SpoofKiller:
You can teach people how to pay, but not how to pay attention.

Markus Jakobsson
PayPal

Hossein Siadati
Semnan University

Abstract

We describe a novel approach to reduce the impact of spoofing by a subtle change in the login process. At the heart of our contribution is the understanding that current anti-spoof technologies fail largely as a result of the difficulties to communicate *security* and *risk* to typical users. Accordingly, our solution is *oblivious* to whether the user was tricked by a fraudster or not. We achieve that by modifying the user login process, and letting the browser or operating system cause different results of user login requests, based on whether the site is trusted or not. Experimental results indicate that our new approach, which we dub "SpoofKiller", will address approximately 80% of spoofing attempts. Free licenses to the technology are offered by the organization owning it, and serious discussions with a major OS vendor have been initiated, with the goal of protecting payments made from apps running on their platform.

1 Introduction

As people interact with each other, they observe cues that indicate the identity of the party they interact with. This is a form of authentication that is implicitly taking place. It is not limited to human-to-human interaction, but people also implicitly form opinions about the identity and validity of websites, as they observe these. Given human inaccuracy, this is a very vulnerable form of authentication, and one that makes *spoofing* possible.

Just as fraudsters may attempt to impersonate a trusted person to an intended victim, they may also *spoof* emails, websites and apps. This is a common technique used by phishers. Phishers use webpage spoofing to dupe Internet users into believing that they are visiting trusted websites, and giving out their passwords (or other credentials) to these sites.

At the risk of stating the obvious, phishers are only successful if (a) they manage to trick their intended victims, *and* (b) the resulting actions of these victims are beneficial to the fraudsters. Both conditions are necessary.

Typical security measures aim to mitigate the threat of spoofing by addressing the first condition, i.e., by avoiding that intended victims are tricked. This is done by conveying security and risk to users – e.g., using locks and conveying recognizable URLs to represent security, and by issuing warnings and requiring unusual user action to represent risk. This general approach is not very effective, as it relies on users paying close attention to subtle cues and to not act out of habit. The simple but somewhat ironic beauty of the approach we introduce is that it turns reflexive user behavior from being a danger (as it is today) to being a distinct *advantage*. When users are habituated to the methods we promote, the very same reactions that currently make these users fail to notice and act on indications of risk are harnessed and made to protect them. The approach we take to achieve this goal relies on undermining the *second* condition for success for phishers, namely that *the resulting actions of victims are beneficial to the fraudsters.*

We modify the user login behavior to include an action that generates an interrupt (i.e., power button press). Normally, this means means "terminate" or "turn the screen off and terminate" (depending on the phone operating system), but the meaning is changed to mean "go ahead" for whitelisted sites. We make this action mandatory for whitelisted sites. As a result, as a user visits a spoofed site – believing she is at a legitimate site – acts just as she does on legitimate sites. On spoofed sites, this causes the termination of the browser, and therefore also of the offending website. (It is worth mentioning that while malware with root access can spoof the pressing of the power button, a spoofed webpage *cannot*; nor can a malicious app without root access.)

The new functionality can easily be achieved by modifying browsers, as demonstrated in a proof-of-concept implementation we have made by modifying the open-source browser Zirco. In our modified version, which runs on Android devices, the meaning of the power button is changed in the context of whitelisted sites. It could either simply be made to mean "go ahead, enter your password now" as in our implementation, or "we have autofilled your user name; now enter your password", to provide a user incentive for make up for the extra button press. However, the meaning of the power button is not changed for other sites. Therefore, if a user presses the power button on a spoof site – not necessarily because she thinks it is a secure site, but simply habitually performing her normal login actions – then the browser session will end and the user be brought back to the home screen, because the interrupt handler did not find the URL on the whitelist.

A technique of potential independent value is one that we developed to force users to comply with the new login procedure, all while respecting legacy sites not to have to be aware of our needs and actions. The approach we take is simply to let the browser inject javascript in the DOM of the visited site (thereby making it appear that this javascript code was part of the website); where the injected javascript code searches for tags indicative of password fields, and rewrite the website source code to signal the whereabouts of such fields to the browser. If a user attempts to enter text in any such field without first having pressed the power button, the browser will give the user tactile feedback and an alert explaining the need to press the power button on trusted sites. This, in fact, can be the only teaching process by which user behavior is changed.

At first sight, this may seem to mean that a phishing site could modify the HTML so make sure that there would be no tag to trigger the detection of the password field. However, this is a misunderstanding, as the detection of the password field is merely a tool to train the user to press power, by recurrent conditioning as the user visits legitimate sites. *Legitimate sites* will not attempt to circumvent the detection of the password field. The abortion of phishing sites does not depend to any extent on the code of the webpages; it is simply a consequence of the user's actions.

Outline. We begin with a brief overview of related work (section 2), after which we describe the psychological principles that our solution is based on (section 3). In section 4, we describe an implementation of SpoofKiller, followed in section 5 by an experimental evaluation of it. We briefly describe a proof of concept implementation in section 7, and end with a brief discussion of future work in section 8.

2 Related Work

The problem of web spoofing was first given attention by Felten, Balfanz, Dean and Wallach [5], years before it was embraced by phishers as a tool of deceit. While credential theft aided by social engineering took place on AOL as early as the mid-nineties, it was not until 2001 that phishing of the type we are used to today started to appear, first targeting e-gold account holders [10] and then gradually becoming a threat against regular banking. Around 2005, phishing was commonly recognized as a significant problem.

Spoofing is a complex socio-technical problem, and researchers have long studied what typical users pay attention to, and *fail* paying attention to [4, 11, 14, 15, 16, 26, 28]. They have also studied the more general question of what makes people assign trust [17, 18, 21, 24, 25]. Much of this research, sadly, supports what can be understood simply from observing the rising trend of online fraud: *Typical users are not good at making proper online trust decisions.*

To attempt to improve how trust decisions are made, substantial efforts have been made to better convey statements of security to users [1, 3, 9, 12, 20, 29] and more generally, to educate users about the need to pay attention to security indicators [19, 23]. While we are not against such efforts, we think of them as last resorts – approaches to take in the absence of automated protection mechanisms.

