EnGarde: Protecting the mobile phone from malicious NFC interactions

J. Gummeson[1], B. Priyantha[2], D. Ganesan[1], D. Thrasher[1], P. Zhang[1]
[1]University of Massachusetts Amherst, [2]Microsoft Research
{gummeson, dganesan, thrasher, pyzhang}@cs.umass.edu, bodhip@microsoft.com

ABSTRACT

Near Field Communication (NFC) on mobile phones presents new opportunities and threats. While NFC is radically changing how we pay for merchandise, it opens a pandora's box of ways in which it may be misused by unscrupulous individuals. This could include malicious NFC tags that seek to compromise a mobile phone, malicious readers that try to generate fake mobile payment transactions or steal valuable financial information, and others. In this work, we look at how to protect mobile phones from these threats while not being vulnerable to them. We design a small form-factor "patch", *EnGarde*, that can be stuck on the back of a phone to provide the capability to jam malicious interactions. *EnGarde* is entirely passive and harvests power through the same NFC source that it guards, which makes our hardware design minimalist and facilitating eventual integration with a phone. We tackle key technical challenges in this design including operating across a range of NFC protocols, jamming at extremely low power, harvesting sufficient power for perpetual operation while having minimal impact on the phone's battery, designing an intelligent jammer that blocks only when specific blacklisted behavior is detected, and importantly, the ability to do all this without compromising user experience when the phone interacts with a legitimate external NFC device.

1. INTRODUCTION

Near Field Communication (NFC) has begun to make its way into major mobile phones, with several Android, Blackberry, and Nokia phones already providing such functionality. The proliferation of NFC on phones can open up a range of applications, from being able to interact with NFC-tagged smart posters to revolutionizing the payment industry, where phones are expected to replace credit cards as the most convenient way to pay for products at the point-of-sale.

These benefits of NFC come at at price — security becomes particularly challenging since phones are general-purpose computing devices that expose a relatively large attack surface that can be exploited by unscrupulous individuals. These issues were exposed in a recent security breach that leveraged the fact that NFC tags can be registered to open applications on a phone such as images, contacts, or web pages *without requiring user assent*. In this attack, a mobile phone was directed to a URL that hosted code that exploited a vulnerability in Android 4.1's web browser [10].

In addition to technical issues, there are also non-technical challenges at play — NFC mobile payments involves interaction between mobile phone manufacturers and OS vendors (Blackberry, Google), mobile phone operators (ATT, Verizon, etc), and banking organizations (VISA), leading to a complex and intertwined web of control. For example, viaForensics announced a Google Wallet vulnerability almost a year ago, but it has yet to be patched because the fix would require a "change of agency" rather than a quick OS patch. Thus, the outcome of business interests and complex business dealings is likely to be more opportunities for attacks that target the fuzzy boundaries between these entities.

Existing efforts attempt to address these security concerns in several ways. First, many mobile operating systems turn off the NFC interface when the screen is locked. But if the OS is compromised, a malicious rootkit can keep the NFC interface turned on when the screen is locked, thereby thwarting this defense. Second, mobile payments ask the user to provide a four digit pin before an NFC-initiated payment. However, this is also vulnerable to attacks such as the one demonstrated in [2], where the pin code was inferred by looking at data stored by an NFC payment application. Once the pin-code is cracked, a rootkit can potentially bypass user input entirely and make a mobile payment that the user is completely unaware of. Third, phones can use hardware (secure elements) that provides security guarantees for mobile payments ([9, 14]), but such hardware is not available for phones acting as readers. Thus, none of the mechanisms fully address the scope of security issues presented by NFC.

We argue that there is a need for a hardware-based "NFC guardian", *EnGarde*, that is perpetually attached to the phone, and acts as an NFC firewall that allows legitimate interactions to occur as normal, while blocking unwanted NFC interactions through jamming. While

the idea of jamming unwanted interactions is reminiscent of RFID blocker tags [11], practical instantiations of such ideas are bulky systems with large power draw, and consequently not in wide use. In contrast, our design is small, passively powered, and can be fully integrated on a mobile phone, thereby making it entirely practical.

Our design contributions are four-fold. First, *EnGarde* has the form-factor of a self-contained and self-powered thin pad that attaches to the back of the phone, and is agnostic of mobile operating system differences as well as idiosyncrasies of different dock connectors. Second, *EnGarde* is easy to use since it operates entirely through power scavenged from the NFC reader on mobile phones (or external readers accessing the phone). Thus, it requires zero effort required on the part of user to change batteries, and only has a small effect on the phone in terms of overall harvesting needs. Third, *EnGarde* defends against a wide range of passive tag and active reader based attacks that cover the spectrum of NFC protocols and operational modes including those that target the phone a) in reader mode interacting with a malicious tag, b) in tag mode interacting with a malicious reader, and c) in active peer-to-peer mode interacting with a malicious phone. Fourth, *EnGarde* can be programmed to trigger upon detecting specific types of messages, protocols, or transactions that are indicative of security violations, and disrupt these interactions through jamming.

Our design presents a range of technical challenges that we address in this work. First, we dramatically reduce power consumption during jamming by requiring no active transmission in most cases, rather we leverage the NFC carrier wave to generate an interfering subcarrier while scavenging energy. Second, we design algorithms that maximize the energy scavenging efficiency from the phone while simultaneously minimizing the power footprint on the phone. Third, we design an early warning mechanism that detects presence or absence of an NFC device in the vicinity without any communication occurring between the phone and the device, thereby enabling *EnGarde* to stay out of the way when there is a legitimate transaction as well as to prime itself to thwart an illegitimate one. Fourth, we prototype the complete system and hardware, and demonstrate that all of the outlined capabilities can fit in a flat form-factor of roughly five square inches, demonstrating its practicality.

Our results show that:

- We can jam tag responses with 100% success rate while consuming only 6.4 uW of power, which is considerably more efficient than prior approaches that have used active jamming.
- We can accurately detect tag presence with an accuracy of 95% under a wide range of conditions, while having negligible impact on legitimate communications.
- We can operate continuously power *EnGarde* solely through NFC-based power scavenging, while being 4× more efficient than a naive harvesting approach that does not consider the host phone's power consumption.
- We can defend successfully against attacks similar to a known URL attack scenario, and show that we can detect and block a particular NDEF URL type with 100% accuracy, while allowing other NDEF messages to reach the phone unimpeded.

2. DESIGN REQUIREMENTS

In this section, we discuss some of the requirements that we used as the rationale for our design choices in *EnGarde*.

Protect all NFC Modes: A central design goal is protecting all NFC modes implemented on a mobile phone. This means that *EnGarde* should be able to block NFC messages between the phone and external entity whether the phone is acting as a reader or in tag emulation mode. The specific types of attacks *EnGarde* should protect against are:

- Malicious tags deployed in infrastructure (such as tags with URLs). When the phone discovers and interprets the tag's information, it is instructed to take some action that compromises the phone's security; this could be the phone being directed to a malicious web site. In this attack, we need to protect the phone while it acts as a Reader and block communication before the phone receives the malicious data.
- A Phone encounters a malicious device in peer-to-peer mode. Since this type of interaction can support arbitrary file transfer, the phone is vulnerable to whatever content is transferred from its peer. *EnGarde* would need to detect and block these malicious data transfers.
- An external reader reads and discovers the ID used by the phone while in tag emulation mode. This would mean that an external entity would be able to track the location of a particular phone user each time the ID is read.
- An external reader initiates communication with the phone, perhaps for initiating mobile payments. This attack occurs during tag emulation mode after the phone's ID is discovered, and could result in the user's financial information being compromised. *EnGarde* should block the release of this information.

