

# ProxB brute: Taking ProxC ard Cloning to the Next Level

By Brad Antoniewicz

Managing Consultant

McAfee® Foundstone® Professional Services

## Table of Contents

Introduction	3
Introduction to RFID	3
Tools: Low-Frequency Readers/Simulators	4
Cloning with the Proxmark III	6
ProxBruite	7
Conclusion	11
About the Author	12

### Introduction

In 2007, Chris Paget was working for IOActive when he decided to give a talk at Blackhat DC on cloning radio-frequency identification (RFID) proximity badges, specifically covering HID ProxCard badges. Once HID Global, the makers of the ProxCard badges, discovered this was about to happen, it was reported that HID Global threatened IOActive with a lawsuit, forcing IOActive to disallow Paget from presenting. Although another researcher, Jonathan Westhues, previously showed ways to clone similar tags using his “proxmark” device, Paget’s very public attempt to raise the concern around the security of proximity cards was most effective. Since that time, it became increasingly practical for anyone with an interest, to obtain the appropriate hardware required to read and clone the ProxCard II badges. Today, HID Global offers a variety of proximity cards with varying levels of security, to be used to control physical access. Unfortunately, the ProxCard II still remains very much a staple for most organizations, and the threat it poses often goes unnoticed as practical security vulnerability.

This paper was written to further educate the reader and describe a newly implemented attack. Since much of the information presented in this paper is already available in scattered forms on the Internet, the writer assumes no responsibility.

### Introduction to RFID

RFID is a technology that facilitates tagging or identification via radio waves. So, rather than having to physically inspect a photo ID, UPC code, or insert a key into your card door, RFID can be used to accomplish the same task, only easier. For every RFID system, you have generally three components: the tag, the reader, and the backend. The tag and the reader are pretty self-explanatory, while the backend is usually some sort of processing system that associates the value on the tag (read by the reader) with something more applicable to what the actual object the tag is representing. In some cases, like physical access controls, the backend may also trigger an event such as opening the door associated with the reader. In other implementations, the tag actually stores a value that is manipulated as the tag interacts with the reader. This is important to note so that we don’t assume that the realm of RFID be limited to simply reading a value and processing it on the backend.

RFID generally operates in three frequency spectrums shown in the below table.

Name	Frequency	Distance
Low Frequency (LF)	120 kHz–140 kHz	<3 ft. (Commonly under 1.5 ft.)
High Frequency (HF)	13.56 MHz	<2.5 ft. (Commonly under 1.5 ft.)
Ultra-High Frequency (UHF)	860–960 MHz (Regional)	~30 ft.

The spectrum that any specific tag/reader may operate in is defined by the specific standard used. Since there are so many different use cases for RFID, many different standards exist, so it is important to identify what standard the technology you’re looking at follows. In addition to the operating frequency, the standard will also define a variety of other things, such as modulation, what data is stored on the tag, and how the reader queries the tag. Some standards are public, while others are proprietary, so identifying the standard used by a specific tag (which may require some reverse engineering) can be a daunting task.

### Tags

Tags are usually of most interest to an attacker since they are used as the identifier. In situations such as retail inventory, this could be useful for changing the price of an item or disabling the anti-theft security mechanism. In more enticing situations, by being able to clone a card, you can then assume the identity and physical access authorizations of the holder.

Tag design and shape will vary from manufacturer to manufacturer, but they always, at minimum, contain two components: an antenna and a chip.

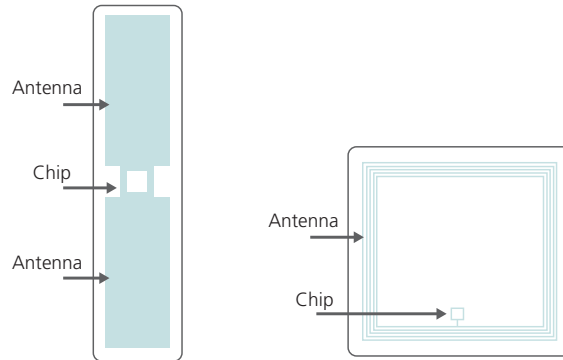


Figure 1. RFID tag design.

There are also three different types of RFID tags, described below.

Type	Description
Active	Uses a battery and automatically transmits.
Passive	No battery, requires reader to provide power and provoke transmission.
Battery-Assisted Passive (BAP)	Uses a battery, requires reader to provoke transmission. Has greater range capabilities.

