Security Analysis of the i>clicker Audience Response System

December 7, 2010

Derek Gourlay, Yik Lam Sit, Yuan Sunarto, Tim Wang

Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada

<derekgourlay@gmail.com, cyber_dragon85@hotmail.com, yuan_hidris@hotmail.com, twji1010@gmail.com>

Abstract – This paper analyzes the security of the i>clicker student response system: a system which enables interactive learning that is widely used in North American schools. This paper uncovers potential security vulnerabilities of the system and shows how they lead to significant exploits involving disruption, disclosure, and impersonation attacks. In addition to identifying potential vulnerabilities and exploits, this paper outlines possible solution sand ways in which to mitigate the risks of assets at stake when using the i>clicker system.

I. Introduction

The i>clicker audience response system enables an interactive learning environment in which students can instantly provide feedback and answer questions posed to them by instructors during lectures. According to the i>clicker corporate website: 679 post-secondary schools, 160 K-12 schools, 72 corporations in North America are currently using i>clicker [1]. At the same time, a brief internet search regarding security analyses of the i>clicker system returns very few relevant results. As the i>clicker can be used for answering everything from marked questions affecting a student's grade to anonymous polling sessions – the importance of security and accuracy in the i>clicker system becomes quite an apparent and pertinent issue to not only the numerous current users but also potential future adopters.

The security analysis of the i>clicker (also stylized i>clicker) outlined within this paper builds upon previous efforts in reverse engineering RF based clicker response systems. Our analysis shows the i>clicker system is susceptible to exploits involving service disruption, disclosure of private information, and impersonation attacks. In the following sections we describe the i>clicker system, outline discovered vulnerabilities, detail methodologies and steps taken to exploit these vulnerabilities, present possible solutions, and offer suggestions as to how to the i>clicker system can be used such that assets at risk are minimized.

II. Analyzed System

The i>clicker audience response system consists of a set of RF remotes used by students to enter responses and an RF base station used by instructors to capture and record responses. A single unique ID stored on each remote allows the base station to distinguish the source of a given response. To avoid interference between base stations when multiple units are used in close proximity - the i>clicker system allows the use of 16 different channels. Instructors and students configure their hardware to a specific channel by means of a pairing process.

Typically, the i>clicker system is used alongside a course management system (CMS) such as Blackboard, Moodle, or Sakai. Students register their i>clickers' unique ID on the CMS to bind their clickers to their CMS accounts. Instructor can then upload polling session data to the CMS and link responses to a specific student's grade set.

Assets placed at risk by incorporating the i>clicker system in a classroom (and potentially as part of a course marking scheme) are primarily the CIA (Confidentiality, Integrity, and Availability) properties of student responses (and their corresponding grades).

III. Related Work

During the course of this project we found one instance online of others attempting to reverse engineer the i>clicker. The individuals managed to disassemble the i>clicker remote and hook it up to their PC. Using software tools they were able to successfully dump the memory contents and attempted to analyze parts of the code (tracing it by hand). This information provided us with an initial starting point to build upon to in order achieve further exploits of the system. Our

work differs in a few matters: the first being that the other individual's work was not verifiable as they did not present a working demo of any potentially developed exploits. As well information obtained during our analysis seemed to differ from theirs. Furthermore our analysis of the i>clicker highlights topics of concern with respect to fundamental security principles and presents possible solutions to vulnerabilities found.

IV. Discovered Vulnerabilities

Three major categories of vulnerabilities were discovered within the i>clicker audience response system, exploitation of which could lead to:

- Disclosure of private information
- Impersonation (ID spoofing)
- Service disruption

A. Disclosure of Private Information

Responses sent by students are often considered to be confidential (even more so when utilizing the "anonymous polling" feature built into the i>clicker software suite). Our experiments showed, however, that a base station tuned to the same frequency as another base station will indiscriminately read responses sent over the air. This appears to be a result of the fact that the pairing sequence between a clicker and given base station does not include any means of authentication. While at the time of development of the i>clicker it may have been assumed that students would not have explicit access to their own base station, a common practice amongst designers of secure systems is to continuously *Question Assumption*s: during which one should re-examining all assumptions made about threat agents and the environment of a system. During the course of this project acquiring an i>clicker base station was as trivial as sending a few polite emails to our university's center for learning technologies.