Resilient to malware: *EnGarde* should also be resilient to attempts by malware and rootkits that

might be present on the phone to compromise its ability to protect the phone. For example, one way to power *EnGarde* would be to connect it via the phone's dock connector which could then be used to supply power. However, this also makes *EnGarde* vulnerable to a compromised OS – for example, an Android rootkit could cut off power to *EnGarde* by setting the USB dock's mode to slave. Similarly, any communication between the phone and *EnGarde*, for example, using the phone to inform *EnGarde* about the initiation of an NFC transaction, could easily be thwarted by a compromised OS. Thus, to avoid vulnerabilities, *EnGarde* should be a physically separate piece of hardware that does not need to communicate with the phone.

No impact on usability: We wanted *EnGarde* to be almost invisible, both in terms of physical form factor as well as in its effect on the usability of the phone for legitimate NFC transactions. This meant that *EnGarde* should be small enough that it can be a patch stuck on a phone (or eventually integrated with a phone's battery). It should also operate perpetually through energy scavenging without any need for recharging or changing batteries. In addition, *EnGarde* should not diminish user experience for NFC transactions that a user wishes to make. In other words, there should be negligible effect in terms of packet loss rates or distance at which NFC transactions are possible if a legitimate NFC device is communicating with the phone.

Programmable Rules: Given that the types of NFC vulnerabilities will almost certainly evolve as new attacks are discovered and known attacks are patched, we want to have a fully programmable platform where the rules upon which to jam can be specified. These rules can range from blocking all exchanges of a certain type (e.g. payments), blocking exchanges with tags that contain a URL and use the browser, blocking when certain sensitive information is transmitted in clear text, and so on. Thus, *EnGarde* should be programmable, and block only those interactions that are known to be vulnerable, while allowing other messages to get through to the phone.

Fail Safe: Since *EnGarde* relies on energy scavenging, one question is what happens if it runs out of power, which might occur after a long period where the phone is not used. In particular, since *EnGarde* decodes messages and jams only when malicious interactions were detected, what would happen if the microcontroller that makes this decision is unable to operate? The duration when *EnGarde* is charging provides a window of opportunity to an attacker. Thus, a key requirement is that *EnGarde* should fail safely, i.e. when the MCU does not have sufficient power to make intelligent jamming decisions, it should default to a mode where it jams

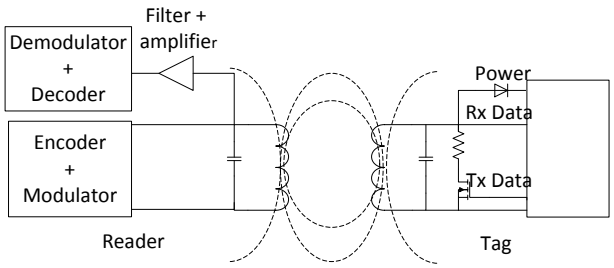

Figure 1: Functional block diagram of HF RFID reader and tag.

NFC interactions as soon as the phone initiates NFC discovery until the MCU is able to operate and make a more judicious decision. In this manner, the phone is protected whether or not *EnGarde* has charge.

3. AN OVERVIEW OF NFC

NFC is a relatively new technology. In this section, we give an overview of NFC by examining the underlying communication standards, protocols, and physical layer characteristics. The design of *EnGarde* is heavily influenced by these details.

3.1 NFC Communication Layer

NFC uses High Frequency (HF) RFID as its communication layer. The NFC standard requires that a compliant device be compatible with all existing HF RFID communication layer standards. HF RFID is used to communicate between a *tag* and a *reader*. The tag contains a globally unique ID and some data which is optionally writable by a reader. The tag is a passive electronic device which is powered by the reader during communication.

The reader powers tags in its vicinity using a magnetic field. Before communicating with a tag, the reader runs a discovery protocol to discover the tags in its vicinity. If multiple tags are seen, a collision avoidance protocol is used to identify individual tags. Once a tag is discovered, the reader uses the tag ID to uniquely address the tag for reading and writing tag data.

Since the reader generates the magnetic field to power the tag, the communication is always reader initiated. During each interaction, the reader generates the field and sends a message addressed to a specific tag; the tag, after interpreting the reader message sends a reply.

Figure 1 show the basic components of the HF reader and tag. The reader generates a magnetic field at 13.56MHz using a tuned reader coil. The tag has a coil tuned to the same frequency. Due to the magnetic coupling between these coils, similar to the operation of an electrical transformer, the reader coil induces a voltage on the tag coil. This voltage is converted to a DC voltage to

power the tag electronics. Since magnetic field strength decays rapidly with distance, NFC systems have a typical range of a few centimeters (with larger reader antennas and high-power readers, the communication range can go up to 1 meter).

Reader to tag communication: Reader to tag communication uses Amplitude Modulation (AM) of the 13.56MHz carrier. The carrier amplitude variation due to AM causes a corresponding variation of the voltage induced at the reader coil.

The tag decodes this signal variation using a simple circuit. Different communication protocol standards use AM as a primitive to encode data using different coding techniques. Table 1 shows various protocol standards and modulation formats.

Tag to reader communication: Tag to reader communication uses load modulation, where the load across the tag coil is varied by switching on and off a parallel resistor (or a capacitor). Since the tag coil receives its power from the reader coil, the varying load causes a varying current and a voltage at the reader coil.

The load modulation is used to generate a 847.5kHz sub carrier which is encoded using different coding techniques (Table 1).

3.2 NFC Device-Level Interactions

Although NFC is based on HF RFID technology, NFC has more capabilities than discovery, reading, and writing of RFID tags by a reader.

Communication modes. Unlike a traditional reader or a tag, an an NFC-enabled phone can take following multiple roles:

Phone as a reader: In this mode, the NFC enabled mobile phone behaves as an RFID reader. The phone periodically runs a tag discovery loop to identify compatible tags in its vicinity, and establishes communication with them. This mode is typically used to scan QR-code like tags that contain a short piece of information such as phone numbers and URLs.

Phone as an emulated tag: In this mode, called the *tag emulation* mode, the mobile phone behaves like an RFID tag. An external reader can discover and interact with the phone. Since the NFC related circuitry is powered by the reader magnetic field, this mode can be active even when the phone has no power. This mode is typically used for mobile payments in transit card-like applications.

Phone as a peer: Here the phone communicates in a peer to peer mode with another NFC enabled device such as a phone. In this mode, which is typically entered after one device discovers the other, both parties take turns generating the carrier. This communication mode supports the highest rate of communication and is used to share small files between mobile phones.

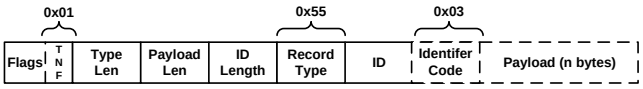

Figure 2: An NDEF message has a regular structure and holds an NDEF record. This one contains a URL that uses a prefix of "http://"

NDEF standard. The NFC Data Exchange Format (NDEF) provides a common language that enables HF RFID tags, which could be based on different HF standards, to exchange data. The well known NDEF message has following structure. In Figure 2, we show an example of one such NDEF message. This particular message contains a record that conforms to a well-defined type, as indicated by the TNF field being set to *0x01*. The ID field of the message increases the degree of specificity – the particular well defined type is a URI, as indicated by the Record Type field being set to *0x55*. The last field in an NDEF message contains the NDEF record; in a URI record, the first byte contains a prefix that is applied to the message. This particular URI uses an identifier code of *0x03* to apply the prefix *http://*; other options could have been *0x00* or *0x04* for example, which correspond to *no prefix* and *https://* respectively.