When we think about RFID, we generally think about passive tags. These tags enter the field of the reader and then use that field for power to communicate back and forth with the reader.

### Tools: Low-Frequency Readers/Simulators

To even get your feet wet cloning the ProxCard II badge or playing with and RFID tags, you'll need a good reader. If you want the ability the clone, you'll also need simulator. There are a couple different devices out there, but the most popular for what we'll be doing is the Proxmark III.

#### Proxmark III

(<http://www.proxmark3.com>)

The Proxmark III is a general all-around RFID testing device. It was originally created by Jonathan Westhues (<http://www.cq.cx/proxmark3.pl>) and released under the terms of the general public license. The nice thing about this is that, if you were so inclined, you could download the specifications and build your own. Unfortunately, many people don't have the knowledge and/or drive to get involved in a project like that, and so to facilitate those people, there is a way to buy a Proxmark III online. On [proxmark3.com](http://proxmark3.com), you can buy an enhanced version of the Jonathan Westhues's original design—and for \$399, one of these little devices can be yours.

The Proxmark III supports both HF and LF reading/simulation, plus a ton of other useful features if you regularly need to identify, query, or otherwise "tinker" with RFID. The thing about the Proxmark III, is that if you're not really interested in RFID, or you don't have a decent level of technical skill, you'll be spending a fair amount of money to feel abandoned by unanswered forum posts and outdated wiki content.

**ProxPick**

(<http://www.proxpick.com>)

Not too long ago, Chris Paget gave another presentation about RFID and demonstrated a home-grown tool he created and named the ProxPick. It seemed like a great idea, and, even better, he said it would be offered for \$50. However, since then, the status of this project is unknown, and Chris's last update (in 2009) was "It's coming soon!"

**ProxClone**

([http://proxclone.com/reader\\_cloner.html](http://proxclone.com/reader_cloner.html))

ProxClone.com was created by an enthusiast to document his/her experiences playing with RFID. On the site, the author details instructions on how to create a proximity card reader/cloner for \$30. The ProxBruite firmware cannot be flashed to this device, but to implement the idea of ProxBruite is relatively easy, so modifying ProxClone's reader/cloner shouldn't be too hard. The site also has a lot of great information concerning a variety of RFID tags and their operations, so be sure to visit.

**ProxCARD II**

The most popular card in commercial card access control systems is the HID ProxCARD. Although known to be flawed, this card is still being deployed in new systems today. The passive card simply stores a 44-bit value, which is read and then sent to the backend systems, which decide whether or not that value has access to the specific door where the tag was presented. If it does, the system triggers the door to open and the cardholder gains access. This 44-bit value is split up in a number of different fields. The most important fields are the facility or site code, and the actual card number. Figure 2 shows the breakdown of the 44-bit value.

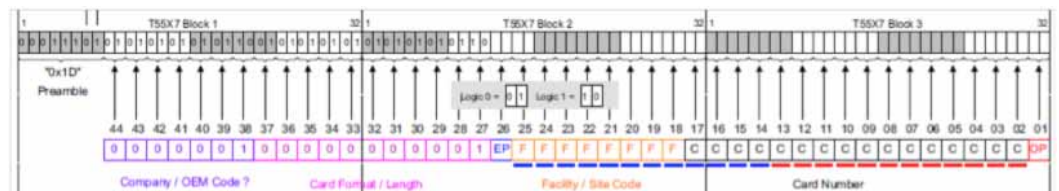


Figure 2. HID ProxCARD II format.

The facility/site code and card number are preset by the manufacturer, so when you place an order for the new cards, you have to specifically provide acceptable values/ranges for these fields.

Of this 44-bit value, only 26 bits are actually used to identify the cardholder, and so, if you're able to obtain that value, then you'll be able to impersonate that user. Additionally, it should be noted that no authentication, encryption, or any other real security mechanism is used to protect the card's value or to validate the card to the reader—all that's there is a 26-bit value.

### Cloning with the Proxmark III

As previously mentioned, the Proxmark III is de facto tool for RFID tinkering (at the time of this writing), and so we'll be using the Proxmark III for everything discussed in this paper. It already has functionality that facilitates HID ProxCard II cloning; however it is limited to just record and playback. ProxBrute (discussed below) takes that functionality one step further by offering brute force capabilities as well.

