, to trace, observe, cyber-manhunt the attacker. Although it is still beyond our ability and courage to bring them to justice, the only thing we can do now is to study their method, and to use it in something righteous.

ACKNOWLEDGMENT

This paper was supported by the Fundamental Research Funds for the Central Universities (2016B14014).
REFERENCES

1. Hanyu L. 2016. “Mobile users in China have exceeded 1.3 billion,” Computer and network, 2016, 42(3), pp.19.
2. Worldpanel K. 2017. “Smartphone OS sales market share,” Kantar World Panel, https://www.kantarworldpanel.com/mobile-os-market-share/.
3. iiMedia R. 2017. “2016-2017 China Smartphone Market Monitoring Report,” iiMedia Research, March 2017, pp. 6-7. http://www.iimedia.cn/148940153308953147.pdf.
4. Oriyano S. P. 2017. “CEH™9: Certified Ethical Hacker Version 9 Study Guide,” John Wiley & Sons, Inc, May 2017, pp.331.
5. Monica D., and Ribeiro C. 2015. “An IDS for Browser Hijacking,” SECURWARE, August 2015.
6. Le V.L., Welch L., Gao X. 2013. “Anatomy of drive-by download attack,” Eleventh Australasian Information Security Conference. Australian Computer Society, Inc., 2013, pp.49-58.
7. Postel J. 2001. “RFC 793: Transmission Control Protocol,” Internet Engineering Task Force (IETF) Request for Comment, pp.15-21.
8. Fielding R., Gettys J., and Mogul J. 1999. “RFC 2616: Hypertext Transfer Protocol,” Internet Engineering Task Force (IETF) Request for Comment, pp.44-47.
9. Yang A., Steele S., and Freed N. 2012. “RFC 6532: Internationalized Email Headers,” Internet Engineering Task Force (IETF) Request for Comment, pp.12-13.
10. Fielding R., and Reschke J. 2014. “RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content,” Internet Engineering Task Force (IETF) Request for Comment, pp.46-47.
11. John L., and David A. 2002. “Computer Architecture: A Quantitative Approach,” The Morgan Kaufmann Series in Computer Architecture and Design, Morgan Kaufmann, pp.707.
12. Jiangs C.A. 2017. “Annual rolling development plan of information and communications industry in Jiangsu in 2017,” Jiangsu Communications Administration, May 2017, http://www.jsca.gov.cn/xxgk/ghtj/fzgh/201705/a20170527_46254.html.
13. Xiu Xia Z. 2015. “Jiangsu Telecom optical network user number ranking first in the country's provinces,” Xinhua News, December 25, 2015, http://www.js.xinhuanet.com/2015-12/25/c_1117574512.htm.
14. David L. 2013. “Advanced Persistent Threats: How to Manage the Risk to Your Business,” Information Systems Audit and Control Association, pp.11-13.