In line with this view is a body of work aimed at protecting the user *without* any attempt at messaging [6, 8, 22]. We believe that in order for the system to be reliable, it *should not* depend on the user making proper security decisions. That is the view on which the proposed solution is based.

3 Understanding Conditioning

In learning theory, two major classes of learning processes have been identified: *classical conditioning* and *operant conditioning*. In his famous classical conditioning experiment, Pavlov described how dogs learn to associate the ring of a bell (which is referred to as the *conditioned stimulus*) to food (the so-called *unconditioned stimulus*) [13]. While classical conditioning relates to performing actions *in response to* a potential reward or punishment, operant conditioning relates to performing actions intended to *cause or avoid* the reward or punishment. More specifically, operant conditioning identifies how an individual learns that a operant or action may have specific consequences (see, e.g., [7]). As a result of operant conditioning, the individual modifies her behavior to increase the chances of the desired outcome.

Operant conditioning could be used to describe the process by which users learn how to interact with computer systems. For example, a user may learn that a click on an X-icon (operant or action) in a window results in the abortion of the associated application (consequence). Similarly, users of Android devices have learnt that pressing the power button terminates an application and locks the phone.

When a user aims to reach a goal for the first few times, she performs a collection of actions until the desirable outcome is caused. As the desired consequence occurs (e.g., the user succeeds in locking the phone), the relation to the operant/action (e.g., to press the power button) is reinforced – we say that she *learnt*.

Similarly, in the context of login, users have learnt to enable username and password entries by a click or tap in order to enter her credentials. This is *both* a matter of classical conditioning, where the opportunity to log in is communicated by the display of the login page; and of operant conditioning, where the user knows that by clicking or tapping on the fields, she will be rewarded by the access to her account.

SpoofKiller habituates users to pressing the power button to log in to legitimate sites, using a combination of rewards and punishments. In the context of whitelisted webpages, the *reward* is access to the associated account, while the *punishment* for not pressing the power button consists of tactile feedback and an alert. At the same time, the desirable login action (i.e., the pressing of the power button) is interpreted by the device as a request to terminate the session outside the context of a whitelisted website. Therefore, as soon as users have learnt the new login procedure (pressing the power button to log in), they are protected against spoof sites, which will be terminated by this action.

This leaves two important cases to be considered. First of all, it is evident that good sites that are not whitelisted would potentially suffer the same fate as spoof sites – the termination of the user's session as a result of the user's intention to log in. Apart from requesting to get whitelisted, this problem can be addressed by the operators of such sites by replacing the conditioned stimulus (the login page) with an alert, as shown in Figure 1, which makes the user aware of the procedural exception.

A second important question to consider is how fraudsters may react to the threat of having their sessions terminated. One general approach is to display an alert similar to that shown in Figure 1, but potentially with even more reassuring messaging. While this may trick some users to proceed, it will at least raise their awareness of the login session being a special case; institutional messaging by whitelisted sites could attempt to minimize this risk by reinforcing that they will *never* ask the user to avoid the power button. Another adversarial strategy is

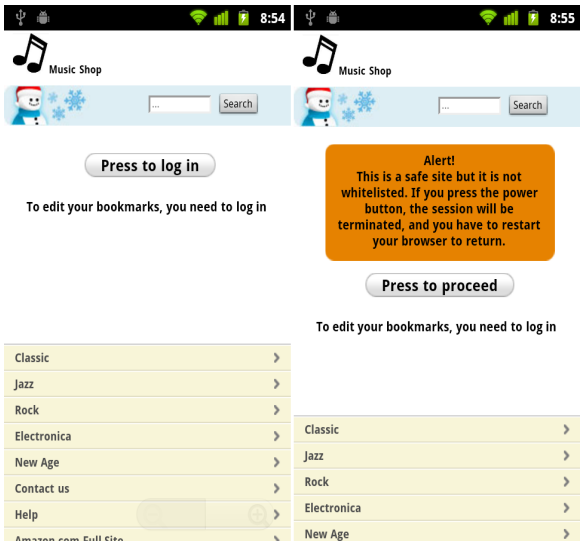

Figure 1: The figure shows how legitimate sites that are not whitelisted can avoid termination. The screen to the left takes the place of the regular login screen, to avoid the operant (power press) by removing the conditioned stimulus (the regular login screen). On the second screen, the user is instructed not to press the power button. The actual login screen (not shown above) is not displayed until the user has acknowledged. While many users may *still* press the button right after having made this acknowledgment and being shown the login screen, they will know how to return and try again.

to make the user experience as similar as possible to the real experience (which is typically the path taken by today's spoofers), and hope that the targeted user is not yet conditioned to pressing power, or will somehow fail to do this anyway.

4 App Implementation

Typical Android devices are equipped with an array of sensors – such as the touch screen; means for voice input; GPS; and an accelerometer. The events are delivered from the underlying device drivers to the OS. The OS forwards the events to active applications, or (for events such as location events or incoming calls) broadcasts them as a new *Intent*. Intents are delivered to all subscribed apps – even those that were not active at the time of the event. As a result of the broadcast of an Intent, a subscribing application may be activated. (Apps subscribe to Intents by setting up a BroadcastReceiver and its associated intent filters in the manifest of the Android application.) There are two exceptions to this rule. First, the *home button* press is just delivered to the *Launcher* (an application responsible to manage the home screen);

second, the *power button* press is not delivered to *any* third party application.

For SpoofKiller to be triggered by the power button, one *either* needs to modify the Android OS to deliver the event to our augmented browser (which would result in complications for us, as it would limit the experiment to users with dev phones), *or* one needs to trigger SpoofKiller using something that is a *consequence* of the power button being pressed – such as the *screen off* event. We did the latter.