#### Manual ProxCard II cloning

The Proxmark III firmware supports ProxCard II natively through the `lf hid fskdemod` and `lf hid sim` commands. By connecting to the Proxmark III and issuing `lf hid fskdemod`, as soon as a tag enters the field of the antenna, it will be read and displayed. When you press the button on the proxmark3, it'll stop the function. An example of this operation is provided below:

```
proxmark3> lf hid fskdemod
#db# TAG ID: 98139d7c32 (5432)
#db# TAG ID: 98139d7c32 (5432)
#db# TAG ID: 98139d7c32 (5432)
#db# Stopped
```

Then, using the `lf hid sim` command, the user can have the Proxmark III transmit the tag captured by the previous command. And, just as before, pressing the button will stop the function. An example of this operation is provided below:

```
proxmark3> lf hid sim 98139d7c32
Emulating tag with ID 98139d7c32
#db# Stopped
```

Manual cloning can be a bit obvious (in a physical sense), considering the type of attack that is the topic of this discussion. A Proxmark III developer, who goes by the name "Samy," created a function for standalone cloning, where a computer would not need to be connected to the Proxmark III.

#### Standalone ProxCard II cloning

##### Power

The Proxmark III can be powered from any power source that will provide a mini-USB connection. A perfect fit for the Proxmark III are those emergency cell phone chargers. You can charge one of these up and use it with the Proxmark III (assuming the emergency cell phone charger provides a mini-USB connection).

One battery pack recommendation that has been proven to work is the Kensington K33396US<sup>1</sup> (<http://us.kensington.com/html/15458.html>).

##### Operation

The standalone mode is defined in the operating system firmware, `appmain.c` under function `SamyRun()`. Basically, the way it works is that when the Proxmark III is powered, if you hold down the button for about two seconds, it will enter standalone mode. If a computer running the Proxmark III client is connected, you'll get debugging information. Otherwise, you'll need to use the Proxmark III lights to identify what mode you're in.

The overall operation of standalone mode is described in the below diagram. A normal clone would be:

1. Hold the button for two seconds to enter standalone mode; the red light will be lit, indicating that the memory slot 1 is selected
2. Hold the button for another two seconds, and now two red lights will be lit; this means the Proxmark III is ready to read a tag and store it in memory slot 1. Once it reads a tag and stores it in slot 1, it will go back to just one lit red light.

---

1. Recommended by Chris Silvers.

- Press the button again, and it will enter simulate mode, where the tag that was just read is played. If you hit the button again, that will stop the simulation and move onto memory slot 2, indicated by the orange light. Operation for slot 2 (orange light) is the same as slot 1 (red light).

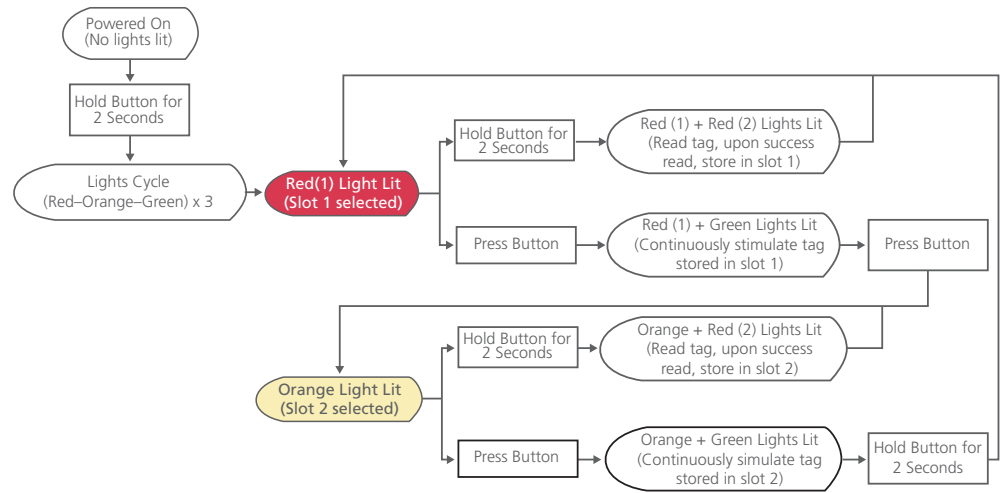


Figure 3. Traditional standalone mode operation.

