SlowDroid: Turning a Smartphone into a Mobile Attack Vector

Enrico Cambiaso*, Gianluca Papaleo†, Maurizio Aiello‡
National Research Council, CNR-IEIIT,
via De Marini, 6,
16149 Genoa, Italy
Email: *enrico.cambiaso@ieiit.cnr.it, †gianluca.papaleo@ieiit.cnr.it, ‡maurizio.aiello@ieiit.cnr.it

***Abstract*—Nowadays, last generation of smartphones are comparable to desktop computers in terms of computational capabilities. Such characteristics can turn a smartphone into a mobile attack vector. In this paper we analyze the use of mobile devices to perpetrate cyber attacks. We present a mobile threat, SlowDroid, running on Android operating system. Such menace implements a Denial of Service attack and it is particularly suitable to a mobile execution, since it makes use of low amounts of computational and bandwidth resources. We present in detail SlowDroid implementation and our choices in terms of design, graphical user interface, and system architecture.**

Keywords*—*android, mobile attack, cybersecurity, slow dos attack, denial of service

I. Introduction

In the modern era, with the advent of Internet ready mobile devices, a big slice of computing market shifted from personal computers to smartphones and portable devices. Companies like Google, Apple, and Microsoft are investing time and resources on mobile development, continuously introducing mobile oriented software and hardware with even more powerful capabilities. In this context, last generation smart devices are equipped with high performance processors, large memory drives, a suite of sensors, and different network connection modules. Thanks to this evolution, mobile devices are now able to perform almost every activity associated to desktop computing, offering in addition functionalities such as portability and sensors-endowed hardware.

Historically, mobile devices always represented a target for attackers, injecting malware, trojans, or viruses inside the device, since phones are able to access sensitive user data. The first attack against mobile phones arrived in 2000 as a worm known as *Timofonica*, designed to send SMS text messages to randomly generated numbers [1]. Later, with the advent of the smartphone era, a wide range of attacks appeared. Some examples are the *CommWarrior* worm for Symbian OS [2] or the *FakePlayer* malware for Android [3]. Recently, some kind of "physical" attacks like the *smudge attack* also appeared, executed in order to detect unlock patterns analyzing the smudges on touch screen surfaces [4].

Although mobile devices always represented a target for attackers, they have rarely been used as an attack tool. In this paper we focus on the adoption of mobile devices for executing network based threats. We report as a test case the SlowDroid attack [5] we have implemented on the Android mobile platform. The attack implements a Denial of Service (DoS) menace, executed to make a network service unavailable for its legitimate users. While the first generation of DoS attacks were based on a particular *exploit* or on *flooding* the victim with a large amount of network traffic, novel threats, known as *Slow DoS Attacks* (*SDAs*) [6], are particularly suitable to a mobile environment, since they require tiny amount of network and computational resources. Moreover, since SDAs are considered particularly dangerous, a mobile execution of these menaces should represent an amplification of the threat, as a portable and position varying threat execution may elude detection methodologies.

The rest of the paper is structured as follows. Section II reports the related work of current attack tools. Then, Section III motivates the advantages of a mobile threats execution. Section IV describes in detail the implementation of the SlowDroid tool, while Section V analyzes how the implemented attack works. Finally, Section VI reports the conclusions of the paper.

II. Related Work

In this section we report the related work on the mobile threats topic. We start covering Slow DoS Attacks, then analyzing attacks solutions accustomed to mobile environments.

A. Slow DoS Attacks

The Slow DoS Attack (SDA) phenomenon represents a recent evolution of Denial of Service (DoS) attacks: such novel threats emerged in the last few years and consolidated as a dangerous menace on the Internet. Unlike the first generation of DoS attacks, SDAs make use of extremely low amount of bandwidth to make a listening service on the Internet unreachable. In order to do that, they usually work at the application layer of the ISO/OSI model, directly affecting the listening daemon running on the targeted host.

If we consider this category of attacks, the first attack is the *Shrew* one, designed to send an attack burst to the victim, giving the illusion of a high congestion on the network link [7]. Maciá-Fernández et al. [8] propose instead the *Low-Rate DoS attack against Application Servers* (*LoRDAS*), able to cause a DoS by executing attack bursts concentrated to specific instants.