As it is shown in the code below, we registered for the Broadcast event of Screen Off. The onReceive method is called when the Power Press is occurred. As a result, we have an event which is not catchable – or possible to generate – by a web page, and which is used to trigger SpoofKiller to check the whitelist.

```
BroadcastReceiver screenoff =
    new BroadcastReceiver() {

    public static final String Screenoff =
      "android.intent.action.SCREEN_OFF";
    //Indicate what to do
    //when the power is pressed
    @Override
    public void onReceive(
    Context context,Intent intent) {
     //Enable password field
    }};

    //Indicate the type of
    //event interested to receive
    IntentFilter offfilter =
     new IntentFilter (Intent.ACTION_SCREEN_OFF);

    //Application registers
    //to receive screen off event
    registerReceiver(screenoff, offfilter);
}
```

Most Android-based browsers use the WebView class, which is incorporated in Android WebKit package. This class is given URI's as input and loads and displays the content of associated web-pages. In addition to performing standard functionality associated with web browsing, such as running Javascript code and rendering HTML and CSS, WebView allows a web page to call a method of the browser. This functionality, as shown in the code below, allows browser manufacturers to incorporate SpoofKiller in their browsers in a straightforward manner.

```
    class JavaScriptInterface {

        @SuppressWarnings("unused")
        public boolean enableSpoofKiller() {
            //set up page to handle Power Press
            //If the page is not in whitelist,
            //this call causes page abortion
        }
    }
    .
    .
    .
    mWebView.addJavascriptInterface(
      new JavaScriptInterface(),
      "spoofkillerhandler");
```

In the code above, the browser provides a JavaScript interface named spoofkillerhandler, which enables JavaScript code in the webpage to communicate with SpoofKiller. This lets a webpage announce that it wants the support of SpoofKiller on a particular page. (Not all pages on a legitimate website needs the support, but just those that ask for credentials).

We also incorporated other functionality, such as a method to give tactile feedback when a user tries to enter his password – without first having pressed power. This has to be triggered by JavaScript in the webpage. To support legacy webpages, we have used a technique we call "on the Air Manipulation of Page" (AMP), which enables browsers to modify the contents of the webpage by injecting scripting code that determines whether the webpage should request SpoofKiller support. This is done simply by injecting a string of JavaScript as a URL to each webpage that is loaded. This is done by the browser, a trick that permits access to the document object model (DOM) of the current webpage in spite of the fact that the JavaScript code was not *really* served by the domain associated with the webpage in question. The code snippet below shows how loading a string of JavaScript as a URL lets us attach an *onclick* event to password elements in a webpage.

In our implementation of ZircoSecure – our proof-of-concept browser supporting SpoofKiller – we used the AMP technique to inject JavaScript code in a page loaded in the browser in order to let this injected routine identify fields of importance (using tags) and communicate to the browser when any such field is accessed by the user. This is to accommodate legacy websites while at the same time making sure that whitelisted pages are modified to help the browser identify fields that the user is not allowed to access without first pressing the power button. This, in other words, is what enables the user conditioning described in section 3.

The AMP technique makes it possible to deploy SpoofKiller locally, without infrastructure changes or modifications of legacy pages. Browser manufacturers – or those writing plugins for browsers – simply need to incorporate *spoofkillerHandler* and the JavaScript injection code into their browsers.

The current implementation of SpoofKiller suffers from the screen blackout, since the operating system performs that task as a direct result of detecting an interrupt caused by the power button being pressed. In order to make the SpoofKiller work smoothly and without this undesirable effect, there is a need for a modification of the Android OS. This is a straightforward modification. Using the OTA (over the air update) technology for Android, is is possible to incorporate this with any new release of the Android OS.

5 Experimental Evaluation

While convinced that SpoofKiller would work in *theory*, based on known observations on human conditioning, we also need to find heuristic support to back this belief, and to estimate the steepness of the typical learning curve as people start to use SpoofKiller. More specifically, we need to answer the following questions:

1. Is it practically feasible for users to change a frequently practiced habit, namely the manner in which they log in?

2. How long does it typically take for users to acquire a *new* login behavior, provided initial instructions and appropriate reinforcement?

These two questions relate directly to the practicability and likely user acceptance of the new approach. In particular, if the new behavior is commonly embraced and quickly becomes habitual, then this reduces the size of the population that is susceptive to abuse and reduces the risk of corruption for those who have adopted the new behavior. A core question to be answered is then:

3. *What percentage of users would be protected against typical phishing attacks after an initial period of learning?*

To find answers to these questions, we designed and carried out an experiment, which we will describe next.

5.1 Experiment Design

We recruited subjects to download and run an experiment app, either from a webpage of ours or from Google's Android marketplace[1]. During setup, we asked subjects to select a username and password – ostensibly so that only *the subject* would have access to his or her environment. Then, subjects were asked to participate in a number of sessions over time, each session having the following two parts:

1. Perform a login, wherein the user name was autofilled, and where the subject had to enter the password; but where he or she had to press the power button before doing so. Unbeknownst to the user, all actions and the time at which they were performed were logged. We will refer to this part as the *authentication phase*.

2. Type three words – chosen at random from a large set of words – as fast as possible. After performing this task, the user would be told how long it took; what his or her average time to date was; and what her ranking based on speed was. This part of the experiment was only there to take the attention away from the first part. (To add to the impression that the timing to typing was what the experiment was about, we named the experiment app *Speed Test*.)

In the authentication phase, the user was given tactile and textual feedback if she attempted to enter her password without first having pressed the power button. The textual feedback was (in blinking font) *"Notice: For security reasons, you always must press the power button before entering your password in our test."* This constituted the main tool of user conditioning.

