1-12-2008

Framework for Anomaly Detection in OKL4-Linux Based Smartphones

Geh W. Chow
British Telecommunications plc

Andy Jones
British Telecommunications plc

Recommended Citation

Chow, G. W., & Jones, A. (2008). Framework for Anomaly Detection in OKL4-Linux Based Smartphones.
DOI: https://doi.org/10.4225/75/57b55ad8b876b

DOI: 10.4225/75/57b55ad8b876b
6th Australian Information Security Management Conference, Edith Cowan University, Perth, Western Australia, 1st to 3rd December 2006.

A Framework for Anomaly Detection in OKL4-Linux Based Smartphones

Geh Wynn Chow and Andy Jones
British Telecommunications plc

Abstract

Smartphones face the same threats as traditional computers. As long as a device has the capabilities to perform logic processing, the threat of running malicious logic exists. The only difference between security threats on traditional computers versus security threats on smartphones is the challenge to understand the inner workings of the operating system on different hardware processor architectures. To improve upon the security of smartphones, anomaly detection capabilities can be implemented at different functional layers of a smartphone in a coherent manner; instead of just looking at individual functional layers. This paper will focus on identifying conceptual points for measuring normalcy in different functional layers of a smartphone based on OKL4 and LiMo Foundation's platform architecture.

Keywords

Smartphone security, anomaly detection, OKL4, Linux, LiMo Foundation

INTRODUCTION

Mobile phones today have evolved to become ubiquitous communication and computing platforms where users can make the usual phone calls, create presentation slides, edit documents, listen to music, watch videos, play games, and access the internet all from a single mobile device, simply known as a smartphone. These devices are able to perform complex computing tasks and communicate through a variety of different methods (3G/GSM, WiFi, RFID, IrDA, Bluetooth). In short, a smartphone is almost the same as a traditional computer, only smaller in size and computing power.

Smartphones today are produced by many different hardware manufacturers including Nokia, Sony Ericsson, Motorola, Samsung, HTC, RIM Blackberry, Palm and Apple iPhone. These hardware manufacturers partner with different mobile operating system vendors such as Symbian (Nokia, Sony Ericsson), Windows Mobile (HTC), Blackberry OS (Blackberry), PalmOS (Palm), Linux (Motorola, Samsung) and Apple iPhone OS (Apple iPhone). According to Canalys, a technology market research firm's 2007 report on mobile operating systems; Symbian is the leader in mobile operating systems; followed by Linux, Windows Mobile, Blackberry OS and Apple iPhone OS. However, Linux is beginning to establish itself as a major mobile operating system platform thanks to pioneering efforts by industry consortiums such as the LiMo Foundation.

The LiMo Foundation was created in 2007 by Motorola, NEC, NTT DoCoMo, Panasonic Mobile Communications, Samsung, and Vodafone as a mobile industry consortium that focuses on creating an open mobile platform based on the Linux operating system. The motivation of using Linux, from LiMo's perspective, is that it is open source, and also that it enables a standardised framework for faster and easier development of applications in an open environment. By collaborating with different smartphone hardware manufacturers, LiMo intends to build a standard and open Linux operating system stack where developers can create applications in different logical functionalities or frameworks that are typically found in a smartphone.

2007 also saw the emergence of virtualisation in embedded systems focussing on security and stability. One of the key players in this area is OKL4, a virtualisation platform for embedded systems based on the L4 microkernel technology. L4 is a microkernel family designed by Jochen Liedtke that is well-known for its minimality principle and performance. A microkernel can be described as a minimal computer operating system kernel that provides only the mechanisms needed to implement services such as low-level address space management, thread management, and inter-process communication (IPC). Other operating system services such as device driver management, file system management, and network protocol stacks are treated

as user mode services. This operating systems kernel design approach is different from the more popular monolithic kernel used in Linux, where user mode services mentioned above are implemented inside the kernel in supervisor mode. Microkernel advocates state that microkernels are more secure and stable due to the fact that there are less parts of the kernel running in supervisor mode, and different parts of the operating system are compartmentalised as different user mode services to prevent faulty components from bringing the entire system down.

This paper will focus on identifying conceptual points for measuring normalcy in different functional layers of a smartphone based on OKL4 and the LiMo Foundation's platform architecture.