	Coding Forward	**Coding Reverse**	**Bit Rate kbps**
ISO 15693	1 out of 4/256	Manchester	1.65, 6.62, 26.48
ISO 14443-A	Mod. Miller	Manchester	106, 212, 424
ISO 14443-B	NRZ-L	BPSK	106, 212, 424
Sony FeliCa	ASK	Manchester	212, 424
ISO 18092	Mod. Miller Manchester	Manchester	106, 212,424

Table 1: Summary of NFC Forum Supported Protocols

Platform	**Card Emulation Support**
Android 4.1	While screen unlocked; only Google Wallet
Windows Phone 8	While screen unlocked/locked; Restricted applications
Blackberry 7	While screen unlocked/locked/off; Any user application supported

Table 2: The management of tag emulation mode varies across platforms.

Platform support. In addition to the various protocols and messaging formats available, there are also differences in how NFC is supported across platforms. We summarize some of these differences in Table 2. All platforms we looked at disallow use of the phone as a reader while the screen is locked. A couple of key differences are that Blackberry 7 and Windows Phone 8 allow card emulation mode to work while the screen is locked. In fact, Blackberry 7 allows any user application to access card emulation mode; however, only core applications have access to the secure element. *EnGarde* is designed to operate across all these platforms and OSs.

4. IDENTIFYING NFC PROTOCOLS

One of our design requirements is that *EnGarde* supports programmable blacklisting rules, which implies that it should be able to both listen to, and interpret all possible NFC message exchanges in a real time, and decide which ones to block and which ones to allow. However, this requires that *EnGarde* be able to decode a wide range of NFC modulation formats to determine which of the NFC protocols is being used so that it can determine the information content in them.

While this may seem similar to what an NFC reader does to read tags that support different NFC formats, there is a key difference. When an NFC reader establishes communication with another NFC device, it first goes through a discovery phase composed of multiple RFID protocol-dependent discovery messages. Hence, when a reader discovers a tag, the reader identifies and agrees upon the modulation protocol to be used with that tag. In contrast, *EnGarde* does not know what protocol is currently being used, and needs to search through all possible protocols to determine which one is being used.

While one option to perform such search might be to use a software radio, this has significant limitations. The first column in Table 3 shows the modulation pulse width for different protocols varies by more than an order of magnitude, so a software radio will need to sample the carrier at the highest rate required to decode all these protocols, and then search through the signal to identify the current protocol. However, this would result in considerable energy overhead, both because of the high rate of carrier sampling (e.g. detecting NFC 15693 requires 31× lower sampling rate than for NFC 14443-A), and because of the substantial processing overhead of performing the search. Thus, it is critical to find a cheaper option for identifying the protocol.

Leverage reader-to-tag message portion. Our key idea is to leverage the reader to tag portion of the each communication round. During each reader and tag interaction, the reader initiates the communication with ASK modulated signal while tag responds back with a subcarrier modulated signal. The ASK modulated carrier signal requires very limited energy resources to decode. We can simply examine the first pulse of the amplitude modulated carrier at the start of an NFC message to group the protocol in to several categories. Table 3 shows how different pulse characteristics map to different protocols.

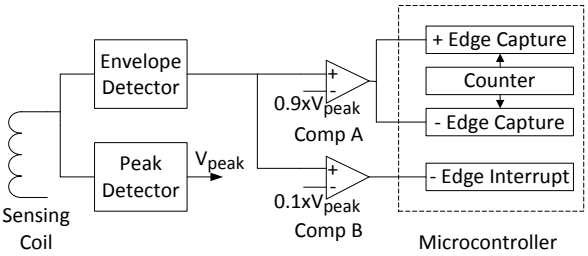

Figure 3: HW block diagram of the carrier pulse characteristics-based protocol classification.

Low-power protocol detector. Figure 3 shows the HW implementation of such a detector that does the first level of protocol classification. *EnGarde* uses a small "sampling coil" consisting of couple of turns to sample the RF signal. Two comparators are used to detect the pulse edges and the modulation depth of the envelope of the modulated carrier signal. A HW timer-bases capture units of the on board microcontroller enables the measuring of pulse width with an accuracy of a 0.5μs. An interrupt pin captures the ASK modulation type. The power consumption of the analog portion of the circuit is only 34 μW.

Once the protocol is assigned to one of the subgroups, a lightweight software solution can uniquely identify the specific protocol by examining the first few starting bytes. Once the RFID protocol is identified, we use an off-the-shelf NFC reader chip for decoding the data.

5. JAMMING NFC COMMUNICATION

Once malicious activity is suspected by examining ongoing message exchanges, *EnGarde* should disrupt the communication by jamming. While past work on protecting RFID transactions have used active jamming techniques, this requires several 100 mWs of power, which is much higher than what we can afford on *EnGarde*. Our goal is to design a cheaper jamming mechanism, that operates within the constraints of the energy that we can scavenge on *EnGarde*.

Since NFC communication has two distinct phases — reader communication and tag response–we look at these cases separately, and design jamming primitives for each of them. We then show how these primitives

Pulse (μs)	OOK	ISO Protocol and speed (kbps)
0.29	N	14443A-848
0.59	N	14443A-424
1.18	N	18092-424, 14443A-212, Felica-424
2.36	N	18092-212, 14443B-424, Felica-212
2.36	Y	18092-106, 14443A-106
4.72	N	14443B-212
9.44	N	14443B-106, 15693
9.44	Y	15693

Table 3: Different protocols that map to given characteristics of the 1st carrier modulation pulse of a NFC data packet.

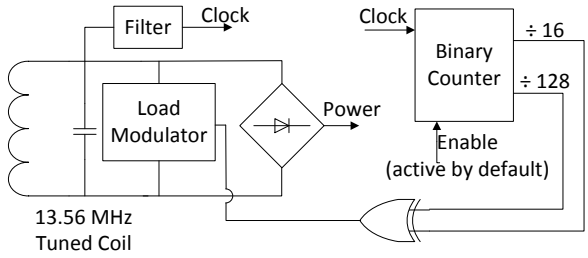

Figure 4: Load modulation-based tag jammer of *EnGarde*.

can be used to effectively jam the different NFC protocols.

5.1 Jamming Primitives

There are two jamming primitives, which we call *reflective jamming* and *pulse jamming* based on what communication modality is being jammed.

Reflective jamming. Tag-to-reader communication uses load modulation of the tag antenna using a subcarrier frequency. Our key observation is that all NFC protocols use a *common 847.5kHz subcarrier*, irrespective of the communication data rate used. Hence, to jam ongoing tag response communications, *EnGarde* only needs to generate a 847.5kHz subcarrier using load modulation of the tuned coil. Such a load-based modulation is particularly attractive since this is exactly what a typical NFC tag needs to perform, and hence can be easily done using energy scavenged from the ongoing communication that is being jammed.

A hardware implementation of such a subcarrier-based jammer is shown in Figure 4 (based on an NFC tag reference design in [4]). This circuit is similar to an NFC tag in that the jamming electronics is completely powered by the energy scavenged from the reader. The circuit has minimal components, and therefore consumes little power. Our measurements show that sub-carrier generation only consumes 6.4 μW of power.

Pulse jamming. Unlike jamming a tag response, jamming a reader (or a peer device) requires *EnGarde* to generate an active magnetic field transmission that interferes with on-going reader transmission. However, as we described earlier, generating active transmissions that are capable of swamping the signal from the reader would require 100s of mW of power (e.g. the TI TRF7970A RFID reader consumes as much as 250mW). This would be almost an order of magnitude more power than what can be scavenged on *EnGarde*, making it infeasible for our purposes.