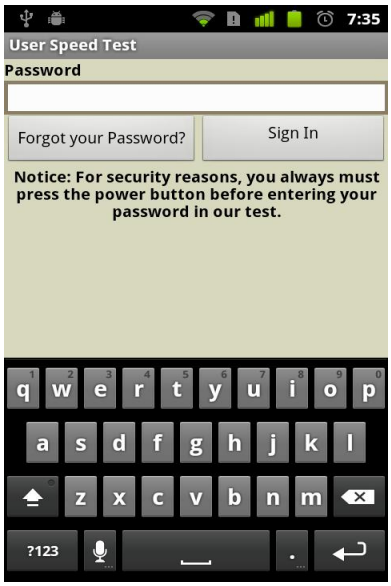

Figure 2: The figure shows the Instruction treatment in our experiment, wherein the user is told "Notice: For security reasons, you always must press the power button before entering your password in our test". In the Empty treatment, that instruction is absent, whereas in the Bad treatment the instruction given to the user is instead "Notice: Do not press power. Enter the password you use to log in to [user's email address]".

[1] Interestingly, many subjects expressed a higher confidence in the marketplace version, in spite of the absence of any user feedback or any rigid screening.

The experiment had three different *treatments*, all of them providing slightly different versions of what was shown to the user during the authentication phase. We refer to the three treatments as Instruction, Empty and Bad: The I treatment contained the instruction *"Notice: For security reasons, you always must press the power button before entering your password in our test"*, as shown in Figure 2. The E treatment was identical to the I treatment, except that it did *not* contain this instruction. Finally, the B treatment, had a "bad" instruction, prompting the user *"Notice: Do not press power. Enter the password you use to log in to [user's email address]."* That last treatment was introduced to determine whether subjects pay attention to instructions after having learnt what to do; and if so, whether they were willing to follow an instruction that has the semblance of being abusive.

To be eligible for the participation incentive, subjects had to participate for 21 days out of a month. Many subjects participated in more than one session per day – probably because we emphasized the competitive aspects of the experiment, and many tried hard to improve their speed during the phase where they typed three words. If a subject participated in more than one session per day, all sessions of that day proceeding the first session were chosen as treatment E.

We ran two versions of the experiment, which we may refer to as the *instruction heavy* and the *instruction light* version. In the instruction heavy version, the I treatment was the most common, while in the instruction light version, it was only used for a small number of days in the beginning. The aim of using these two experiments was to determine whether the conditioning that we expected to take place was a result of pre-action messaging (i.e., the instruction); post-action reinforcement (whether success or the tactile/textual feedback); or a combination of the two.

More specifically, in the instruction heavy version, the treatment shown to a user was always I, except on days 9 and 18 on which treatment E was used, and on day 21, on which treatment B was used. In contrast, in the instruction light version, I was only shown on days 1-5, after which treatment E was used until day 21, at which treatment B was used.

5.2 Subject Recruiting

Before starting to recruit subjects, we attempted to estimate the number of subjects we would need, for a desired confidence of 95%. Since the population of the smart phone users is large, we used Cochrans' formula. Based on this, we established that we needed to recruit 385 subjects, given z= 1.96 (i.e., confidence level 95%), e=0.05 (the precision level), p = 0.5 and q = 0.5 (the maximum variability). We assumed maximum variability at first since we did not know to what extent different users behave differently. As we analyzed the results, it become evident that users behave similarly to each other.

The drop-off rates in the unsupervised experiments are different based on the assigned task and the reward which is given to the participants. Based on our experience with structurally similar experiments in the past, we assumed an approximate 50% drop-off rate, suggesting the need to recruit close to 800 participants.

Recruitment of such a large number of participants was challenging, given the fact that users had to be over 18 years of age (to comply with guidelines outlined in the Belmont report); have an Android Phone; be willing to install an application; and to participate for 21 days during the course of a month. Moreover, in order to avoid bias, we avoided recruiting anybody who knew what the experiment was about, which excluded some of the otherwise most passionate potential participants.

Instruction Heavy Version. We recruited subjects by requesting participation from our LinkedIn contacts; our Google+ contacts; and our Facebook contacts. Moreover, we recruited participants among colleagues at PayPal and Google; and from members of HCI research groups. Subjects were incentivized by the chance of winning a raffle for an iPad2[2], with extra raffle tickets given to people who helped recruit subjects. Out of 198 subjects who registered, 15 entered as a result of a referral. A total of 77 of the 198 registered users completed their participation; 6 of those were due to referrals. All of these users participated in the instruction heavy version of the experiment, which was intended as the only version of the experiment until the disappointing numbers prompted us to recruit another batch of users – at which time we also decided to tweak the experiment to see whether the amount of instructions would matter much.

Instruction Light Version. In the second round of the experiment, which corresponded to the instruction light version, we recruited workers from Amazon Mechanical Turk to participate[3] and gave them the option of a $5 bonus or the chance to win an iPad/Android pad. Among the 307 who registered, 231 completed the study; more than 90% selected the cash bonus.

[2]After plentiful feedback from participants and would-be participants, we changed the raffle prize to the winner's choice of an iPad2 or an Android pad.

[3]It is against the terms of service of Amazon to ask a user to install a piece of software. While we used the payment methods associated with Amazon Mechanical Turk to pay participants, we did not use their services to *recruit* participants, and so, did not break the terms of service. These users had *voluntarily* provided contact information in previous interactions, and were contacted in this manner to ask whether they would like to participate.

Table 1: Ages of subjects in the experiment versions.

Age Range	Heavy %	Light %	Combined %
18-25	28.1	36.8	33.5
26-32	29.5	34.6	32.7
33-45	28.1	21.9	24.3
46+	14.4	6.6	9.5

Table 2: Subjects' gender in experiment versions.

Age Range	% Heavy	% Light	% Combined
Female	47.8	24.5	39.0
Male	52.2	75.5	61.0

Demographics. Tables 1 and 2 show the breakdown in terms of age and gender among the subjects in our two experiment versions (instruction heavy vs light.) This is very similar to the demographic of the Android phone owners [27, 2]. Table 3 shows the experience with entering passwords on handsets of the subjects.