Other attacks aren't related to research works, but instead they have been published on the Internet and obtained popularity thanks to the high impact offered. Among these attacks, one of the most known SDA is *Slowloris*, implemented by Robert "RSnake" Hansen [9]. This attack targets the HTTP protocol by establishing a large amount of pending requests with the victim [10].

The *Apache Range Header* attack, which should no longer be considered a menace [11], has been published on the Internet as a script by a user known as "KingCope" [12]. The attack exploits the byte-range parameter of the HTTP protocol and makes use of it to force the web server to replicate in memory a requested resource.

Instead, in 2012 Sergey Shekyan released `slowhttptest` [13], a tool able to accomplish a set of known SDAs. Together with the tool, his team introduces the *Slow Read* attack [14], able to slow down the responses of a web server through the sending of legitimate requests.

The SlowDroid attack proposed in this paper works similar to Slowloris. Nevertheless, our tool requires a minimum amount of bandwidth and computational resources, and it is therefore more suitable on a mobile environment. Moreover, unlike Slowloris, which is bounded to the HTTP protocol, the proposed menace can affect different protocols, since sent payload is not compliant to a specific protocol. Nevertheless, as payload can be customized by users, it is possible to send specific well-formed messages targeting particular protocols.

B. Mobile Attack Tools

If we consider network attacks, in [15] an exhaustive taxonomy of desktop based threats has been provided. Nevertheless, to the best of our knowledge, a mobile attacks execution has not been deeply analyzed yet. Indeed, until a few years ago, mobile phones have never been considered as attack vectors, but only a target for attackers: due to the limited software, hardware, and network resources, they were not used to perpetrate attacks. Recently, with the advent of smartphones and last generation mobile operating systems, such as Android, iOS or Windows Phone, designed to easily develop apps running on such environments, attacking tools started to move to the mobile world. We will now only consider Android operating system, that represents the most common mobile system on the market [16]. Moreover, Android OS has always attracted the hacking community, due to the assisted apps installation process, which does not require a third party approval, to the ease of obtaining administrator priviledges on the devices, and to the open source nature of the operating system.

Mobile attacking tools cover various kinds of malicious activities: *WiFiKill* [17] has been designed to exploit a shared wireless network, disabling Internet connection on specific devices connected to the same network.

Instead, *DroidSheep* [18] is an application for session hijacking: indeed, it allows to retrieve session cookies from devices on the same network, thus providing users the ability to use retrieved cookies, thus impersonating the victim. Another similar attack that intercepts web session profiles is *Faceniff* [19].

Unlike the proposed SlowDroid attack, the mobile threats mentioned until now require administrator priviledges on the device running them. This *rooted device* requirement represents an important limit, since most of the devices are not rooted.

If we consider instead mobile Denial of Service attacks, various applications [20], [21], [22], [23] are based on the *Low Orbit Ion Cannon* (*LOIC*) [24] desktop tool, an attacking software known for being adopted by the Anonymous hacktivist group. Although the implementations differ one from the other, they all act by flooding the victim with a high amount of packets, thus requiring large amounts of bandwidth and computational resources. Therefore, they are not suitable on a mobile environment with limited capabilities.

Instead, the (unofficial) Slowloris mobile app [25] represents a Slow DoS Attack implemented in a mobile enviroment. Nevertheless, the amount of bandwidth needed to execute an attack is particularly high compared to the SlowDroid requirements.

The SlowDroid attack presented in this paper should therefore be considered an innovative tool, since it has been designed to particularly fit on a mobile environment.

III. Mobile Deployment Advantages

As we have described above, we have implemented SlowDroid as a mobile application running on Android operating systems for smartphones and tablets [5]. The choice of such operating system is driven by the fact that currently it's one of the most used operating systems of the world [16]. Moreover, Android apps can be written in Java, a multi-platform programming language, thus offering the ability to reuse the same components for both mobile and desktop applications. On the other side, this attack is particularly effective if executed from mobile devices, due to the reduced attack network bandwidth and computational requirements.

From the attacker point of view, a mobile deployment of an attack tool is preferrable for many reasons. In the following, we briefly describe these key points.

Mobility: One first important reason is based on *mobility*: mobile devices was born for telecommunications and they are carried out by people for the entire day. Because of this, a mobile attack application offers the ability to execute an offensive operation from a wide range of places. Moreover, since such devices often provide various connection methods, such as Wi-Fi, 3G, LTE, or Bluetooth, the attack is usually conveyed through one of these channels.