However, direct access to an i>clicker base station is not necessarily needed to intercept and decode i>clicker wireless traffic. The i>clicker operates at approximately 915MHz. The specific frequencies for each particular channel are outlined in Table I – Base Station Frequencies [2]. Upon disassembly of an i>clicker remote it was discovered that a common wireless module, the Semtec XE1203F ISM band RF

TABLE I
BASE STATION FREQUENCIES

Channel	Frequency	Channel	Frequency
AA	917.0 MHz	CA	922.0 MHz
AB	913.0 MHz	CB	923.0 MHz
AC	914.0 MHz	CC	907.0 MHz
AD	915.0 MHz	CD	908.0 MHz
BA	916.0 MHz	DA	905.5 MHz
BB	919.0 MHz	DB	909.0 MHz
BC	920.0 MHz	DC	911.0 MHz
BD	921.0 MHz	DD	910.0 MHz

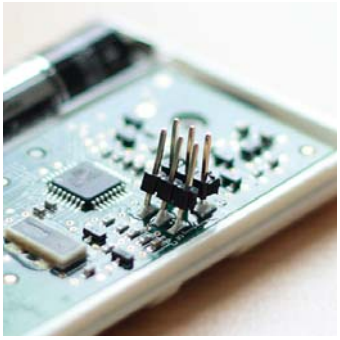

Fig. 1. Header pins soldered to ISP port

transceiver, was used to enable RF communication. It is theorized that one could develop a device based upon the XE1203F module that could intercept and decode wireless i>clicker traffic. The process of reverse engineering i>clicker wireless transmissions is not further addressed during the course of this analysis; however, as a general guide it is offered that one could attempt to look for a known clicker ID accounting for obfuscations before transmission outlined in Section C – Service Disruption.

B. Impersonation (ID Spoofing)

Each i>clicker remote is identified by a unique 4-byte ID. It was our initial assumption that this ID would be stored within a non-volatile memory location of the remote. Upon disassembling a remote this assumption was confirmed as it was discovered it is possible to use an AVR Mk-II, an in-system programmer (ISP), to extract content from the onboard ATMega8 microcontroller as well as to reprogram it. This was all possible because the microcontroller's feature that prevents dumping program memory and EEPROM content, called the lock bits, had not been enabled. The process of connecting our programmer was eased by the fact that the ISP port to the onboard microprocessor was directly accessible via the i>clicker PCB (to which we soldered 6 header pins as shown in Fig. 1). The first 3 bytes of the remote's unique 4 byte ID was found stored at address 0x00 to 0x02 in the EEPROM (as illustrated in Fig. 2). The 4th byte of any given ID is directly calculated by taking the XOR of the first 3 bytes. Modifying the 3 byte value stored in EEPROM changes the ID a remote transmits to a base station. The process of impersonating another remote, hereby referred to as ID spoofing, is implemented by writing a new set of 3 bytes to the EEPROM corresponding to the remote to be copied.

The ability to spoof another remote's ID has two major consequences. The first consequence being that one could override and change any given answers by another remote, thereby reducing the integrity of any collected responses. A second more serious consequence is the fact that ID spoofing allows an attacker to hijack control of an i>clicker session. The i>clicker software suite allows a single remote to be designated as an instructor remote – allowing this remote to be used to start / stop polling sessions, hide /

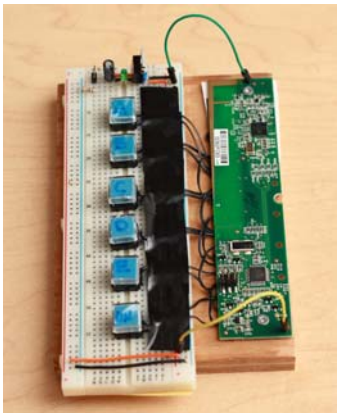

Fig. 3. i>clicker Development Platform

display answers, and even control navigation of PowerPoint presentation slides. Given the ID of the instructor remote an individual could reprogram their remote to duplicate it and acquire the same functionality. Acquisition of an instructor's remote ID can be achieved in one of two fashions: with physical access to the remote one simply needs to read the ID off the attached labeling, without physical access it is possible to use a secondary base station to record any wireless i>clicker traffic (as highlighted in Section A) from which the ID can be located.

V. Service Disruption

Disruption of the i>clicker polling service was accomplished by means of flooding a polling session with fake clicker responses. The i>clicker receiver can "process up to 1500 votes and accepts up to 750 per second" [3]. By modifying the firmware upon a remote it was possible to quickly exceed this limit. In order to achieve this it was necessary, to an extent, to reverse engineer the i>clicker firmware.