Table 3: Password use on handsets.

Use	% of subjects
Daily	30
Weekly	26
Rarely	33
Never	11

5.3 Observation of Actions

The experiment app recorded all the user actions as the user ran our app, including page taps, keyboard presses, and hard-key presses (volume, home, back, menu, and power press) – along with the time at which each such action was performed. It stored the recorded data in a local database on user' handset, and then transmitted it to a back-end server for analysis. (The data was submitted asynchronously, to make it possible for test takers to take the test when they are offline, and to avoid the data lost in the case of exceptional conditions.)

From the collected data, we could determine the time it took for subjects to press the power button, after starting the authentication phase, and the number and type of actions – if any – that she performed before pressing power.

5.4 Findings

Using the data collected in the experiment, we used statistical analysis techniques to answer the questions and validate the hypotheses outlined at the beginning of section 5.

Feasibility and Learning Curve. The cumulative performance, shown in figure 3, is a measure of the how quickly subjects adopt to the new behavior. It shows the percentage of subjects performing the correct action – pressing the power button before attempting to log in – as a function of the number of days of participation in the experiment. It shows the performance of subjects in both experiment versions – instruction heavy and instruction light.

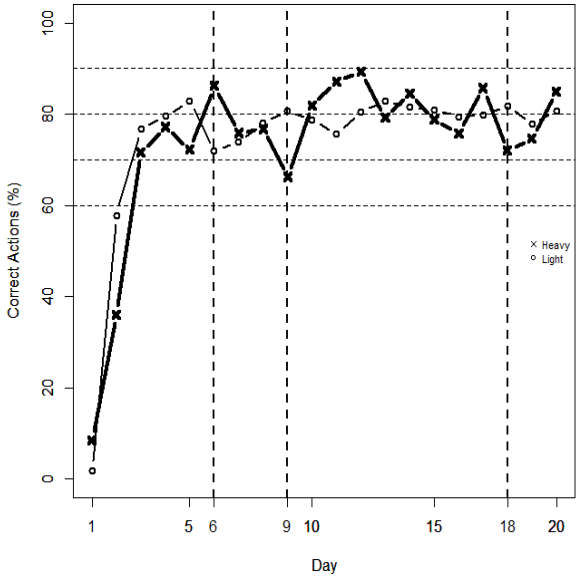

Figure 3: The figure shows the cumulative distribution of the acquisition of the safe habit (to press power before entering a password), as a function of days of exposure to the new procedure. We see a dip on days 9 and 18 in the heavy version, and one on day 6 for the light version; these all coincides with a sudden E treatment after a number of I treatments.

The learning curve shown in figure 3 is Sigmoid-like for both experiment versions, and the cumulative performance exhibits a dramatic increase during the first few days of participation; we refer to these days as the *acquisition period*, during which user tries different actions (like keyboard press, screen touch), and finally is conditioned to performing the correct operant. We can see that the proportion of correct actions is 80% $\pm$5% (with $n = 305, \chi^2 = 0.0461$) for both versions, once the users have acquired the new habit (starting at day 10). We refer to the period starting at that point in time as the *protected period*. The average performance of the users during the protected period is 79.6% $\pm$3%. This is also a measure of the probability with which these users would be protected against a spoofing attack that they would otherwise have fallen for.

A reverse regression model suggests that the cumulative percentage of correct actions is: $88.415 - 76.986/t$, where t denotes the number of days of participation. This suggests a cumulative performance of 87.5% after 84 days, with a significance level of 99%.

In the instruction heavy version (n=73), we used the E treatment (in which no instruction is provided) on days 9 and 18 in order to determine what portions of subjects have internalized the pressing of the power button by then. Our results show that 52% of the subjects had acquired the secure behavior by day 9, and 72% by day 18. In the light version (n=246), treatment E is used from day 6 to day 20. As it could be seen in figure 3, the performance is hovering around 80% during this time, with a mean of 80% $\pm$1.1%.

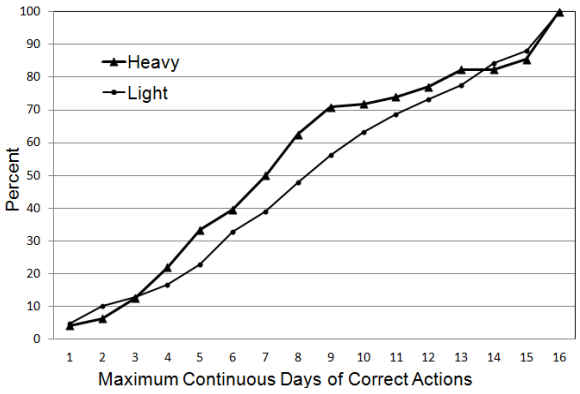

Figure 4: The figure shows the cumulative distribution of the extinction as a function of maximum number of continues days of continuously correct behavior.

The speed with which a user "forgets" an acquired habit is referred to as the *extinction rate*. During the *protected period* (days 10 to 20), the average extinction rate for subjects of the heavy version is 4.70 (n=86), meaning that users make one mistake after 4.70 days on average, and then reacquire the habit again. During the same period, subjects in the light version have an average extinction rate of 5.07 (n=246). See figure 4 for a distribution of this aspect of the users behavior. We argue that the instruction light approach is preferable to the instruction heavy approach due to the similar user performance and its cleaner user interface.

We do not believe that the differences in behavior between the instruction heavy and light versions are due to a bias in the choice of subjects – in particular since the instruction light subjects were faster learners, while – coming from Mechanical Turk – are believed to be less likely to care as much as colleagues and friends & family would.

Protection and Prediction. We want to establish the extent to which practice makes perfect in the context of SpoofKiller – or put another way, how the probability of performing a login action that is "spoof killing" depends on the number of days of exposure.

It is evident that the effects of conditioning are most notable during the first few days of exposure, given the rather rapid learning process (see figure 3 above.)