### ProxBruite

ProxBruite is simply a modification to the available firmware of the Proxmark III. It enables functionality, so that one can capture a valid card and, using the facility/site code from that card, brute force another valid card number. This can be used to vertically or horizontally elevate physical privileges. For instance, let's say you're able to read one person's card and have the ability to access a common area but need to access a more privileged area. You can use ProxBruite to read a user's card, then brute force another valid tag ID. Or perhaps you're able to obtain a temporary card or previously disabled card. You can then use ProxBruite to brute force a valid card tag ID using the facility/site code of the temporary or disabled card, and gain access.

### Brute forcing approach

An important note is that ProxBruite does not attempt to brute force both the facility/site code and the card number. Instead, it starts with the value of a valid card, retrieves the facility/site code, and decrements the card number until it reaches 0x00000000. The reasoning behind this is that it will take a substantial amount of time to brute force the entire key space. It is, therefore, likely you'll have better luck when you already know the facility/site code. The entire key space for the 26-bit facility/site code and card number is  $2^8 * 2^{18}$ , or 67,108,864. It takes about one second per try so, you're looking at standing in front of the door causing a loud beeping sound every second for somewhere around 776 days (over two years) to try every attempt. Granted, you don't need to exhaust the entire key space, but it's just a lot easier and likely when you already know the site code.

That said, it does not, by any means, mean brute forcing a valid card number and facility/site code is impossible or totally impractical. When you purchase a new set of cards you're asked to provide a facility/site code and a starting number or range for the card numbers. This indicates that card numbers may be sequential or based on some predictable algorithm. So let's say card numbers are sequential starting at 0x00000001. If you guess every possible site ID, (256 possible values), with the card number 0x00000001, you may be able to identify both the facility/site code and card number in the same guess, in under five minutes.

Additionally, you could use a number of other approaches to deduce valid values while reducing the key space. For instance, instead of guessing just one value for each site ID, you can guess maybe 10 to 20 (or even 500) values spread out across possible values for the card number. If you land somewhere within the sequential/predictable range of card numbers purchased, you've guessed successfully and gained access.

The algorithms that can be conjured up to help reduce the brute force time seem limitless and perhaps in the later releases of ProxBruite, one or many of them will be implemented. Unfortunately, for the version timed with this white paper (ProxBruite v0.3), the only method implemented requires prior access of a (once) working tag, containing a valid site/facility code. In practice, over the course of a number of different tests, using the ProxBruite decrementing method, some "starting" tags would successfully guess multiple, valid tag IDs in seconds, while others would take longer. On average though, I discovered that by standing with the Proxmark III antenna in the field of the reader for about five minutes, I'd likely guess at least one valid tag ID.

**Standalone brute forcing**

Standalone brute forcing is similar to the default functionality within the Proxmark III. However, instead of performing the same playback functionality on slot 2 as slot 1, when the orange light and green lights are lit, the Proxmark III is in ProxBruite mode, meaning it is guessing card ID values using the same facility/site code that was previously stored in slot 1, or explicitly read and stored in slot 2. The method by which it guesses card ID values is very simple—it just sequentially decrements the valid value by one and keeps trying until it reaches an ID value of 0x00000000.