SECURITY THREATS ON SMARTPHONES

Smartphones face the same threats as traditional computers. As long as a device has the capabilities to perform logic processing, the threat that malicious logic will run exists. The only difference between security threats on traditional computers and security threats on smartphones is the challenge of understanding the inner workings of the operating system on different hardware processor architectures (eg. x86, SPARC, MIPS, PowerPC, etc… on traditional computers; ARM, ATOM etc… on smartphones). These threats have already been observed in the wild. For example, a virus has been detected in the wild infecting Symbian-based smartphones. A handful of other viruses have also been detected that were infecting Windows Mobile-based smartphones.

To counter these mobile-based threats, anti-malware companies have developed antivirus software, host-based firewalls and intrusion detection/prevention systems for smartphones. The antivirus and intrusion detection/prevention system monitors the smartphone for known malicious activities by scanning files, memory and network packets for malicious behaviour. Firewalls in smartphones function in a similar manner to hostbased firewalls in traditional computers by controlling IP-based network access such as 3G and WiFi. However, as in the case of traditional computer security, these solutions also have difficulty detecting unknown threats in smartphones. This is due to the fact that antivirus and host-based intrusion detection/prevention systems rely, for the most part on attack signatures (misuse detection) to successfully mitigate threat.

A different approach of anomaly detection for mitigating unknown threats has been explored and proposed by the intrusion detection research community. The holy grail of host-based anomaly detection is achieved when a host is able to formally measure normalcy for every instance in the system, and as a result, would be able to detect and mitigate unknown events or threats that may occur. In reality, the challenges of anomaly detection lie in the identification and measurement of normalcy. Many approaches have been proposed by the research community, such as measuring normalcy in traditional computer system calls using statistical models. In the smartphone space, one approach to the measured of 'normalcy' is based on a user's call, text and location behaviour using neural networks. To improve upon the security of smartphones, anomaly detection capabilities can be implemented at different functional layers of a smartphone in a coherent manner; instead of just looking at individual functional layers.

ANATOMY OF A SMARTPHONE

A generic smartphone can be divided into the following functional layers as shown in the diagram below.

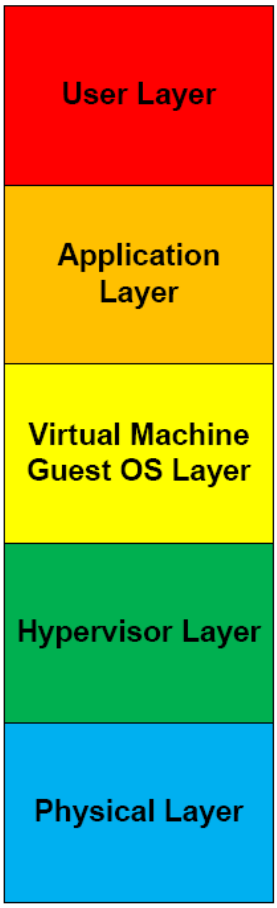

Figure 1 – Smartphone Functional Layers

Physical Layer

The physical layer corresponds to the smartphone's hardware. The common hardware components of smartphones today are shown below:

1. Processor (Central Processing Unit)
2. Power Management Unit
3. Flash Memory (non-volatile memory that can be erased and reprogrammed)
4. SDRAM
5. GSM/GPRS/3G Modem
6. Digital and Analog Baseband
7. Radio Frequency Transceiver
8. GPS Chipset
9. Accelerometer
10. Graphic Accelerator (Graphical Processing Unit)
11. External Storage (microSD Card)
12. LCD Touch Screen Module
13. Bluetooth Module
14. WiFi Module
15. Vibrator

16. USB Device
17. I2C Device
18. Audio System
19. Microphone System
20. Camera

Hypervisor Layer

The hypervisor layer can be thought of as an interface between the hardware layer and the operating system layer. This layer plays the role of separating the operating system and preventing it from having total privileges to access the smartphone hardware for stability and security reasons. In other words, the hypervisor layer is like a virtual machine manager, managing different operating systems installed on top of it with each performing very specific tasks. For example, real-time capabilities are needed for functions that need predictable and deterministic behaviour such as making cellular calls. A real-time operating system can be installed on top of the hypervisor layer to perform this task. Functions that do not require real-time capabilities such as 3rd party applications can be serviced by another operating system specialised in application management. Separating certain functionalities of the phone into different layers as explained above would enhance stability and security. An unstable 3rd party application would not be able to crash the entire smartphone because the application manager operating system is separate from other operating systems that perform more critical smartphone functions. This makes it possible for the smartphone to continue initiating and receiving calls instead of being frozen as the result of a crashed application. From a security perspective, a compromised application in a phone would not constitute a total compromise of a smartphone due to the separation of duties and privileges between different operating systems. OKL4 is an example of a hypervisor layer for mobile devices and is based on the L4 microkernel family design.