Our approach to address this problem is to generate a targeted pulse that disrupts an ongoing communication. Since different ASK-based carrier modulation schemes require carrier modulations at bit-time durations, a carrier pulse only needs to be $\simeq$20μs (2 bit durations at the lowest data rate) long in-order to corrupt the message. Such a pulse-based jamming mechanism is orders of magnitude shorter than the duration of the shortest valid NFC message, which means that it can easily be supported via scavenged energy.

Our pulse-based jamming approach has one drawback — it is possible that a high-powered NFC reader generates a strong enough signal that our attempts at corrupting the signal does not result in a sufficiently high signal-strength difference, and is therefore unsuccessful. While this is a weakness of the technique, we think that we would block a large fraction of reader transmissions, particularly because *EnGarde* would be much closer to the phone than an external NFC device that is initiating such a signal.

5.2 Jamming During NFC Communication

Given the two primitives, we now look at how to use them to jam different NFC communication modes.

- *Tag Reader Mode:* In this mode, the phone acts as an RFID reader and reads passive tag content. A possible attack could be a malicious tag that directs the mobile phone to a malicious website. In this mode, *EnGarde* uses subcarrier-based jamming to disrupt the data exchange with the tag.
- *Tag Emulation Mode:* Here, the phone acts as a tag and responds to queries from another NFC device (a phone or an infrastructure reader). In this mode, *EnGarde* can use the subcarrier-based jamming to prevent leakage of sensitive information from the phone.
- *Peer-to-Peer Mode:* During this mode, the phone and an external NFC device exchanges information by both actively transmitting signals. *EnGarde*

needs to transmit a jamming pulse signal to block malicious interactions in this instance. While *EnGarde* may not be able to block malicious high power transmitters, we note that peer-to-peer interaction starts with a discovery step during which nearby devices are discovered. Hence, subcarrier-based jamming can be used at this stage to disrupt the establishment of a peer-to-peer communication thereby nipping such a transaction in the bud.

6. ENERGY SCAVENGING

Energy scavenging is central to the design of *EnGarde* since it allows the device to operate perpetually despite having a small energy buffer. Our approach is to leverage the same inductive coupling based harvesting mechanism by which NFC tags harvest power to communicate with a phone. This gives *EnGarde* the unique ability to jam communications while at the same time scavenging energy from the source.

While NFC enables energy transfer, one question is how much power can be harvested by *EnGarde* from the phone, and how much power is expended by the phone for this transfer. To understand this, we measure the power draw of the phone using a Monsoon power meter on a Samsung Galaxy Nexus phone running Android 4.1 (Jelly Bean) when an NFC tag is in front of the phone vs when NFC is completely turned off. We also measure the peak AC power harvested on *EnGarde* during NFC activity. Our results show that the phone's carrier is continuously switched on, hence we are able to harvest 30mW of power at *EnGarde*. This means that *EnGarde* can potentially have a fairly significant power profile. However, we also see that the phone consumes 301.5mW of power during this process, i.e. the transfer efficiency is only 9.95%. In this section, we ask how to balance the need to buffer sufficient energy in *EnGarde*'s energy buffer while maximizing transfer efficiency so that *EnGarde* has only a small impact on the phone's battery.

6.1 Scavenging mechanisms

We now outline three scavenging alternatives that have better power transfer efficiency than the full-NFC mode for scavenging power from the phone.

Opportunistic. The first harvesting mode, which we refer to as *opportunistic*, is essentially for the tag to do nothing special, and just opportunistically harvest energy coming from discovery messages that are transmitted by the phone. When a phone is unlocked, the phone sends out discovery messages periodically (10% duty-cycle) to check for the presence of nearby devices. The advantage of this mode is that the phone incurs no additional overhead beyond what it is already incurring.

The transfer efficiency in this mode reveals a surprising result — the average power consumed by the phone increases only by 14.1 mW for transmitting discovery messages, which gives us a transfer efficiency of 17.3%, which is almost double of the transfer efficiency when the phone is in full NFC active mode. The likely cause for the difference in efficiency seems to be that the bulk of the NFC protocol is implemented on a dedicated NFC reader chip that resides in the phone's battery. Only valid responses from an NFC tag results in interrupts that are handled by the operating system. Thus, when an NFC tag is transmitting valid responses, additional CPU cycles must be spent handling the read events, resulting in the extra power consumption. Thus, opportunistic harvesting is efficient but *EnGarde* only gets 10% of the power that it would have received were the phone's carrier continually active.

Tag-Spoofing. The second harvesting mode, which we refer to as *tag-spoofing*, tries to trick the dedicated NFC chip into delivering more power without interrupting the Android OS as frequently as in full NFC mode. To implement this strategy, we look at how an NFC reader performs initial detection of tag presence. After sending energy to a potential tag, the first hint that a tag may actually be present is looking for a change in the voltage of the carrier it sent to a tag. A transponder influences this voltage by *modulating* its transponder coil. If a reader observes this change in voltage, it may decide to send additional energy for subsequent communications.

We test this theory by modulating the harvesting coil via a resistor using a short pulse that is 10 us in length. Indeed, we found that this did result in the phone providing more power for a short period, after which it times out and reverts to discovery mode. The process can be repeated to ensure that the reader continues to provide additional power.

This harvesting strategy results in the phone's subcarrier being active for 33.3% of the time, so *EnGarde* gets about three times the power that was harvested in opportunistic mode, but it also incurs 41mW overhead on the phone (i.e. 27mW more than for discovery messages). However, the transfer efficiency is high at about 19%, which is even higher than what was obtained in opportunistic mode.

Subcarrier. Our third harvesting mode, *subcarrier*, closes the gap between tag-spoofing and full-NFC in terms of amount of harvested power at *EnGarde*. As discussed in §5, jamming is performed by generating a 848 kHz subcarrier. Our measurements show that subcarrier is able to increase the amount of time the carrier is active to 86.6%, and results in 168.2 mW of harvesting overhead on the phone. This gives us a transfer efficiency of 12.49%, which, while not as good as the opportunistic and tag-spoofing modes, is about 50% better in effi-

Harvesting Strategy	Phone's NFC Duty Cycle	Phone Power Consumption Overhead	Energy Transfer Efficiency
Opportunistic	10.1%	0 mW	17.30%
Tag-Spoofing	33.3%	27.2 mW	19.16%
Subcarrier	86.6%	154.1 mW	12.49%
Full NFC	100%	287.38 mW	8.04%

Table 4: Tradeoff between harvesting efficiency and phone's transfer efficiency

ciency than the full-NFC mode while giving *EnGarde* close to the maximum average power.

6.2 Demand Harvesting Algorithm

We now have three harvesting schemes that are more efficient than full NFC mode — opportunistic, tag spoofing, and subcarrier — that give us different options in terms of amount of scavenged energy at *EnGarde* and transfer efficiency.

We now describe a demand-based harvesting algorithm that runs on *EnGarde* and ensures that sufficient power is harvested from the phone to maintain its energy buffer at close to its maximum capacity, while minimizing the energy cost incurred by the mobile phone to transfer power. The algorithm works in two steps: First, it estimates the length of the next unlock interval by observing history of phone use. We use a simple EWMA filter over the history of unlock durations in our implementation. Second, it uses the estimated unlock duration to determine the fraction of time to use each harvesting mode needs to be used. Intuitively, if the buffer can be filled up just using opportunistic harvesting, then this is the cheapest approach since discovery messages are transmitted by the phone whether or not *EnGarde* is harvesting this power. If this is not sufficient to replenish energy in the buffer, then the algorithm needs to decide how to use a combination of the other two harvesting modes to ensure that the buffer is filled while maximizing transfer efficiency.