The reverse engineering process was accomplished by initially developing an i>clicker development platform to work upon (as depicted in Fig. 3). By means of our AVR MK-II programmer we were able to dump the program memory contents of the i>clicker remote to a hex file which was then decompiled into assembly using the freely available AVR Studio software suite. With use of common electronic bench

INTEL HEX FILE FORMAT

:10000000**0D92FC**00000000000000000000000000055

Fig. 2. Sample Hex File Dump

top tools, such as multimeters and digital logic probes, we were able to determine that the buttons of an i>clicker remote are attached to PINC of the onboard ATMega8 microcontroller.

In order to achieve the goal of simulating fake clicker resposnes it was necessary to locate two critical code segments (amongst the roughly 2000 lines of assembly acquired from decompilation). First it was necessary to locate the code segment in which button presses of the remote are registered and read within the software. Modifying this code would allow us to simulate button presses upon the remote. The second critical code segment to be located was the point at which a clicker's ID is read from the known location in EEPROM memory. By referencing the microcontroller's datasheet [4] we were able to locate the memory address of pertinent special function registers – namely the PINC data register and the EEPROM data register (EEDR). By searching for these memory addresses in the acquired assembly code and tracing through code execution using the AVR studio debugging tools, we were able to locate both of the desired critical code segments.

With the critical code segments located, we began to modify the i>clicker's firmware by writing a series of patches to inject code that implements our desired functionality. We modified the firmware in a fashion such that when a button is held down the subroutine responsible for reading button presses alternates between registering a button press and a button release. While at first our attempts to simulate rapid button presses was unsuccessful it was later determined that the i>clicker firmware implments a form of software debouncing explicitly to avoid registering multiple button presses mistakenly). The debouncing subroutines of the firmware appeared to utilize timer 0 of the ATMega microcontroller for timing (as did the flashing LED indicators upon the remote). By adjusting the timer0 prescaler we were able to significantly increase the rate at which button presses would be registered.

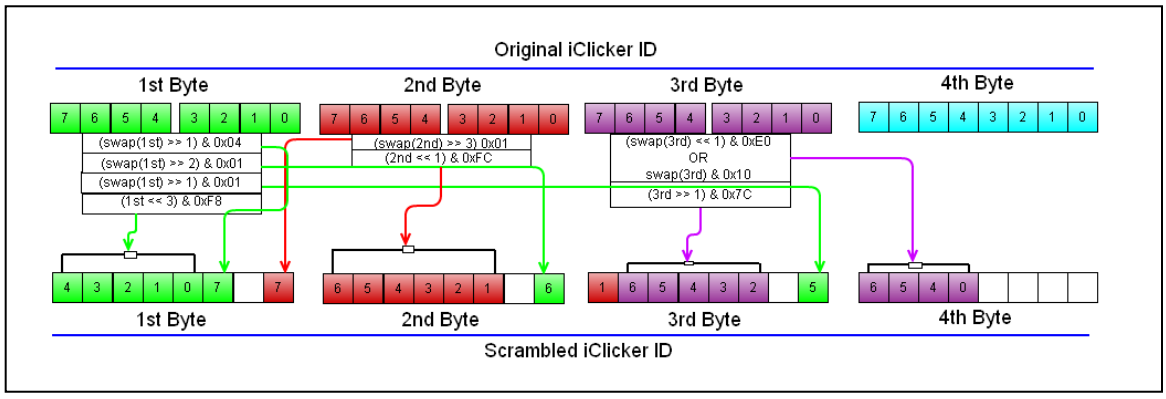

Fig. 4. i>clicker Remote ID Obfuscation

With the ability to rapidly simulate button presses upon a remote the only remaining factor required to flood a system with fake responses was the ability to generate a new remote ID upon each answer sent. By tracing through the located subroutines that loaded a clicker's unique ID from the EEPROM we were able to locate the address in program memory at which each byte of the ID was stored (and later read from upon transmitting a response). Generating a new ID was as simple as incrementing the byte values stored at each of these addresses each time a response was to be transmitted.

As mentioned previously the first 3 bytes of an i>clicker's ID XOR to produce the 4th byte. Initially we were concerned that by simply incrememting the byte values we would not produce a valid ID. This did not appear to be a problem, however, as all responses transmited were sucessfully received by an i>clicker base station. Investigation into this matter lead to the discovery that a remote's ID is transformed through an obfustcation process before transmission (as outlined in Fig. 4). Furthermore, the 4th byte of an original i>clicker remote's ID is never used. Our only conclusion as to why this process occurs was an attempt at *Security through Obscurity*. Discovery of this process lead to the ability to not only be able to transmit responses from randomly generated IDs but also to programatically specify the IDs to use. As an aside, we were also able to modify the firmware to allow one to change the ID a remote transmits with each answer on the fly by entering a new ID one nibble at a time into the buttons of the i>clicker.