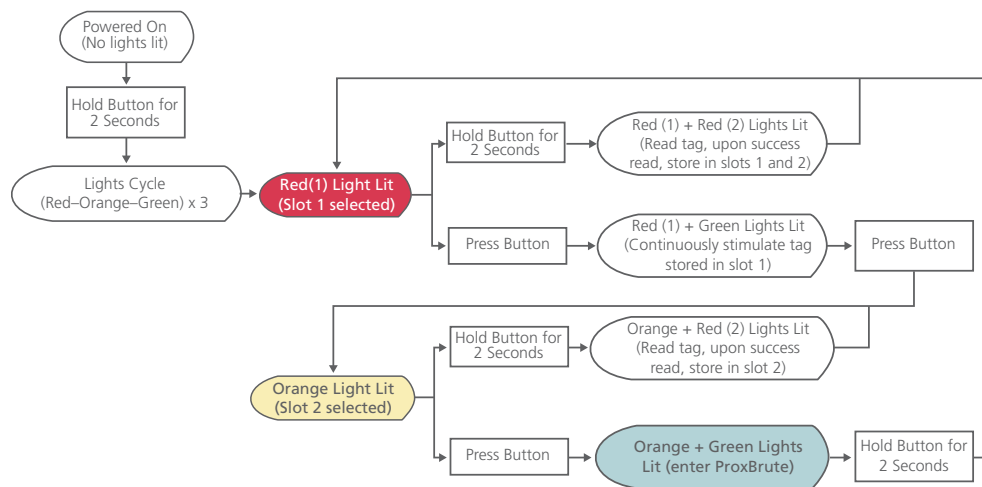


Figure 4. ProxBruite's standalone mode.

**NOTE:** Keep in mind that if a computer is powering the system via standalone mode, you'll get debugging information via the Proxmark III client. This is really useful if you want to do something like record successfully brute forced tags for usage later on.

Here's some output from a valid run with a computer connected.

Action	Debug Output
Hold down button for two seconds. Enter standalone mode. Lights cycle colors, then red LED becomes lit (slot 1 selected).	#db# Stand-alone mode! No PC necessary.
(Red lit) Hold down button for two seconds. Enter record mode. Two red LEDs become lit. After successful read, tag is stored in slot 1 and slot 2, and only one red LED is lit.	#db# Starting recording #db# TAG ID: 98139d7c32 (5432) #db# Recorded 98139d7c32 #db# [ProxBrute] In Mode Red, Copying read tag to Orange
(Red lit) Press button. Briefly enter play mode. Red and green LEDs are lit.	#db# Playing #db# Red is lit, not entering ProxBrute Mode #db# 98139d7c32
(Red lit and green lit) Press button. Slot 2 selected. Orange becomes lit.	#db# Done playing
(Orange lit) Press button. Enter ProxBrute mode, using previously recorded tag in slot 1. Orange and green are lit. Button is held down to stop and exit, then no lights are lit.	#db# Playing #db# Entering ProxBrute Mode #db# brad a. - foundstone #db# Current Tag: Selected = 1 Facility = 00000098 ID #db# Trying Facility = 00000098 ID 139d7c32 #db# Stopped #db# Trying Facility = 00000098 ID 139d7c31 #db# Stopped #db# Trying Facility = 00000098 ID 139d7c30 #db# Stopped #db# Told to Stop #db# Exiting proxmark3>

### Source code

The source code for ProxBrute is incredibly simple. Original functions were duplicated so that no other functionality would be unintentionally affected.

ProxBrute functionality is enabled by reflashing your Proxmark III with the ProxBrute operating system firmware. The ProxBrute operating system firmware is just a fancy name for the modified version of the Winter 2010 release (SVN revision 465) that has been patched. The patch is available via the ProxBrute bundle, which contains the patch, the precompiled ProxBrute operating system firmware, and Linux client utilities compiled on BackTrack Linux 4.

Source code is available from <http://downloadcenter.mcafee.com/products/tools/foundstone/proxbrute-bundle-v0.3.tar.gz>.

### Compiling

Within this paper, we're dealing specifically with revision 465 from the Proxmark III trunk. The Proxmark III (bootloader, fpga, and os) was first flashed with the Winter 2010 release, then modifications were made to the operating system image from revision 465 and then the Proxmark III's operating system was reflashed. During testing, I observed unpredictable results within Microsoft Windows, so I'd highly recommend just booting into Backtrack to set up your build environment.

Install all the required packages:

```
sudo apt-get install build-essential libreadline5 libreadline-dev
libusb-0.1-4 libusb-dev libqt4-dev perl pkg-config
```

Check out revision 465:

```
svn co -r 465 h http://proxmark3.googlecode.com/svn/trunk/proxmark3-r465
```

Download devkitARM (<http://sourceforge.net/projects/devkitpro/files/devkitARM/>), and extract. Then adjust your PATH to point to the appropriate location of the extracted devkitARM.

```
export PATH=${PATH}:/root/devkitARM/bin
```

Patch the source to enable ProxBruite functionality:

```
cd proxmark3-r465
patch -p1 < ../proxbrute-bundle-v0.3/proxbrute-v0.3.patch
```

Your output should be this:

```
patching file armsrc/appmain.c
patching file armsrc/apps.h
patching file armsrc/lfops.c
patching file armsrc/version.c
```

To compile, just execute the following:

```
cd armsrc
make
```

### Flashing

The ProxBruite bundle was created so that you could boot up BackTrack 4, extract the bundle, and flash the Proxmark III with the ProxBruite firmware. However, you may want to compile everything from source, so there is additional information listed here.