We formally define the parameters described in the model as: a) T is the current estimate of unlock duration from an EWMA-based estimator, b) B is the current energy buffer level, and B_{max} is the desired energy level, c) $\{E_{opp}, E_{so}, E_{sub}\}$ are the Energy harvested from {opportunistic, tag-spoofing, subcarrier} modes if they were exclusively used for the time T, d) $\{C_{so}, C_{sub}\}$ represents the phone energy overhead for tag spoofing and subcarrier modes; note that the overhead in opportunistic mode, $C_{opp} = 0$, since the phone is expending this energy whether or not *EnGarde* is present, and e) $\{f_{opp}, f_{so}, f_{sub}\}$ are the fraction of time opportunistic, tag-spoofing and subcarrier modes are used within the interval T, where $f_{opp} + f_{so} + f_{sub} = 1$.

Our optimization problem can now be formulated as minimizing the overhead on the phone, where overhead is defined as the additional energy that the phone needs to use above and beyond what it is expending in the opportunistic mode, given the constraints that the total energy harvested in time T should be sufficient to get the buffer to B_{max}.

$$\begin{aligned} &\textit{min:} && Tf_{so}(C_{so} - C_{opp}) + Tf_{sub}(C_{sub} - C_{opp})) \\ &\textit{subject to:} && E_{opp}f_{opp} + E_{so}f_{so} + E_{sub}f_{sub} + B = B_{max} \\ & && f_{opp} + f_{so} + f_{sub} = 1 \\ & && f_{opp} > 0, f_{so} > 0, f_{sub} > 0 \end{aligned}$$

This linear optimization can be simplified using well-known approximation methods to run in real-time on *EnGarde*. We do not provide a complete algorithm in the interest of space. The intuition behind the approximation is that $f_i, i = \{so, sub\}$ should be chosen to maximize the harvested energy E_i while minimizing the cost C_i. So the ratio of $\frac{E_i}{C_i}$ determines the selection between tag-spoofing and subcarrier modes. Because opportunistic mode has zero overhead, it will be used whenever possible.

7. NFC DEVICE DETECTION

The ability to detect the presence or absence of an NFC device in the vicinity of the phone is important in two ways: a) *EnGarde* can avoid disrupting legitimate interactions between the phone and an NFC device (smart tag, payment station, etc), and b) *EnGarde* can stop jamming when it detects that the offending device is no longer in the vicinity and no longer a threat.

One approach to solving this problem would be to look at the message interactions to determine whether or not there is another NFC device present. The phone switches from discovery mode to active mode (or software tag emulation mode) once it starts communicating with another device in the vicinity. Since *EnGarde* has the capability to decode messages, it can detect a message that indicates the start of an interaction with another device.

But this solution has a problem. If *EnGarde* is harvesting energy in any of the three modes, even if it were just doing it opportunistically, it hampers the coupling between the phone and the external device. This means that we need to detect devices prior to communication occurring between them. Similarly, while we are jamming, we cannot decode messages to detect when the NFC device leaves the vicinity of the phone, and there-

fore when we should stop jamming.

Our solution to this problem has two key contributions: a) a reliable and fast NFC device detector that leverages changes in the mutual coupling, and b) a dual-coil hardware design that includes a harvesting coil and a call sampling coil that are tailored to different needs.

7.1 Mutual Coupling-based NFC Detection

Our key idea to detect the presence of an NFC device is to leverage the manner in which inductive coupling works when several coils are present. NFC coils operate using the property of electromagnetic induction *i.e.* one coil induces a voltage in the other coil (mutual inductance). If multiple coils are present in the vicinity of an inductor, then the mutual inductance is split across these two coils. Therefore, the voltage induced in each of the coils reduces. Our idea is to detect this change in voltage at the output of the rectifier, and use it as an indicator of the presence of another NFC device.

One drawback of such a detector is that nearby metallic materials that might couple to have the same effect on voltage. When a coil generating a magnetic field is brought near a conductive material such as aluminum, it induces eddy currents that reduce the amount of flux detected in *EnGarde*. However, we argue that false positives is not a significant concern since if *EnGarde* detects no NFC interaction for a time period, it can revert to harvesting mode.

To test this theory we attach a tuned coil and voltage regulator circuit to a Galaxy Nexus phone and bring tags of various technologies in proximity of the phone / harvester pair. In Figure 5, we plot the voltage across the rectifier. The plot shows two interesting observations. First, we see that the decrease in voltage is proportional to the amount of power the tag draws. A simple tag, such as an ISO 14443-A charlie card transportation transponder, has a small impact, while a more complex tag, such as an ISO 14443-B EEPROM tag, has a much more noticeable impact. Second, we see that, as expected, other metallic objects (in this case a large aluminum plane) also cases large voltage changes.

To ensure reliable NFC device detection, we tune the detection threshold such that even a slight dip in the voltage compared to no tag being present causes *EnGarde* to backoff. To test our detector, we placed a set of tags (same as those used in Figure 5) in and out of the proximity of the phone and turned the phone's screen on and off. The results are over 100 such tag presence events, and we observe a detection accuracy of 95%, which shows that we only miss a small fraction of the cases. Note that even in these cases where a tag is not detected, *EnGarde* is still securing the phone since it is continuously listening for any message interaction that could be indicative of malicious behavior. The only downside of missed detection is a diminished user experience since the phone might need to be moved closer to the tag to ensure that *EnGarde* backs off and enables communication to occur.

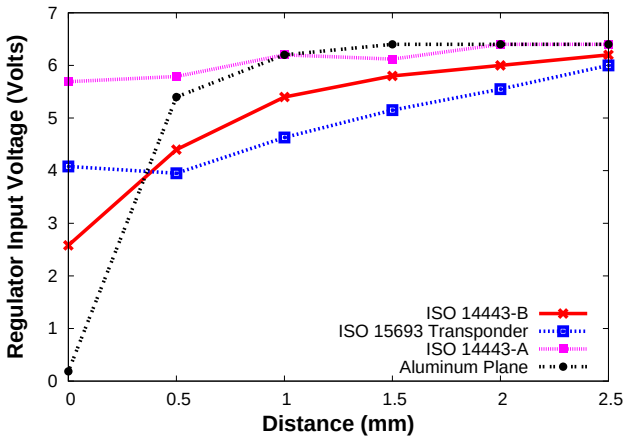

Figure 5: The presence of NFC transponders can be identified by observing a change in the output of a voltage rectifier; tag technologies that draw more power see a larger change in voltage. The presence of metal causes a huge change in voltage over a short range.

7.2 Dual-coil Design

What should *EnGarde* do when an NFC device is detected in the vicinity? One option is to have a switch and detach the load from the coil, but in doing so, *EnGarde* loses the ability to listen to messages and decide when to jam based on message content.

Our key insight is that we can decode communications by using a small "call sampling" coil that has fewer turns and is detuned to the carrier, and use a "harvesting" coil solely for harvesting and jamming purposes. The call sampling coil would reduce the level of interference to be small enough not to impact communication between the tag and external device while still retaining the ability to decode messages.