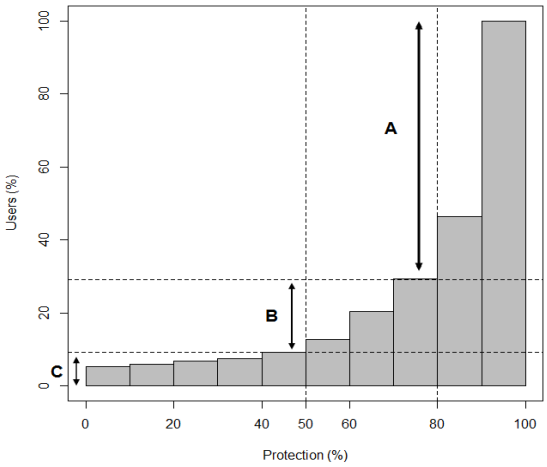

Figure 5: Clustering of users based on their performance during days 10 to 20. We informally refer to the high performers as "A students", the intermediate performers as "B students", and the low performers as "C students".

It is also very clear that not all users are equally protected, as can be seen in figure 5. Therein we show the results of performing hierarchical clustering on the experimental data based on the subjects' performance during the period we refer to as the "protected period". Three meaningful clusters are detected; we refer to these as the "A students", the "B students" and the "C students", the names indicating the extent to which subjects in these clusters learnt the new login procedure. Having made this distinction, it is interesting to see the adoption behavior of the subjects, partitioned into these three classes. See figure 6. The partition of users into different risk classes, as above, suggests the potential need for additional security measures for users who are identified as being more risk prone – what we refer to as "C students". These are easily detected based on their behavior.

One difficulty facing "C students" is that they do not maintain the desirable behavior when the instruction is removed. This can be seen from figure 6, wherein we see that the performance drops for these subjects after the instruction is removed in the instruction light version on day 6. C Students have a very high extinction rate (mean=0.47 days), which means that they have not internalized the desired habit. In comparison, "B students" have an extinction rate of 1.72 days, while "A students"

6.60 days on average. In general, there is a strong correlation (cor=0.7) between user's performance and the extinction rate.

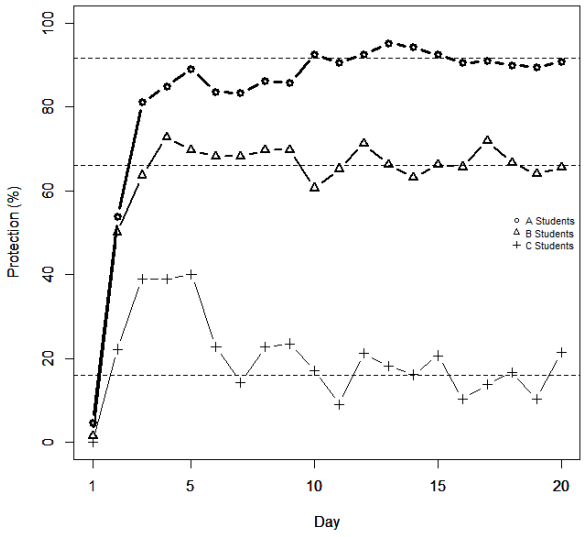

Figure 6: The performance of subjects as a function of time, where the subjects are partitioned into the classes "A students" (70% of all subjects, 92% average performance), "B students" (20% of the subjects, 66% average performance) and "C students" (10% of the subjects, 14% average performance).

Fraud Protection On the last day of the experiment, we used treatment Bad, in which users were asked not to press the power button, and to enter their email password instead of the password used in the experiment. (We did not record what password was entered, but only whether the same password was entered as during the previous session; this was to avoid stealing anybody's email password.)

Table 4: User behavior in Bad treatment (%)

Instruction	Correct	Delayed	Oblivious	Tricked
Heavy	22%	4%	48%	26%
Light	27%	4%	57%	12%
Combined	27%	4%	55%	15%

The B treatment was used to mimic a fraud case. As Table 4 shows, roughly 30% of the users pressed the power button – most of them as rapidly as during "normal" days (the "correct" reaction), and about 4% after a slight delay. The rest of the users did not press the power button. Approximately 55% of the users – independently of whether they pressed the power button or not – entered the same password as previously during the experiment, apparently oblivious to the request to enter something else, or unwilling to do so. 15% entered something else – supposedly the password to their email account.

Table 5: Behavior of classes of users in Bad treatment

Class	Correct	Delayed	Oblivious	Tricked
A Student	29%	3%	55%	11%
B Student	21%	3%	50%	25%
C Student	3%	14%	60%	21%

Table 5 shows that A students are better protected in comparison to others, in contexts involving deceit.

6 User Reactions

After a few days of participation, the added action – to have to press the power button – added less than half a second to the time the login took; see figure 7.

The reaction time is also a factor of age; younger subjects have shorter reaction time, in comparison to older subjects. Table 6 shows average speed from page load to power press for different age groups.

Table 6: Reaction time for different age groups.

Age Range	Average reaction time (s)
18-25	3.00
26-32	3.50
33-45	3.67
46+	6.25

As subjects completed their participation in their experiment, we asked them what they thought about having to press the power button before logging in with a password. Out of the 227 subjects, 127 subjects (56%) selected the response "I got used to it quickly" while 24 subjects (11%) selected the opposite response "I would have a hard time getting used to that", leaving 76 subjects (33%) having expressed neither opinion.

The average performance of the users who responded "I got used to it quickly" (n=112) was not statistically distinguishable from that of the users who responded "I would have a hard time getting used to that" (n=21).

At the same time, 114 subjects (50%) selected "If it would provide extra security of any kind, I would be happy to do that", 52 subjects (23%) selected the opposite response "I would rather not have to do that, even if it had some security benefit," leaving 61 subjects (27%) having expressed neither opinion.