For instance, we could think of a user being at the restaurant and exploiting the public Wi-Fi connection to execute an attack, without being noticed by the other customers of the restaurant.

Attack Hiding: Another key point directly related to the mobily feature is the *attack hiding*. Indeed, adopting an approach similar to the previous one, we could point out that the attack is not interrupted if the attacker is moving from a (i.e. 3G, 4G, Wi-Fi, etc...) cell to another one. In particular, in case the attacker makes use of horizontal/vertical handover [26], attack detection and mitigation are hampered, since it's more difficult to detect the continously varying source of the attack or a moving perpetrator.

For instance, we could imagine an adversary executing an attack while cycling. In this case, the attacker would easily pass unobserved while the attack would continue its operations.

IV. SlowDroid Implementation

In this section we describe the implementation of the SlowDroid tool. In particular, a certain amount of connections is opened with the victim, trying to seize all the service queue on the targeted host, thus trying to reach a DoS, and maintaining it during the time, until the attack is running. We have deployed the attack for Android operating system for mobile devices, releasing the tool both on the web and the Google Play Store as an Android application [5].

We will now describe in detail our implementation.

A. User Inputs

The application has been designed to require less information possible to the attacker, therefore an attack execution does not require mastering the topic. However, since payload requests may be customized, an attacker is still able to execute a custom and advanced attack, exploiting the targeted server.

1) Basic Settings: Following basic information are needed to address the attack to a specific target:

- *Server IP Address* (`String`) identifies the IP address or the domain name of the server to be targeted/tested;
- *Port* (`int`) identifies the listening port of the server;
- *Connections* (`int`) identifies the number of connection to establish and maintain active with the server during the attack execution;
- *Wait Timeout* (`int`) identifies the timeout used during the data sending phase, in seconds.

2) Advanced Settings: The SlowDroid app allows users to set up an advanced attack, by customizing the requests payload sent to the targeted server. In particular, following advanced settings can be provided:

- *Request Generation* (`enum`) identifies how request payload is generated. Three possible values are provided:
  - *Default*: in this case requests are composed by a sequence of spaces;
  - *Random*: in this case each character composing a request is randomly chosen, accordingly to the following regular expression:
    `[0-9a-zA-Z\-_␣.,;:?/=*]`
  - *Custom*: in this case requests payload respects the custom request format specified in the next setting.
- *Custom Request Format* (`String`) identifies the custom request to send as payload for the established connections[1].

3) Additional Settings: It is provided an additional setting:

- *Test Max Duration* (`int`) identifies the duration of the attack, in seconds: after such time passing the attack would automatically be interrupted.

[1]In case the *Request Generation* field is not configured to *Custom*, the *Custom Request Format* parameter is ignored.

In particular, since SlowDroid has not been designed with the purpose of maliciously attacking a service, we have limited the attack operations as much as possible. Indeed, the concept behind the *Test Max Duration* setting is that an attack can't be executed indefinitely, but it is sooner or later interrupted: the SlowDroid app is in fact configured to execute an attack for at most 3600 seconds. Behind the same concept there is the fact that the attack is automatically interrupted in case the device's screen is turned off, or in case the application losts the focus.

B. Graphical User Interface

In order to maintain compatibility with different versions of the Android operating system, the main Activity class implements `PreferenceActivity`, an Activity class provided in the Android development framework to manage user preferences. Such class allows developers to define the preferences structure, automatically generating the user interface. Therefore, the choice to adopt this library allows us to reuse already implemented and consolidated software components, integrating them in our app. For the same compatibility reason, along with system integration, we have integrated a menu which can be opened by users through the physical menu button on their devices, or by clicking on the menu button at the top-right corner of the app.