To understand how well our dual-coil design works, we look at the cases when the coil is connected and disconnected from our harvesting circuit. With the harvesting coil disconnected, we measured an average latency of 20 ms across a set of ISO 14443-A, ISO 14443-B, and ISO 15693 tags. In all test cases, we found that the phone was able to read the tags even though *EnGarde* was physically present. We then connect it to our harvesting coil and repeat the previous experiment. We found that while harvesting power, tags had an increase in communication latency of 3 ms. We also found that in a handful of test cases (15% of the cases we tested), ISO 14443-B EEPROM based tags could not succesfully read. This emphasizes the importance of the tag detection as described above.

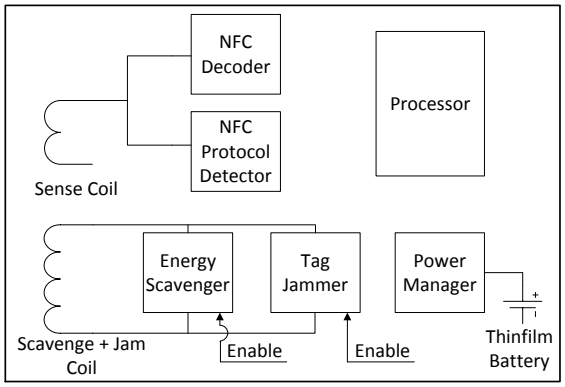

Figure 6: Block diagram of the *EnGarde* implementation.

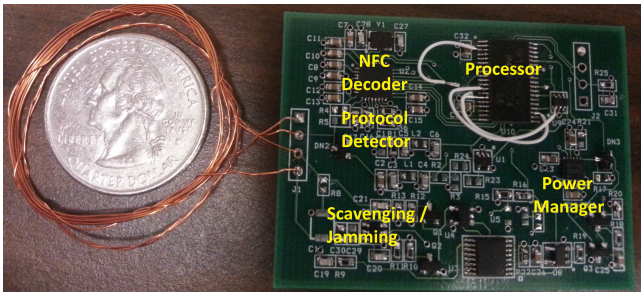

Figure 7: An image of the *EnGarde* implementation.

8. ENGARDE IMPLEMENTATION

Figure 7 shows a prototype version of *EnGarde*; our current hardware implements all the design elements, except for the pulse jamming, described in the previous sections. The current prototype measures 2 by 2.6 inches, and is well within the form-factor of a typical smartphone. We believe that future revisions can shrink this even further. We now briefly describe the key hardware sub-components used in the prototype and describe its operation using a state machine abstraction that uses the hardware primitives to enable selective jamming.

8.1 Hardware

The goal of our *EnGarde* implementation was to build a form-factor prototype that can actually be attached to the back of a mobile phone. We show how hardware subcomponents are interconnected in Figure 6.

The first key hardware element is a small "call sensing" coil that is used to sense the magnetic field in vicinity of the phone. The NFC protocol detector module uses this coil's output to detect the active NFC protocol type. The NFC decoder block uses the sense coil's output and the Rx chain of a TI TRF7970A NFC reader; the reader is configured in software by the microcontroller to decode a particular RFID protocol. The sense coil's output is also used by the microcontroller for tag presence detection.

The next key design element is a tuned coil and a capacitor arranged in parallel; this coil is used for both jamming and energy scavenging. The jamming module is controlled by the onboard microcontroller and may be enabled or disabled depending on security or harvesting needs. One important characteristic of this circuit is that it fails safe if *EnGarde*'s energy buffer is depleted – this enables protection against malicious RFID attacks and also improves the energy available via scavenging (Section 6).

A critical element of our hardware design is the energy scavenging module used to harvest energy from active reader transmissions. This module can be disabled to reduce the impact on the phone's NFC communications (Section 7). Since the microcontroller needs energy to boot, the scavenging module, much like the jamming module, defaults to active mode in the event that the energy buffer is depleted. Since jamming is based on load modulation, jamming is automatically disabled when the scavenging module is disabled.

To condition harvested energy for storage, a MAX17710 battery manager chip manages the charging of the onboard Thinergy MEC201 1 mAH thinfilm battery. The use of a diminuative thin-film battery is particularly compelling, since *EnGarde* needs to minimize thickness in addition to length and width.

Finally, an MSP430F2274 16-bit low power microcontroller manages the various sub-components of *EnGarde*. This particular microcontroller was chosen because it has an ADC that enables tag detection, has low power operating modes and can transition between power states quickly.

8.2 State Machine

EnGarde follows the state machine show in Figure 8. When *EnGarde* is drained of power or when its energy reserve is depleted, the device is in the state *no power* where the microcontroller is not active. However, our device fails safe, so jamming module is used in conjunction with the tuned coil in this mode of operation. Whenever an NFC signal is seen either from the phone, or an external device, this circuit simultaneously jams the signal while increasing power transfer from the reader. After accumulating sufficient energy, control is relinquished to *EnGarde*'s microcontroller.

After the microcontroller boots, it enters a low power state referred to as *idle mode*; while in this mode, the microcontroller listens for interrupts from the sensing coil / tag presence detector. If an NFC device is found to be present, it uses the protocol detector module to decode reader-side messages.

If an external device has entered the vicinity of the

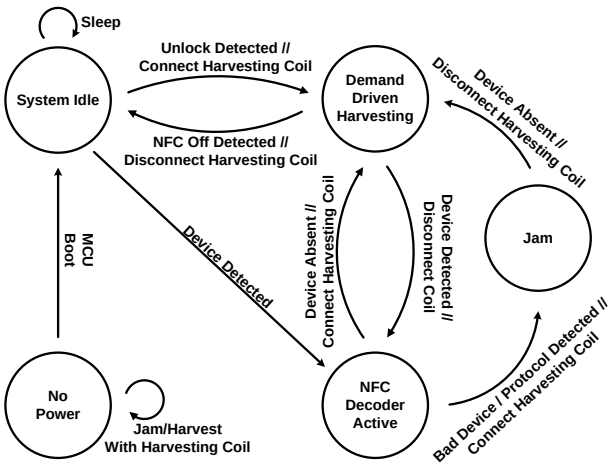

Figure 8: *EnGarde* switches between several different operational states to harvest energy, detect NFC devices, decode messages, and jam malicious NFC devices.

mobile phone, *EnGarde* also switches on the decoder and enters its highest power state where it decodes NFC transactions; before entering this state *EnGarde* detaches its harvesting coil and listens with the call sensing coil.

After activating its NFC decoder, *EnGarde* decodes messages sent by the phone, as well as messages coming from the external device. It goes through its list of blacklisting rules, and if there is a match, it enters a jamming mode. If no such blacklisted device is present, *EnGarde* will continue to listen to message exchanges until the external device exits the vicinity of the mobile phone at which point it reverts to demand-based harvesting using its tuned coil.

While jamming, *EnGarde* continuously generates a subcarrier that makes communication with external passive devices impossible for the phone to decode. If *EnGarde* detects a message from an active external device, as in peer-to-peer mode, it can generate an active subcarrier pulse for two bit durations per frame to disrupt active communications. As in the previous case, *EnGarde* continues to jam until it detects the external device has left the vicinity and resumes demand-based harvesting.

9. EXPERIMENTAL EVALUATION

Our evaluation covers three major aspects of our system: a) how well does our scavenging scheme perform over a long-term phone usage dataset, b) how effective is our jamming scheme in blocking interaction between the phone and other NFC devices, and c) a demonstration of *EnGarde*'s capability to perform targeted jamming of malicious tags while allowing benign ones to interact with the phone.

9.1 Scavenging performance

Our first evaluation looks at the performance of the scavenging subsystem. Since this evaluation depends on the actual time for which the phone is unlocked, and the duration between unlock events, we perform a trace-driven simulation using traces provide by the LiveLab project at Rice University[12]. These traces were collected from 35 users over the span of a year and contain the screen unlock data needed to fully understand the behavior of our demand-driven algorithm. However, this trace does not contain information about NFC interactions, hence we use a simple model where we vary the amount of time that a phone is interacting with an NFC device.