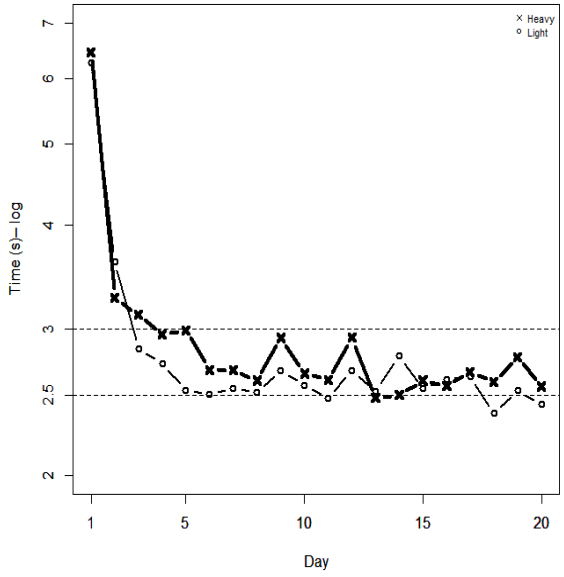

Figure 7: The graph shows the average time for subjects from page rendering to the subject pressing the power button. This is shown as a function of the day of participation. We only show the time for users who perform the correct action. The average reaction time in the protected period (day 10 onwards) for the heavy version is 2.62 seconds, and 2.56 seconds for the light version. In comparison, the user reaction time for similar-style pages but where the user does not need to press the power button is 2.5 seconds – the average time for this corresponds to the dotted line. The time associated with pressing the power button is therefore less than half a second.

Furthermore, we asked all subjects if they had any comments or questions. Several subjects felt frustrated with having to press the power button, mostly due to the fact that it turned the screen off and on again, and for one subject also to lock the phone. Neither of these would occur for a real SpoofKiller implementation, and these experiences were the results of us simplifying the implementation a bit, at the expense of a flawless user experience. More specifically, the feedback on this topic was *"It made my screen go black for a second to press the power button, so it kind of just slowed down the whole process, if it was quick I wouldn't have minded so much," "Having to press the power button was horribly annoying. To the point that I almost didn't finish the study. I would NOT use an app regularly if that was part of it," "If a power button on a phone fails by pressing it to often - that's a bad thing and that's why I responded that I would rather not have to do that, even if it had some security benefit," "My phone automatically locks every time I press the power button, which meant I had to put in my phone password to get back to the program. Sometimes when this happened, the program would again tell me that I needed to press the power button. This was frustrating," "I didn't know why I was pressing the power button. I felt weird doing it like I was being scammed or something. I wanted to know more about why I was doing it, not just because it is secure," "The power button thing was strange, and I couldn't get used to it. I don't really know how effective it could be either. Either way, this was an interesting experiment that I hope leads to greater security and password storage. I will say that I prefer the three word password to the uppercase letter-special number or symbol type passwords that are more common. I have a hard time remembering them."*

Several subjects also were puzzled or concerned about the request for a different password on the last day of the study. Feedback relating to this was *"At the end the test asked me to put in my gmail password. I didn't do that. I put in my regular password. I feel like that's a security issue," "When you prompted me for my email password on the last day, I typed the password that I had been typing each day. I wouldn't have typed in my email password at any rate, but after a month of typing the same password into the same app, the user likely will type in what he has each day, regardless of what the blinking red text tells him," "On the last day, entering the same password as every other day isn't going to disqualify me from the reward, right?"* In addition, one of the subjects provided negative feedback on our experimental app on the Android marketplace, explaining that the app tried to steal his password: "Do not install the application because in the day 21, asks for email password and application obviously wants to gain your password."

We also performed in-depth interviews with subjects willing to discuss their thoughts further. In these interviews, one user stated *'I likely forgot pressing the power button over 50% of the time. No other app requires this and 20 days of once per day use of Speed Test were not enough for the motion to become part of my muscle memory."* Another subject also made a comment that showed that he had not only been conditioned to pressing the power button before logging in, but that he had to resist pressing it *twice* – as a result of seeing the screen go off –*"It's funny, but I had a hard time shaking off the instinct/reflex to double hit the power button, as if to turn the phone right back on again. Perhaps I'm not the only one."* Again, this would not be an issue in a real implementation, but is a testament to how common reflexive behavior is.

7 Availability

A proof-of-concept version of SpoofKiller has been implemented and tested, using the open-source mobile browser Zirco. The modified browser, which we refer

to as ZircoSecure, is available for download on the Android marketplace, at `www.zircosecure.com`, which redirects to a page on the Android marketplace.

In addition to the added SpoofKiller functionality, ZircoSecure has a scrollable whitelist menu to which a user can locally add entries. Several ZircoSecure testers called out the backwards feeling of using the whitelist, bringing to mind the functionality of early web portals during days when the web was small enough for all commonly accessed sites to be listed on one portal webpage. Ihe scrollable whitelist, it should be noted, is only to make up for the fact that the whitelist we include with ZircoSecure is so limited that knowledge of what is on the list is important for a meaningful demo. A real implementation would not suffer this limitation.

Testers also pointed to the awkward browsing experience – which is a simple artifact of the Zirco browser experience, and not intrinsic to SpoofKiller or any implementation of it. Finally, testers of ZircoSecure and experiment subjects alike complanied of the temporary blacking out of the screen after the power button was pressed. This, too, is not a necessary consequence of using SpoofKiller, but simply a consequence how we chose to do it, as described in section 4.

8 Future Work

We have described how to implement the principles of SpoofKiller to protect web browsing sessions. Very similar techniques can also protect against app spoofing. This can be achieved by modifying the interrupt handler of the operating system. We are currently considering the intricacies of doing this.