Virtual Machine Guest OS Layer

The virtual machine guest OS layer may consist of one or many different operating systems paravirtualised to run on top of the hypervisor layer. A paravirtualised operating system is described as an operating system that is intentionally reprogrammed to run in a hypervisor environment; where the kernel is customised to interface hardware instructions with the hypervisor application programming interface (API) instead of directly to the hardware. The hardware abstraction is important, because this will enable the operating system to run with lower privileges, and supervisor instructions will be passed to the hypervisor to be translated to actual hardware instructions. For example, OLK4 has developed OK Linux, a paravirtualised Linux kernel that exists as a virtual guest operating system sitting on top of the OKL4 hypervisor layer. Inside the paravirtualised operating system layer, memory management, file system management, process management, inter-process communications, interrupt handling, device driver management and network management are passed to the hypervisor layer to be handled and translated into real hardware instructions.

Application Layer

For the application layer, LiMo Foundation's platform architecture is used to describe the application functionalities inside a Linux mobile operating system for smartphones. LiMo's platform consists of the following logical functionalities:

1. Application Manager Framework
   Responsible for managing application installations and invocations.
2. Application User Interface Framework
   Responsible for defining the graphical user interface of the applications.
3. Registry
   Responsible for storing and managing access to application related data/values.
4. Conflict Management
   Responsible for resolving conflicts arising from concurrent requests to shared resources. This functionality may be offloaded to the hypervisor layer.
5. Event Delivery / Inter Process Communications (IPC)
   Event delivery is responsible for generating, processing, filtering, subscribing and publicising application related notifications. IPC is responsible for application service startup, service

deployment configuration, service discovery by clients, communication between clients and servers, parameter passing between clients and servers, and service shutdown. This functionality may be offloaded to the hypervisor layer.

6. Security Framework
   Responsible for scanning new applications to determine their security trust level before installation, and
   also applying security access control policies in the application.
7. Telephony Framework
   Responsible for cellular activities such as SIM handling, network registration, voice and video calls. This functionality may be offloaded to the hypervisor layer.
8. Networking Framework
   Responsible for handling IP networking functionality. This functionality may be offloaded to the hypervisor layer.
9. Messaging Framework
   Responsible for handling Short Message Service (SMS) and Multimedia Message Service (MMS) functionalities.
10. Multimedia Framework
    Responsible for managing multimedia services such as audio, image, and video services.
11. Digital Rights Management Framework
    Responsible for controlling access to licensed digital media using cryptographic functions.
12. Database
    Responsible for providing database functionalities to applications for storing data.
13. Other Frameworks
    LiMo provides a modular design for future frameworks to be integrated into the platform architecture. Other possibilities may include a GPS framework or even an accelerometer framework.

User Layer

The user layer is more of an abstract layer, used to describe the usage pattern of a smartphone by an individual. The "who, what, where, when, how and why" of a user can be profiled through this layer. For example, a user may only contact a few phone numbers frequently who may either be family, close friends, colleagues or business partners. This user may normally contact this particular group of people using voice calls and SMS, mostly in the morning on weekdays. The user may be constantly on the move, so he or she may use his or her smartphone regularly to check emails and surf the web for news. As we can see in the user layer, a profile of a user can be developed based on the smartphone's usage pattern.

AN OVERVIEW OF ANOMALY DETECTION

According to Roy A Maxion et al. (2002), an anomaly is an event (or object) that differs from some standard or reference event, in excess of some threshold, in accordance with some similarity or distance metric on the event. They further described two types of data that can be used for anomaly detection. The first type, called the intuitive type, is where data is numerical and can be mathematically calculated to define anomalies. For example, a threshold value is set on a network anomaly detector to flag an alarm if the monitored network traffic exceeds more than 50 megabits per second. This type of anomaly is intuitive because a threshold value can be defined numerically. The second type, called the categorical type, is where data does not have any numerical or mathematical representation that can be measured. For instance, a police surveillance camera may flag an alarm if the system detects a person that fits a profile of a suspected criminal based on parameters such as suspicious behaviour, illogical commuting patterns and other related factors. Hence, an anomaly detector may employ either an intuitive type or categorical type of approach depending on the event or subject being monitored.