Simulation Parameter	Value
Quiescent Power Consumption	38.8 uW
Reader Power Consumption	32.7 mW
Opportunistic Power Harvesting	3.03 mW
Semi-Opp Power Harvesting	10.0 mW
Subcarrier Power Harvesting	26.0 mW

Table 5: A summary of the parameters used in our simulation study

Our trace-driven simulation implements a complete *EnGarde* state machine in Figure 8, and uses measured power numbers shown in Table 5. We compare the demand harvesting algorithm in *EnGarde* against exclusively using one of the other three strategies until the battery level reaches the target threshold. We use two metrics to show the performance of the harvesting strategy: a) the average battery level across the users and b) the total energy overhead on the phone to provide this energy. The results are shown in Figure 9.

The results show that an opportunistic-only harvesting strategy provides *EnGarde* with barely enough power to make it through the day, but does not consume any extra power on the phone since NFC is already on for discovery mode. The tag-spoofing mode is slightly better in that it keeps the battery level at a higher threshold, but it has higher energy drain from the phone. The subcarrier harvesting mode provides *EnGarde* with 19.5% more energy per day on average, but also consumes 4× more energy from the mobile phone's battery. The demand-driven algorithm leverages all three harvesting schemes to be close to the battery threshold and has energy efficiency close to the discovery mode.

9.2 Jamming Effectiveness

An understanding of how effectively *EnGarde* is capable of jamming NFC devices is critical towards proving that it sufficiently protects a mobile phone from exter-

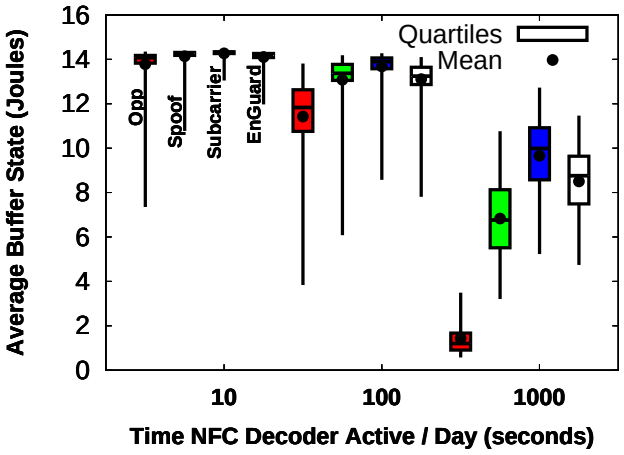

Figure 9: To be effective, *EnGarde* needs to keep energy in its buffer to log interactions between a mobile phone and NFC tags. As these interactions increase in frequency, it becomes harder to keep up with demand. By using our demand-driven harvesting approach, *EnGarde* can keep energy in its buffer while logging 1,000 seconds of NFC interactions.

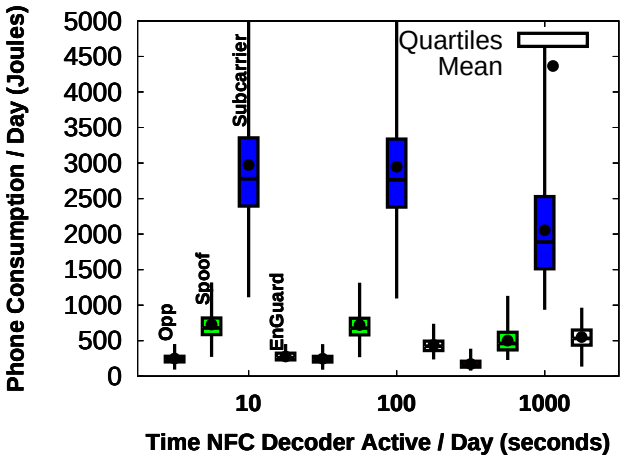

Figure 10: *EnGarde* has minimal impact on a phone's battery as workload increases.

Figure 11: Our *EnGarde* prototype meets the form factor needs required for semi-permanent attachment to a mobile phone. Here, we show it on the back of a Galaxy Nexus

nal NFC threats. In particular, we want to understand what types of tags can circumvent our jamming signal and which types of tags the phone might be more vulnerable to. We show images of our jamming setup in Figure 11.

Jamming malicious tags: We installed *EnGarde* on the back of a Galaxy Nexus phone and moved several different tags towards the phone, such that they were in direct contact with the back of the phone. The types of tags that we looked at were: ISO 14443-A. ISO 14443-B, ISO 15693, and a IT TRF7970 operating in ISO 14443-B tag emulation mode. We found that *none* of these tags could successfully communicate with the phone while the subcarrier was active. While we don't want to make any claims that communication with the phone is not possible, we haven't been able to find a tag that can get past our jamming signal.

Jamming malicious readers: Another important jamming on *EnGarde* is when an NFC reader, such as a mobile payment station, tries to read the mobile phone while in card emulation mode. We program a TRF7970 as a general purpose NFC reader, sending queries at its highest power level (200 mW). We found that when *EnGarde* is installed on the back of the phone, we effectively block 100% of the phone's ISO 14443-A response.

***EnGarde* v.s. RFID Guardian [11]:** While a direct comparison against active jamming approaches, such as the RFID Guardian, would require designing another hardware platform, we briefly discuss the key differences. NFC Guardian actively generates two 424 KHz sub bands around the 13.56 MHz, which can block NFC tags within a half meter radius. Since we are only interested in protecting the mobile phone, we are able to passively generate a similar signal at negligible energy cost. For example, in the above experiments, if change the setup so that we moved *EnGarde* some distance away from the phone, and place a tag directly on the back of the phone where *EnGarde* would normally be installed, we find that *EnGarde* blocks all communication provided that it is within 1.0 mm of the phone, but has limited effect after that distance. Thus, our jamming is extremely targeted, which improves our efficiency.

9.3 Targeted blocking of malicious interactions

We now look at a case where there is a malicious tag and other non-malicious ones, and show that *EnGarde* can be programmed with blacklisting rules that allows real-time decoding of NFC interactions and targeted jamming of malicious ones. Specifically, we look at a case study where *EnGarde* is programmed to block a particular set of URLs on an ISO 14443-B NDEF tag.

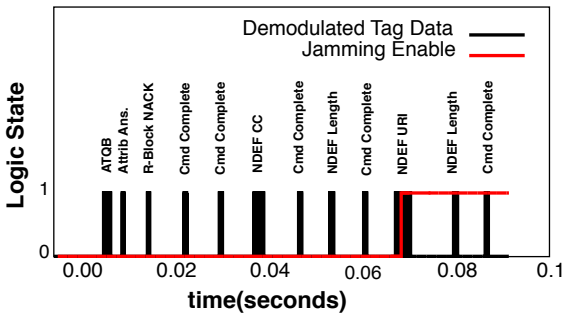

Figure 12: *EnGarde* monitors the messages sent from an emulated ISO14443-B tag, detects a malicious URL type and jams all subsequent communications

In our study, we program a TRF7970A evaluation module to behave as an emulated ISO-14443-B NDEF tag. This emulated tag approaches a Galaxy Nexus phone; in a scenario when *EnGarde* is not present, the phone uses the discovery phase to identify a tag is present. The phone then sends a series of messages that select the NDEF message stored on the emulated tag, leading up to where the tag sends its reply that contains the requested NDEF message.