It is possible to combine the use of a whitelist – as described in this paper – with heuristics and a blacklist. If a site is not on the blacklist, and the heuristics indicate that it is plausible not to be a threat, then after the user presses the power button, she may be cautioned not to enter her password if she does not trust the site, or to whitelist the site for her own purposes if she is convinced that it is secure. We have not attempted to assess the practicality if this approach, nor the impact on the user experience, but it is an interesting problem to resolve. Another interesting problem is how to best design the heuristics to determinine the likely security of a website.

Acknowledgments

We have benefitted from insightful discussions with Nathan Good, William Leddy and Diana Smetters, and from helpful feedback on an earlier draft of the paper by M. Mannan. We are thankful to Dahn Tamir for help with MTurk experiments. We also appreciate the helpful feedback we have received from numerous colleagues, and from participants in our user study.

References

[1] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. C. Mitchell. Client-side defense against web-based identity theft. 2004.

[2] P. Daniel. Android users demographics, November 19, 2010, http://www.phonearena.com/news/Android-users-demographics_id14786/.

[3] R. Dhamija and J. D. Tygar. The battle against phishing: Dynamic security skins. In *Proceedings of the 2005 symposium on Usable privacy and security*, SOUPS '05, pages 77–88, New York, NY, USA, 2005. ACM.

[4] R. Dhamija, J. D. Tygar, and M. Hearst. Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 581–590, New York, NY, USA, 2006. ACM.

[5] E. W. Felten, D. Balfanz, D. Dean, and D. S. Wallach. Web spoofing: An internet con game, Technical Report 540-96 (revised Feb. 1997), Department of Computer Science, Princeton University http://www.cs.princeton.edu/sip/pub/spoofing.pdf.

[6] I. Fette, N. Sadeh, and A. Tomasic. Learning to detect phishing emails. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 649–656, New York, NY, USA, 2007. ACM.

[7] E. Fulcher. Cognitive psychology. 2003, http://www.eamonfulcher.com/CogPsych/page5.htm.

[8] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM workshop on Recurring malcode*, WORM '07, pages 1–8, New York, NY, USA, 2007. ACM.

[9] S. L. Garfinkel and R. C. Miller. Johnny 2: a user test of key continuity management with s/mime and outlook express. In *Proceedings of the 2005 symposium on Usable privacy and security*, SOUPS '05, pages 13–24, New York, NY, USA, 2005. ACM.

[10] I. Goldberg. e-gold stomps on phishing?, http://www.financialcryptography.com/mt/archives/000190.html, July, 2004.

[11] A. Herzberg. Why Johnny can't surf (safely)? Attacks and defenses for web users. *Computers & Security*, pages 63–71, 2009.

[12] A. Herzberg and A. Gbara. Security and identification indicators for browsers against spoofing and phishing attacks. Cryptology ePrint Archive, Report 2004/155, 2004.

[13] G. V. A. Ivan Petrovich Pavlov. *Conditioned reflexes : an investigation of the physiological activity of the cerebral cortex*. Dover Publications, September 2003.

[14] C. Jackson, D. R. Simon, D. S. Tan, and A. Barth. An evaluation of extended validation and picture-in-picture phishing attacks. In *Proceedings of the 11th International Conference on Financial cryptography and 1st International conference on Usable Security*, FC'07/USEC'07, pages 281–293, Berlin, Heidelberg, 2007. Springer-Verlag.

[15] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Commun. ACM*, 50(10):94–100, 2007.

[16] M. Jakobsson and J. Ratkiewicz. Designing ethical phishing experiments: a study of (ROT13) rOnl query features. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 513–522, New York, NY, USA, 2006. ACM.

[17] M. Jakobsson, A. Tsow, A. Shah, E. Blevis, and Y.-K. Lim. What instills trust? a qualitative study of phishing. In *FC'07/USEC'07: Proceedings of the 11th International Conference on Financial cryptography and 1st International conference on Usable Security*, pages 356–361. Springer-Verlag, 2007.

[18] I. Kirlappos and M. A. Sasse. Security education against phishing: A modest proposal for a major re-think. *IEEE Security and Privacy*, 99(PrePrints), 2011.

[19] P. Kumaraguru, Y. Rhee, S. Sheng, S. Hasan, A. Acquisti, L. F. Cranor, and J. Hong. Getting users to pay attention to anti-phishing education: evaluation of retention and transfer. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, eCrime '07, pages 70–81, New York, NY, USA, 2007. ACM.

[20] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing is believing; using camera phones for human verifiable authentication. *Int. J. Secur. Netw.*, 4:43–56, February 2009.

[21] J. Riegelsberger, M. A. Sasse, and J. D. McCarthy. The mechanics of trust: a framework for research and design. *Int. J. Hum.-Comput. Stud.*, 62:381–422, March 2005.

[22] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. In *Proceedings of the 14th conference on USENIX Security Symposium - Volume 14*, pages 2–2, Berkeley, CA, USA, 2005. USENIX Association.

[23] S. Srikwan and M. Jakobsson. Using cartoons to teach Internet security. *Cryptologia*, 32(2):137–154, 2008.

[24] F. Stajano and P. Wilson. Understanding scam victims: seven principles for systems security. *Commun. ACM*, 54:70–75, Mar. 2011.

[25] R. Wash. Folk models of home computer security. In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, SOUPS '10, pages 11:1–11:16, New York, NY, USA, 2010. ACM.

[26] T. Whalen and K. M. Inkpen. Gathering evidence: use of visual security cues in web browsers. In *Proceedings of Graphics Interface 2005*, GI '05, pages 137–144, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[27] L. Woolston. Mobclix index: Android marketplace, November 17, 2010, http://blog.mobclix.com/2010/11/17/mobclix-index-android-marketplace/.

[28] M. Wu, R. C. Miller, and S. L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 601–610, New York, NY, USA, 2006. ACM.

[29] M. Wu, R. C. Miller, and G. Little. Web wallet: preventing phishing attacks by revealing user intentions. In *Proceedings of the second symposium on Usable privacy and security*, SOUPS '06, pages 102–113, New York, NY, USA, 2006. ACM.