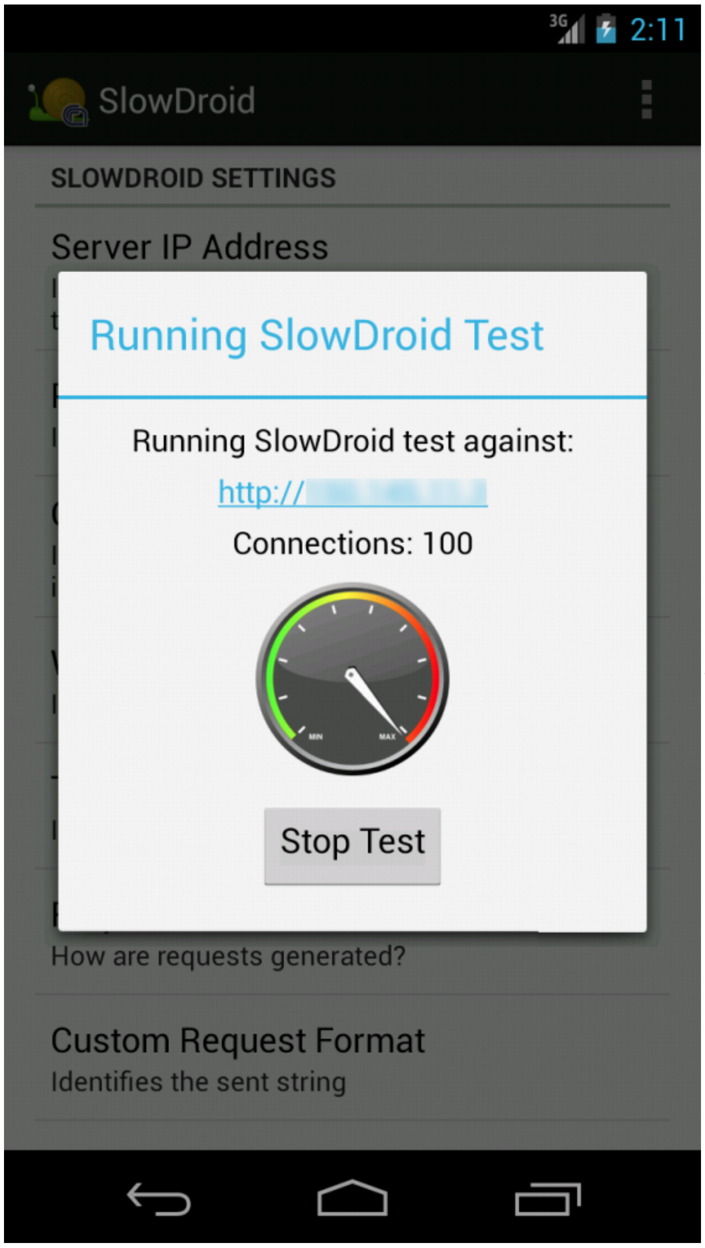
The menu allows users to execute a SlowDroid test/attack with current settings, or to obtain information developers information. Both the cases are treated similarly: in both the cases a `Dialog` object is opened. A dialog is a small window that does not fill the screen and is normally used for modal events.

We will now focus on the Dialog showing attack information, depicted in Figure 1. As shown in figure, users can stop the test/attack at any time.

Relatively to the number of connections established with the server, we show a percentage over the connections number specified by the user as a graphical odometer, through two different `ImageView` elements. Such percentage, along with the exact number of established connections, is updated every second. In order to enhance user experience, the odometer is updated by creating a `RotateAnimation` object, which gives users a more realistic impression.

C. SlowDroid and SlowDroid Library

In order to make the attack reusable, mostly on a desktop environment, we have implemented two different projects: the first one is a Java attack library designed to instantiate a new attack and execute it, while the second project, the SlowDroid application, makes use of the attack library and implements a mobile Android application. Indeed, since Android applications can import Java libraries, this splitting choice provides reusability of the attack library on different projects, maintaining compatibility with the Android application. For instance, the library may be (even stealthly) included in different (mobile or not) projects, designing a new executable application from scretch, building a more exhaustive test tool, or extending/enhancing the attack itself.


SlowDroid

SLOWDROID SETTINGS

Server IP Address

Running SlowDroid Test

Running SlowDroid test against:

http://[redacted]

Connections: 100

Stop Test

How are requests generated?

Custom Request Format

Identifies the sent string

Figure 1. SlowDroid Attack Dialog

V. Attack Description

The SlowDroid attack implements a variant of the SlowReq threat [27]. SlowDroid exploits a vulnerability on most server applications implementations, which usually limit the number of simultaneous connections to an extremely low value, in order to decrease the number of requests simultaneously managed by the server. SlowDroid implements a Slow DoS Attack that, unlike flooding DoS attacks, which aim to overwhelm the network of the victim host, directly affects the application layer of the victim, opening more request than the one the server is able to accept. In this way, due to the daemon limit relative to simultaneous connections, less attack bandwidth is requested. Therefore, due to the reduced available resources, the attack is particularly suitable to the mobile environment.