After successfully decoding the NDEF response, the phone takes action according to the contents of the NDEF message. In this case, the NDEF message has its TRF field set to 0x01, which means that it is a well understood type. After checking the ID type field, it finds that this message is a URI type message that contains a URL, Phone Number, or other address from a variety of different protocols. In the first byte of the NDEF record, the phone finds the value 0x01, which corresponds to the string "http://www." The subsequent characters correspond to the rest of the URL "malware.com". The phone automatically loads this webpage in its web browser.

Next let's look at the case where *EnGarde* is installed on the back of the phone. *EnGarde* decodes all of the bits corresponding to the emulated tag's reply; we show the bits actually decoded by *EnGarde* the time series shown in Figure 12. We can see that the tag first responds to the phone's REQB discovery message with an ATQB that contains the tags pseudo unique ID. After identifying the emulated tag, the phone sends an Atrrib message that indicates this particular tag has been selected for further communication, after which the tag replies with a standard Attrib answer message.

EnGarde next observes the sequence of messages corresponding to the NDEF message selection. After observing that the tag has sent it's capability container (NDEF CC) and subsequent NDEF record length value, *EnGarde* knows where to find the NDEF message. It looks in the byte location that contains the URI identifier code 0x01, which corresponds to "http://www/" and immediately activates it's subcarrier jamming circuit to block the rest of the message. It's also important to note that *EnGarde* will parse individual characters if the URI identifier code contains 0x00, which means that no compressed prefix is applied to the URI. If the characters correspond to "http://", again the rest of the message is blocked. We tried to get the phone to read the tag 20 times and the phone was never successful.

Finally, we show that *EnGarde* allows transactions that don't satisfy our blocking rules. To prove this, we use another emulated tag, but program it with the URL "https://www.cs.umass.edu". In this case, the URL is not blocked and the page opens in the phone's web browser. Again, we found that this was robust to various placements of the tag. While we did not quantify the impact *EnGarde* had on the benign tag's read range, it wasn't noticeably different than during a typical NFC interaction.

This evaluation proves that *EnGarde*'s programmable blocking mechanism is effective, and we can decode and block in a targeted manner. *EnGarde* is designed to be flexible and support whatever rule sets satisfy potential security needs.

10. RELATED WORK

Most similar to our work is the RFID Guardian [11]. The RFID Guardian monitors and jams specific NFC communication sessions in its vicinity. This longer range performance comes at a cost in form factor and power consumption. While useful for monitoring and protecting arbitrary sets of readers and tags, *EnGarde* is considerably more targeted and is designed to protect an individual mobile phone.

The Proxmark RFID tool [3] has been used extensively in NFC security research. It has the capability of decoding arbitrary protcocols with an FPGA and additionally, it can emulate a tag. Since it uses an FPGA to decode and emulate tag responses, it can be programmed to decode any potential protocol. Two major drawbacks are its size and power consumption – while a valuable tool for debugging and security analysis, it is not suitable for continuous use on a mobile phone.

Another way to harvest energy from a mobile phone is through the audio interface [7]. Much like our NFC energy scavenger, the audio jack is universal across different phones. While a wired connection can harvest energy more efficiently, we instead opt to power *EnGarde* from the same power source as the attack surface.

Selective jamming devices are of particular interest in the area of implanted medical devices (IMD). One application is the implementation of zero power defenses [6]. *EnGarde* behaves much like a zero-power defense in that it generates a jamming signal completely

passively. More recently, a non-invasive approach towards IMDs was proposed [5]. While the proposed IMDShield has parallels to our approach towards non-invasive jamming, they use a power-hungry software radio while ours operates entirely passively.

Security is also of critical importance to mobile health. One approach towards providing security for on-body sensors is to provide a security proxy with which they communicate through [13]. While not currently implemented as such, *EnGarde* could potentially be used as a similar security proxy.

Since NFC is a relatively new technology, new vulnerabilities are constantly being exposed [8], [10]. *EnGarde* addresses these issues by providing a flexible set of security features that protect a mobile phone while remaining decoupled from platform vulnerabilities.

Finally, there have been significant efforts in securing RFID technology at the software level [1], and to place secure hardware elements in mobile phones [14]. We view our work as complementary. *EnGarde* serves as a hardware firewall that can augment software protection mechanisms to protect a mobile phone from potentially devastating attacks via NFC.

11. CONCLUSION

In this paper, we have outlined a practical, fully functional hardware shield, *EnGarde*, for phones that can intelligently protect phones from malicious NFC interactions while letting benign ones pass through. Interestingly, our design is entirely passive thereby making our form-factor small enough to be placed as a patch on a phone or even integrated within a phones case. Perhaps the most compelling aspect of *EnGarde* is that it is widely deployable as-is on NFC mobile phones that are emerging in the market, thereby making our system market-ready.

While this paper focuses on jamming, *EnGarde* has immense potential in forensic analysis of NFC interactions. There is currently limited understanding of how NFC interactions work in practice — what information is sent in the clear? how do different phones implement mobile payments? and so on. *EnGarde* is a powerful tool that can log any NFC interaction that it decodes (including those in the vicinity of the phone), which can be used to perform such analysis. We defer such analysis to future work.

12. REFERENCES

[1] V. Bhole, R. More, and N. Khadke. Security in near field communication (nfc) strengths and weaknesses. In *In proc. of Eit-2007)*, page 71. IK International Pvt Ltd, 2007.

[2] S. Clark. http://www.nfcworld.com/2012/02/09/313079/researcher-hacks-google-wallet-pin-on-rooted-android-phone/, Feb. 2012.

[3] G. de Koning Gans, J. Hoepman, and F. Garcia. A practical attack on the mifare classic. *Smart Card Research and Advanced Applications*, pages 267–282, 2008.

[4] K. Finkenzeller. *RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication*. Wiley, 2010.

[5] S. Gollakota, H. Hassanieh, B. Ransford, D. Katabi, and K. Fu. They can hear your heartbeats: non-invasive security for implantable medical devices. 2011.

[6] D. Halperin, T. Heydt-Benjamin, B. Ransford, S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *In Proc. of SSP*, pages 129–142. IEEE, 2008.

[7] Y. Kuo, S. Verma, T. Schmid, and P. Dutta. Hijacking power and bandwidth from the mobile phone's audio interface. In *In Proc. of Dev 2010*, page 24. ACM, 2010.

[8] R. Lemos. http://www.eweek.com/c/a/security/android-phone-hacked-by-researchers-via-nfc-843123/, June 2012.

[9] H. Liu, S. Saroiu, A. Wolman, and H. Raj. Software abstractions for trusted sensors. In *In Proc. of MobiSys*, pages 365–378. ACM, 2012.

[10] C. Miller. Exploring the nfc attack surface. In *Proceedings of Blackhat*, 2012.

[11] M. Rieback, G. Gaydadjiev, B. Crispo, R. Hofman, and A. Tanenbaum. A platform for rfid security and privacy administration. In *USENIX LISA*, pages 89–102, 2006.

[12] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum. Livelab: measuring wireless networks and smartphone users in the field. *ACM SIGMETRICS Performance Evaluation Review*, 38(3):15–20, 2011.

[13] J. Sorber, M. Shin, R. Peterson, C. Cornelius, S. Mare, A. Prasad, Z. Marois, E. Smithayer, and D. Kotz. An amulet for trustworthy wearable mhealth. In *In Proc. of Hotmobile*, page 7. ACM, 2012.

[14] J. Sorber, M. Shin, R. Peterson, and D. Kotz. Plug-n-trust: practical trusted sensing for mhealth. In *In Proc. of MobiSys*, pages 309–322. ACM, 2012.