Once the data's type of anomaly (intuitive or categorical) is determined, different types of mathematical models can be applied to the data for further processing. Intuitive types are usually applied with statistical models to identify normalcy. Statistical models such as Bayesian theory and logistic regression can be used by an anomaly detection system to learn the behaviour of a monitored system based on numerical data. Categorical types are usually applied with models such as Markov model and artificial neural networks for learning behaviours based on different states of a monitored system. Identifying the types of mathematical

models used for anomaly detection is beyond the scope of this paper; hence the focus will only be on identifying different points for measuring normalcy in different functional layers of a smartphone based on OKL4 and LiMo Foundation's platform architecture.

APPLYING ANOMALY DETECTION CAPABILITIES AT DIFFERENT SMARTPHONE LAYERS

Many smartphone anomaly detection papers propose the learning of a user's behaviour when using a smartphone. This paper aims to improve on earlier proposals through the concept of implementing anomaly detection capabilities at different layers in a smartphone, instead of just the user layer. This includes the hypervisor layer, operating system layer, application layer and the user layer. The key points in implementing anomaly detection in these layers is to identify what to measure and define what is normal in the measured entity. Once these capabilities are in place, anomalies can be detected and actions can be taken to raise an exception or perform mitigation procedures.

Anomaly Detection in the Hypervisor Layer

The following explores possible entities that can be measured for normalcy at the hypervisor layer, based on the OKL4 microkernel. OKL4's application programming interfaces (API) are made up of the following logical elements:

1. Address spaces and threads
    - System level instructions and data usually live in a protected area in memory. Since certain system level instructions and data does not change unless the hypervisor firmware is modified, normalcy can be defined by learning where and how frequently these instructions and data are stored in memory. Through statistical methods, it is possible to build a normalcy profile of this element and detect anomalies.
2. Data types, data constructors and data fields
    - This element is used to characterise objects stored in memory or virtual registers. Since objects are defined with their own data types, normalcy can be defined and measured here by learning the data types of these objects.
3. System parameters
    - This element is an OKL4 API used for simplifying the implementation of OKL4 on specific architectures. OKL4 API parameters are constants, and its values are specified by each OKL4 architecture. Normalcy can be defined here since any changes to these constants would be considered an anomaly.
4. Virtual registers
    - Each thread in the hypervisor is associated with a number of virtual registers that are defined by the OKL4 microkernel. Normalcy can be defined and measured by learning the virtual registers of each thread; hence anomalies can be flagged when the thread is associated with the wrong virtual register.
5. System calls
    - This element is conceptually similar to system calls in traditional computer operating systems. Hence, normalcy can be measured using known methods of tracing and learning the system calls of different processes. Once normalcy profiles for the system call sequences of processes are built, anomalies can be detected when the system call sequence is different from the ones measured in its normalcy profile.
6. Communication protocols
    - This element concerns the inter process communications (IPC) in the microkernel. IPC behaviour between processes can be learnt, and a normalcy profile can be built to detect anomalies.

7. IP Networking
    - Normalcy of network packets traversing the hypervisor can be measured using known protocol analysis techniques. For example, a TCP packet with the SYN/FIN flag turned on is considered anomalous, and an anomaly flagged.

Anomaly Detection in the Virtual Machine Guest OS Layer

Normalcy at the virtual machine guest operating system layer can be implemented based on monitoring the paravirtualised system calls between the guest operating system and hypervisor. In OKL4, many different instances of a paravirtualised operating system run at the same time offering different services such as file system management, networking and graphical user interfaces. Each of these separate instances would perform its own system calls via the hypervisor. These system calls usually contain instructions intended for the actual hardware (via the hypervisor), hence they can be used as categorical data for an anomaly detection system to learn the behaviour of normal paravirtualised system calls. Identifying all the different system calls for monitoring normalcy is beyond the scope of this paper; however the key purpose is to illustrate the possibilities of performing anomaly detection by monitoring system calls between the paravirtualised operating system and the hypervisor.