We will now describe in detail how attacks belonging to the same category of SlowDroid work.

A. *Long Requests DoS Attacks*

SlowDroid attack is a Slow DoS Attacks belonging to the category of *Long Requests DoS* attacks [6]. These threats work by sending a large amount of slow (and endless) requests to the server, saturating its buffer resources while waiting for the completion of the requests.

For completeness, we have noticed that while other attacks such as flooding based ones may compromise adjacent neighbors while tracking the route, this is not so immediate on Long Requests DoS attacks, since the target the application layer, which is not crucial for network nodes such as routers [28].

1) Slowloris: For instance, the Slowloris attack [10] exploits the HTTP protocol and it works by establishing a certain amount of connections with the victim and sending for each connection the content reported in Code 1.

```
GET / HTTP /1.1\ r\n
Host: www.target.com\r\n
User Agent : Mozilla /4.0 [...]\ r\n
Content-Length: 42\r\n
```

Code 1. The First Partial Content of a Slowloris Request

In this case the server would wait for the final `\r\n` characters identifying the end of the request. Nevertheless, a Long Request DoS attack would never send such characters, thus forcing the server to an endless wait. Moreover, since a server side connection close occurs in case no characters are sent during a certain time period, such menaces make use of a Wait Timeout to periodically send a low amount of data, thus preventing a connection closure. In particular, Slowloris typically and repeatedly sends the additional HTTP parameter reported in Code 2.

```
x  a :b\r\n
```

Code 2. The Partial Content of a Slowloris Request

2) SlowDroid: SlowDroid attack works similarly to other Long Requests DoS attacks such as Slowloris. Nevertheless, in this case at any period a single character is sent to the server. By default, a single space is always sent, but in general each non-newline character may be good. Just like other similar menaces, this behavior leads the server to a DoS.

Moreover, in this case a minimum amount of bandwidth is used, thus making the attack particularly suitable to a mobile environment, where resources are limited and expensive.

Additionally, since unlike attacks such as Slowloris payload content is not bounded to a specific protocol, SlowDroid is able to affect other protocols as well (i.e. SMTP, FTP, etc...).

B. *SlowDroid Effects on a Server*

Analyzing the attack effects on the server, it often occurs that in the moments immediately following the beginning of the attack all available slots of the server are occupied by the attacker. Therefore, in this case each new connections experiences a Denial of Service. This fact is confirmed in

Figure 2, which reports the number of connections established with an Apache2 web server without any protection module active during a 600 seconds attack execution.

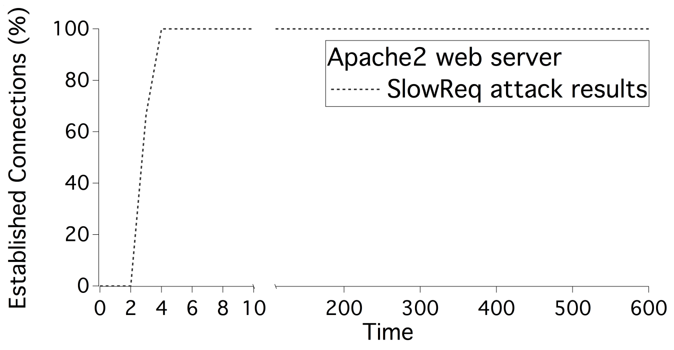


Figure 2. SlowDroid Effects on an Apache2 Web Server

It is possible to analyze that after about 4 seconds all the connections are seized by the attacker. In this case, no additional connections can be established (with the listening daemon) by legitimate clients, thus resulting in a Denial of Service experienced by the server. Therefore, any additional (legitimate or not) connection wouldn't be able to communicate with the listening daemon. Indeed, although it's possible to connect to the server, connection is not passed to the application layer, thus to the server, until a connection slot is freed on the server. Since this never happens, due to the attacker's behavior, the server would be under a DoS attack for the whole attack duration.

Nevertheless, after the launch of a SlowDroid attack, some active connections may already be established with the server, from other (legitimate) clients. These clients would be perfectly able to communicate with the server through the already established connections, until they are closed. At this point, since the attack aims to seize all the available connections on the server, the attacker would probably seize those connections as they become available.