If you are building everything from source, you may have to recompile the flasher and the Proxmark III :

```
cd ../client
make clean
make
```

Then to flash, execute the following:

```
./flasher ../armsrc/obj/osimage.elf
```

Or, if you just downloaded the ProxBruite bundle with the precompiled image in it, you can use this command:

```
/path/to/bundle/client/flasher /path/to/bundle/osimage-proxbrute-v0.3.elf
```

Here's the example output of the flashing process:

```
# client/flasher osimage-proxbrute-v0.3.elf
Loading ELF file 'osimage-proxbrute-v0.3.elf'...
Loading usable ELF segments:
1: V 0x00110000 P 0x00110000 (0x0000bc44->0x0000bc44) [R X] @0xb8
2: V 0x00200000 P 0x0011bc44 (0x00000d68->0x00000d68) [RWX] @0xbd00
Note: Extending previous segment from 0xbc44 to 0xc9ac bytes
```

```
Waiting for Proxmark to appear on USB.....
```

```
Connected units:
```

```
1. SN: ChangeMe [006/013]
```

```
Found.
```

```
Entering bootloader...
```

```
(Press and release the button only to abort)
```

```
Waiting for Proxmark to reappear on USB....
```

```
Connected units:
```

```
1. SN: ? [006/014]
```

```
Found.
```

```
Flashing...
Writing segments for file: osimage-proxbrute-v0.3.elf
 0x00110000..0x0011c9ab [0xc9ac / 202 blocks].....
.....
.....
..... OK
```

```
Resetting hardware...
All done.
```

Have a nice day!

### Conclusion

The design of the ProxCard II (and other, similar proximity cards) is significantly flawed, which not only facilitates cloning, but other more serious attacks. ProxBrute aims to demonstrate the risks associated with implementing this type of proximity card system and hopes to further highlight the fact that these systems should no longer be installed. HID offers a number of different secure proximity card systems, which have since been developed to correct the issues found in the ProxCard II. Unlike the ProxCard II, many of the newer system design specifications are publically available and have been reviewed by the community to ensure a secure design. Unfortunately, however, the ProxCard II is still widely implemented, and the threats to these systems are relatively unknown to those outside of the hacking community.

### About the Author

Based out of the New York office of McAfee Foundstone, Brad is a managing consultant focusing on internal/external penetration testing, web application penetration testing, firewall configuration reviews, network architecture reviews, and 802.11 wireless assessments. Brad is a contributing author to the sixth edition of *Hacking Exposed* and the second edition of *Hacking Exposed: Wireless*. He has authored articles and white papers such as *802.11 Attacks*, *Defeating the iPhone Passcode* and *Java Basics—Extracting Decompiling, Recompiling, and Signing*. He has developed a variety of different hacking tools, such as FreeRADIUS-WPE and multiple internal McAfee Foundstone testing tools. Brad is also involved in security research and is an active member of the McAfee Foundstone internal vulnerability discovery team, which focuses on finding flaws in popular software.

### About McAfee Foundstone Professional Services

McAfee® Foundstone® Professional Services, a division of McAfee, offers expert services and education to help organizations continuously and measurably protect their most important assets from the most critical threats. Through a strategic approach to security, McAfee Foundstone identifies and implements the right balance of technology, people, and process to manage digital risk and leverage security investments more effectively. The company's professional services team consists of recognized security experts and authors with broad security experience with multinational corporations, the public sector, and the U.S. military.

### About McAfee, Inc.

McAfee, Inc., headquartered in Santa Clara, California, is the world's largest dedicated security technology company. McAfee is relentlessly committed to tackling the world's toughest security challenges. The company delivers proactive and proven solutions and services that help secure systems and networks around the world, allowing users to safely connect to the Internet, browse, and shop the web more securely. Backed by an award-winning research team, McAfee creates innovative products that empower home users, businesses, the public sector, and service providers by enabling them to prove compliance with regulations, protect data, prevent disruptions, identify vulnerabilities, and continuously monitor and improve their security. [www.mcafee.com](http://www.mcafee.com).