Anomaly Detection in the Application Layer

Normalcy at the application layer can be implemented based on the LiMo Foundation's platform architecture as shown below:

1. Application Manager Framework
    - A list of installed applications can be used as a baseline for normalcy. Any tampering or illegal installation of applications will flag off an anomaly.
2. Application User Interface Framework
    - Normalcy profiling in this framework is not necessary.
3. Registry
    - A list of authorised registry entries can be used as a baseline for normalcy. Any illegal modifications to the registry will flag as an anomaly.
4. Conflict Management
    - Normalcy profiling in this framework is not necessary.
5. Event Delivery / Inter Process Communications (IPC)
    - Event delivery notifications and IPC behaviour can be learnt to establish a normalcy profile. Any deviations from the learnt behaviour will flag as an anomaly.
6. Security Framework
    - The integrity of this framework is important. If it is compromised, then the integrity of all other applications installed will be suspect. Hence, a normalcy profile for the health of this framework can be learnt. Any deviations of the normalcy profile of this framework will flag as an anomaly.
7. Telephony Framework
    - This normalcy profiling of this framework can be done concurrently with the user layer to learn the behaviour of the user's cellular usage.
8. Networking Framework
    - Normalcy profiling in this framework is not necessary as it can be done at the hypervisor layer.
9. Messaging Framework
    - This normalcy profiling of this framework can be done concurrently with the user layer to learn the behaviour of the user's messaging usage.

10. Multimedia Framework
    - Normalcy profiling in this framework is not necessary.
11. Digital Rights Management Framework
    - Normalcy profiling in this framework is not necessary.
12. Database
    - Normalcy profiling for this framework would require learning of which application is authorised to insert, update and delete data in the database.

Anomaly Detection in the User Layer

Normalcy at the user layer can be implemented based on the characteristics of an individual. An example of the types of characteristics that can be monitored are shown below:

1. Subscriber Identity Module (SIM) password
    - The SIM password can be monitored in the smartphone every time a user keys it in. If the smartphone is used with another foreign SIM with a different password, the anomaly detector can flag of an alarm.
2. Phone password
    - Phone passwords can be monitored in the smartphone. Since these types of passwords are usually static for a long time, a change to this password can also be perceived as an anomaly.
3. Key stroke analysis
    - Different individuals have their own unique way of using their smartphones, and this can be used as a means of learning the behaviour of a user based on their keystrokes. If the phone is used by somebody other than the original owner whose keystrokes have been learned, an anomaly can be flagged off.
4. T9 usage analysis
    - The T9 predictive text dictionary found on most smartphones today are widely used as a means of making it easier and faster to type text messages on a 9-key keypad; hence the name T9 (Text on 9 Keys). T9 has the ability for users to key in their own frequently words that are not found in the built-in dictionary; hence this can be used as a parameter for monitoring anomalies when the smartphone is texted by a foreign user.
5. Text Message Spelling analysis
    - Different individuals may text using their own style of spelling for certain words. This can be used a parameter for monitoring anomalies when the smartphone is texted by a foreign user.
6. Top-n called numbers/contacts based on hour, day, week, month
    - Top-n number of calls to a certain number/contact based on different time scales can be used as a parameter for learning the user's smartphone usage behaviour.
7. Top-n texted numbers/contacts based on hour, day, week, month
    - Top-n number of texts to a certain number/contact based on different time scales can be used as a parameter for learning the user's smartphone usage behaviour.
8. Top-n received call numbers/contacts based on hour, day, week, month
    - Top-n number of received calls from a certain number/contact based on different time scales can be used as a parameter for learning the user's smartphone usage behaviour.
9. Top-n received text numbers/contacts based on hour, day, week, month
    - Top-n number of received texts from a certain number/contact based on different time scales can be used as a parameter for learning the user's smartphone usage behaviour.