Our SlowDroid implementation tries indeed to detect a connection close as it happens, re-establishing the connection with the server as soon as possible. Actually, there could be a race condition on these connections between the attacker and other legitimate clients. In practice, sooner or later the attacker would obtain the connections, since it aggressively and continuously tries to connect to the victim.

C. Attack Functioning

The SlowDroid attack opens a certain amount of connections against the victim. This connections number is usually greater than the maximum number of simultaneous connections the server can manage. In this way, by seizing all the available positions, a Denial of Service is potentially reached.

Moreover, in order to prevent server side connection closures, SlowDroid keeps them alive by periodically sending data. In particular, unlike other slow DoS threats, SlowDroid sends a single byte character at any period.

We have designed the attack to execute three different program flows/threads for connections management:

- *connect flow*: it takes care of connections establishing, without sending any data to the server;
- *maintain flow*: it maintains the connections with the server alive, by slowly sending data to the victim through the established channels, preventing server side connection closures;
- *control flow*: it identifies connections that have been closed by the server.

The three flows share a common variable that includes all active connections.

VI. Conclusions and Future Work

In this paper we have introduced the SlowDroid Denial of Service testing tool. The tool implements an innovative attack which makes use of a tiny amount of bandwidth. Because of this, we have decided to develop SlowDroid on a mobile environment to demonstrate that even a single smartphone with limited capabilities is potentially able to lead a DoS on a corporate server. Since the tool has been implemented to be suited in a mobile operating system, we have also deeply analyzed and described the advantages of a mobile attack execution.

The published SlowDroid attack should be considered as a testing tool for system administrators willing to test the resilience of their servers to such attack. Although the tool is able to lead a DoS on a server, it is particularly easy to protect from a non-distributed menace such as SlowDroid [29]. Nevertheless, many servers on the Internet seems to be affected and unprotected from SlowDroid.

In order to avoid malicious and dangerous activities, we have deliberately reduced the functionalities of the published SlowDroid app: in particular, an extended and more dangerous version of the tool could have been published, in order to affect more server by specifying additional attack paramteters and by implementing a distributed menace. Nevertheless, our purpose is not to deploy a cyberweapon, but to provide a testing tool and to prove that today's smartphones can be used as attack vectors. Because of this, the SlowDroid code has also been obfuscated, in order to hinder decompiling.

Further works on the topic may involve a porting of SlowDroid on different systems. In particular, since the attack has been implemented and is included in a separate Java library, a Java implementation aimed to execute the attack on different operating systems is facilitated.

It is also important to consider extensions needed from compatibility needs: since SlowDroid is bounded to the Android operating system, which is continuously evolving, future Android versions may require amendments to maintain compatibility.

References

[1] S. Coursen, "The future of mobile malware," *Network Security*, vol. 2007, no. 8, pp. 7–11, 2007.

[2] S. Furnell, "Handheld hazards: The rise of malware on mobile devices," *Computer Fraud & Security*, vol. 2005, no. 5, pp. 4–8, 2005.

[3] Y. Zhou, X. Jiang, "Dissecting android malware: Characterization and evolution," *Security and Privacy (SP), 2012 IEEE Symposium on*, pp. 95–109, 2012.

[4] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, J. M. Smith, “Smudge attacks on smartphone touch screens,” *Proceedings of the 4th USENIX conference on Offensive technologies*, pp. 1–7, 2010.

[5] E. Cambiaso, G. Papaleo, M. Aiello, “SlowDroid Denial of Service tool for Android - Available at http://software.netsec.ieiit.cnr.it/projects/slowdroid/,” p. Pages, Date Accessed on April 23, 2013.

[6] E. Cambiaso, G. Papaleo, G. Chiola, M. Aiello, “Slow DoS attacks: definition and categorisation,” *International Journal of Trust Management in Computing and Communications*, vol. 1, no. 3, pp. 300–319, 2013.

[7] A. Kuzmanovic, E. W. Knightly, “Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants,” *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 75–86, 2003.

[8] G. Macia-Fernandez, J. E. Diaz-Verdejo, P. Garcia-Teodoro, “Evaluation of a low-rate DoS attack against iterative servers,” *Computer Networks*, vol. 51, no. 4, pp. 1013–1030, 2007.