VI. Presented Solutions

The main assets placed at risk when using the i>clicker audience response system, as mentioned previously, are the CIA properties of student responses and their associated grades. In order to mitigate these risks we present a series of possible solutions for the i>clicker system and ways in which it should be used:

As shown, the i>clicker system lacks the security mechanisms required to ensure the authenticity of a message. The system is vulnerable to deception attacks as a result. In our experiments we were able to deceive the i>clicker base station in regards to the origin of a response by falsifying a remote's ID, this lead to ID spoofing attacks as well as being able to flood a base station unit with fake responses. One possible solution to this problem is to obtain a list of remotes that are expected to participate in a poll beforehand. This can be easily done when the i>clicker system is used with a CMS. Another benefit of registering the clickers before a poll is that the system no longer needs to be concerned with interference with another polling session using the same frequency because each base station knows its intended participants. The system could also implement a one-way authentication scheme to prevent spoofed responses. A possible protocol is as follows:

Each remote has an embedded asymmetric crypto-processor. Students first register the public keys of their remotes in their CMS account. Before a polling session, the base station obtains a list of the participants' public keys. During the poll it broadcasts challenges, which the remotes use to generate responses containing the concatenation of the answer and the challenge, signed by the private key inside the remotes. To prevent traffic eavesdropping and allow secure response acknowledgement from the base station, it is then necessary to implement two-way authentication. To do that, the remotes need to obtain the public key of the base station before a poll. This could be achieved through a process in which students must attach their remote to a base station unit at the start of each semester. This process also has the added benefit in that a professor may revocate a certificate suspected of being compromised (requiring students to obtain a new certificate thereafter).

As many of the findings of this analysis were made possible by the ability to dump program memory and EEPROM contents of an i>clicker remote it is suggested that physical tamper resistance mechanisms be put in place as well. The simplest mechanism would be to secure the firmware in remotes by disabling the extraction of code and data. This could be achieved by setting the lock bits after programming each remote. Adopting a *Defense in Layers* approach, attempts to reverse engineer proprietary firmware could be hampered by setting circuitry and onboard components of a remote in an epoxy-based potting compound to seal against tampering attempts as well.

It is noted that the above proposed counter measures may impact the ease of use of the i>clicker system. Instructors would need to obtain a list of remotes before a poll. As well the cost of the system may also be affected by including embedded crypto-processors and utilizing epoxy based compounds as tamper resistance mechanisms. As a result we acknowledge that the proposed approaches may not be economically or even socially practical to implement. In this case, the users of the i>clicker audience response system should be made aware of the limitation of the system and adjust their use accordingly.

As a general guiding principle, if the confidentiality of student responses is valued it is suggested that institutions tightly regulate and restrict the distribution of base station units. As outlined previously, failure to do so can result in students being able to easily intercept and record wireless i>clicker traffic with ease. Furthermore, it is suggested that the i>clicker, with its current security flaws, is sufficient to encourage attendance, class participation and active learning. Instructors should realize, however, that it is not wise to use the i>clicker for an exam or pop quiz, for example, in which the integrity of responses directly affects a student's grade.

VII. Conclusion

In this study, we analyzed the operation of the i>clicker audience response system and determined that the RF communication used by the system is inherently insecure. We were able develop working exploits that lead to disclosure of private information, impersonation (ID spoofing), and service disruption. The findings in this paper are significant due to the fact that the i>clicker is a widely deployed system and its users often incorrectly assume that its communication are free from attacks. The proposed solution to highlighted problems included the adoption of embedded crypto-processors, implementation of tamper resistance mechanisms, locking access to proprietary firmware by means of microcontroller lock bits, implementation of an authentication scheme, and educating users in how to safely use the i>clicker system as to mitigate risks to assets. The findings within this paper could very well apply to similar audience response products.

VIII. References

[1] **i>clicker.** Who Uses the i>clicker? *i>clicker Corporate Web Site.* [Online] [Cited: 12 7, 2010.] http://www.i>clicker.com/dnn/Abouti>clicker/WhoUsesi>clicker/tabid/147/Default.aspx.

[2]. **Kramer, Doug.** *EMC Test Report.* Lincoln : Nebraska Center for Excellence in Electronics, 2006. p. 6, FCC Test Report. Test Report No. R032806-01-01.

[3] FAQs. *i>clicker Corporate Website.* [Online] [Cited: 12 06, 2010.] http://www.i>clicker.com/dnn/Support/FAQs/tabid/179/Default.aspx.

[4] **Atmel.** Atmega8/L Datasheet. [Online] [Cited: 12 6, 2010.] http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf. Rev.2486X–AVR–06/10.