10. Top-n called numbers/contacts based on location
    - Top-n number of calls to a certain number/contact based on location can be used as a parameter for learning the user's smartphone usage behaviour.
11. Top-n texted numbers/contacts based on location
    - Top-n texts to a certain number/contact based on location can be used as a parameter for learning the user's smartphone usage behaviour.
12. Top-n received call numbers/contacts based on location
    - Top-n number of received calls from a certain number/contact based on location can be used as a parameter for learning the user's smartphone usage behaviour.
13. Top-n received text numbers/contacts based on location
    - Top-n number of received texts from a certain number/contact based on location can be used as a parameter for learning the user's smartphone usage behaviour.
14. Top-n called numbers/contacts based on call duration
    - Top-n number of calls to a certain number/contact based on call duration can be used as a parameter for learning the user's smartphone usage behaviour.
15. Top-n texted numbers/contacts based on call duration
    - Top-n number of texts to a certain number/contact based on call duration can be used as a parameter for learning the user's smartphone usage behaviour.
16. Top-n received call numbers/contacts based on call duration
    - Top-n number of received calls from a certain number/contact based on call duration can be used as a parameter for learning the user's smartphone usage behaviour.
17. Top-n received text numbers/contacts based on call duration
    - Top-n number of received texts from a certain number/contact based on call duration can be used as a parameter for learning the user's smartphone usage behaviour.
18. Frequently executed smartphone applications
    - Frequently executed smartphone applications can be used as a parameter for learning the user's smartphone usage behaviour. For example, a user may always like to play a certain smartphone game, access the camera, view pictures via a certain smartphone image viewer and other similar actions.
19. Smartphone application usage analysis
    - The manner in which a particular smartphone application is used can be used as a parameter for learning the user's smartphone usage behaviour. For example, the user may like to tweak the smartphone camera to a certain setting before snapping photos.
20. Bluetooth usage analysis
    - The most frequent devices paired via Bluetooth to the smartphone can be used as a means of learning the smartphone's Bluetooth connectivity behaviour. If there are rogue Bluetooth applications that intend to pair with the smartphone monitored by an anomaly detection system, the smartphone can flag an alarm.
21. WiFi usage analysis
    - The most frequent access points connected by the smartphone can be used as a means of learning the smartphone's WiFi connectivity behaviour. If there are rogue WiFi access points attempting to associate with the smartphone monitored by an anomaly detection system, the smartphone can flag an alarm.

Anomaly Detection Challenges in Smartphones

Anomaly detection at different functional layers in a smartphone can help enhance the accuracy of determining normalcy and detecting anomalies in a smartphone. However, the smartphone is still limited by processing power, battery power and storage and this may impede the ideal implementation of anomaly detection in embedded systems. Several proposals have been put forward, such as Mark Burgess (2005); where processing anomaly detection data can be distributed to other similar devices. Advances in making smartphone processorsfaster and the ability to have longer battery life and larger solid state device storage are other solutions at the hardware level that may make anomaly detection in smartphones a reality.

CONCLUSION

Anomaly detection is a complimentary security solution for detecting unknown threats to smartphones, and in order to enhance its ability to determine normalcy and detecting anomalies, it should be implemented at a numberof different functional layers to obtain a more fine-grained view of how each layer works. This paper proposes4different functional layers (hypervisor, virtual guest operating system, application, user) in a smartphone whereanomaly detection can be implemented; and also identified each individual layer's high level parameters where normalcy can be monitored.

REFERENCES

Azzedine Boukerche et al. (2002) Behavior-Based Intrusion Detection in Mobile Phone Systems, Journal of Parallel and Distributed Computing 62, 1476–1490 (2002)

Cyrus Peikari et al. (2004) Details Emerge on the First Windows Mobile Virus, URL http://www.informit.com/articles/article.aspx?p=337069 Accessed 12 Jun 2008

Darren Mutz et al. (2006) Anomalous system call detection, ACM Transactions on Information and System Security (TISSEC) Volume 9 , Issue 1 (February 2006)

F-Secure Corporation, URL http://www.f-secure.com/v-descs/commwarrior.shtml, Accessed 5 Aug 2008

Gernot Heiser (2007), Virtualization for Embedded Systems, Open Kernel Labs Technology White Paper

Jochen Liedtke (1995), On µ-Kernel Construction, Proceedings of the 15th ACM Symposium on Operating System Principles (SOSP), Copper Mountain Resort, CO, December 1995

LiMo Foundation, URL http://www.limofoundation.org/, Accessed 3 Jun 2008

Mark Burgess (2005), Probabilistic anomaly detection in distributed computer networks, Oslo University College (2005)

N.L. Clarke et al. (2006) Authenticating mobile phone users using keystroke analysis, Source International

Journal of Information Security Volume 6 , Issue 1, December 2006

Open Kernel Labs Community Wiki, URL http://wiki.ok-labs.com/ Accessed 12 Jun 2008

Open Kernel Labs, OKL4 Microkernel Reference Manual API Version 0316 (2008)

Roy A Maxion et al. (2002), Anomaly Detection in Embedded Systems, IEEE Transactions on Computers, Vol.51, No.2, February 2002

COPYRIGHT