[9] Wikipedia, “Slowloris - Available at http://en.wikipedia.org/wiki/Slowloris,” p. Pages, Date Accessed on April 23, 2014.

[10] ha.ckers, “Slowloris HTTP DoS - Available at http://ha.ckers.org/slowloris/,” p. Pages, Date Accessed on April 23, 2014.

[11] SpiderLabs-Anterior, “Mitigation of Apache Range Header DoS Attack - SpiderLabs Anterior - Available at http://blog.spiderlabs.com/2011/08/mitigation-of-apache-range-header-dos-attack.html,” p. Pages, Date Accessed on April 23, 2014.

[12] S. Alam, “Apache released patch for ApacheKiller.pl Range Byte Flaw - Available at http://www.hackersgarage.com/apache-released-patch-for-apachekiller-pl-range-byte-flaw.html,” p. Pages, Date Accessed on April 23, 2014.

[13] Google-Code, “slowhttptest - Application Layer DoS attack simulator - Available at https://code.google.com/p/slowhttptest/,” p. Pages, Date Accessed on April 23, 2014.

[14] S. Shekyan, “Are you ready for slow reading? - Available at https://community.qualys.com/blogs/securitylabs/2012/01/05/slow-read,” p. Pages, Date 2012.

[15] N. Hoque, M. H. Bhuyan, R. Baishya, D. Bhattacharyya, J. Kalita, “Network attacks: Taxonomy, tools and systems,” *Journal of Network and Computer Applications*, 2013.

[16] H. McCracken, “Whos Winning, iOS or Android? All the Numbers, All in One Place - Available at http://techland.time.com/2013/04/16/ios-vs-android/,” p. Pages, Date Accessed on April 23, 2014.

[17] xda-developers, “WifiKill - disable internet for network hoggers - Available at http://forum.xda-developers.com/showthread.php?t=1282900,” p. Pages, Date Accessed on April 23, 2014.

[18] “DroidSheep [ROOT] - Available at http://droidsheep.de/?page_id=263,” p. Pages, Date Accessed on April 23, 2014.

[19] “FaceNiff - Facebook (and other services) Session Hijacker for Android - Available at http://faceniff.ponury.net,” p. Pages, Date Accessed on April 23, 2014.

[20] Google-Play, “LOIC - Low Orbit Ion Cannon - Available at https://play.google.com/store/apps/details?id=genius.mohammad.loic,” p. Pages, Date Accessed on April 23, 2014.

[21] ——, “Loic - Available at https://play.google.com/store/apps/details?id=l.o.i.c,” p. Pages, Date Accessed on April 23, 2014.

[22] ——, “PlexeDOS - LOIC - Available at https://play.google.com/store/apps/details?id=genius.plexe.loic,” p. Pages, Date Accessed on April 23, 2014.

[23] ——, “LOIC - Low Orbit Ion Cannon - Available at https://play.google.com/store/apps/details?id=genius.mustafa.loic,” p. Pages, Date Accessed on April 23, 2014.

[24] Wikipedia, “Low Orbit Ion Cannon - Available at http://en.wikipedia.org/wiki/Low_Orbit_Ion_Cannon,” p. Pages, Date Accessed on April 23, 2014.

[25] Google-Play, “Slowloris - Available at https://play.google.com/store/apps/details?id=com.kanuusan.slowloris,” p. Pages, Date Accessed on April 23, 2014.

[26] Wikipedia, “Vertical handover - Available at http://en.wikipedia.org/wiki/Vertical_handover,” p. Pages, Date Accessed on April 23, 2014.

[27] M. Aiello, G. Papaleo, E. Cambiaso, “SlowReq: A Weapon for Cyberwarfare Operations. Characteristics, Limits, Performance, Remediations,” *International Joint Conference SOCO’13-CISIS’13-ICEUTE’13*, pp. 537–546, 2013.

[28] Y. Shang, W. Luo, S. Xu, “L-hop percolation on networks with arbitrary degree distributions and its applications,” *Physical Review E*, vol. 84, no. 3, p. 031113, 2011.

[29] E. Cambiaso, G. Papaleo, M. Aiello, “SlowDroid Mitigation - Available at http://security.ge.cnr.it/?q=slowdroidmitigation,” p. Pages, Date Accessed on April 23, 2013.