# TALC: Using Desktop Graffiti to Fight Software Vulnerability

**Kandha Sankarapandian, Travis Little, W. Keith Edwards**

Georgia Institute of Technology
85 Fifth Street NW, Atlanta, GA 30308, USA.
*{kandha, tlittle , keith}@cc.gatech.edu*

**ABSTRACT**

With the proliferation of computer security threats on the Internet, especially threats such as worms that automatically exploit software flaws, it is becoming more and more important that home users keep their computers secure from known software vulnerabilities. Unfortunately, keeping software up-to-date is notoriously difficult for home users. This paper introduces TALC, a system to encourage and help home users patch vulnerable software. TALC increases home users' awareness of software vulnerabilities and their motivation to patch their software; it does so by detecting unpatched software and then drawing graffiti on their computer's background wallpaper image to denote potential vulnerabilities. Users can "clean up" the graffiti by applying necessary patches, which TALC makes possible by assisting in the software patching process

**ACM Classification:** H.5.m Information interfaces and presentation, H.5.2 User Interfaces, K.6.5 Management of Computer and Information Systems: Security and Protection, D.4.6 Operating Systems: Security and Protection

**General terms:** Human factors, security, management

**Keywords:** Usable security, Internet security, home users, patch management, software vulnerabilities, security framework, graffiti

**INTRODUCTION**

One of the most significant computer security threats faced by users today results from vulnerabilities in the operating system and application software installed on users' computers. Software defects—bugs such as susceptibility to buffer overflow attacks [7], cross site scripting [26], and so forth—represent vectors through which malware can infect and compromise users' machines. Once machines have been compromised, malicious parties can extract personal information from them, or enlist them into botnets to serve in further attacks on network resources. The latter threat, in particular, has a significant impact on the entire Internet community as botnets are the means to Distributed Denial of Service (DDoS), spam and phishing attacks [14]; the exponential increase in size and number of botnets [6] is a stark reflection on the number of vulnerable machines that exist in the Internet. Ironically, in many cases, patches exist to repair these vulnerabilities; however, users are often unaware that such patches exist, or are unmotivated to install them, or may not know how to install them.

Numerous reports from both government and industry sources highlight the magnitude of the threat posed by unpatched software vulnerabilities. For example, statistics from the Computer Emergency Response Team (CERT/CC) show the rapid increase in reported software vulnerabilities since 1995 [5]. NIST's report on the economic impacts of inadequate software testing estimates damage from attacks exploiting software vulnerabilities at US$60 billion/year [25]. Furthermore, testimony from the US General Accounting Office notes the importance of effective and continual patch management in addressing the "staggering" increase in software vulnerabilities [29].

Industry sources echo these same concerns. The importance of routine patching is highlighted in Symantec's security report [28], which notes that after having a firewall and antivirus software, the single most important practice for consumers to maintain their computer's security is to stay current on software patches. The SAGE report [17] from McAfee Avert Labs estimates that known software vulnerabilities are increasing at a rate of about 30% annually. Microsoft's LaMacchia [16] also notes that the window of time between when new software is released and when an exploit has been created has decreased considerably (leading to so-called *zero day* attacks, in which exploits are ready to be employed the day new software is released).

Unfortunately, just as the necessity of maintaining up-to-date patches is increasing, the complexity of doing so is also increasing: users must now be responsible for patching not only their operating system software, but also the


*CHI 2008,* April 5–10, 2008, Florence, Italy.

multiplicity of application software on their systems. While operating systems have built-in facilities (such as Windows Update) to download patches and encourage users to install them, other applications use a diverse range of update mechanisms, including requiring that users explicitly visit vendors' web sites for newer versions. Worryingly, the SAGE report indicates that in the period between December 2005 and May 2006, the vulnerabilities targeted were moving away from OS attacks, to *attacks on other software*, such as Internet Explorer and Firefox. Thus users must now contend with a host of disparate and confusing patch systems in order to ensure that all of the software on their machines is protected.

In order to patch vulnerable software, users (1) must know that such software exists in the first place, (2) know how to go about patching it, and (3) be motivated to do so in a timely manner.[1] Although there are a number of existing systems that address patching in some capacity (described below), none of these systems specifically address making home users aware of the threats that vulnerable software poses to their computer's security and their privacy, nor do they provide a holistic approach to patch management across multiple vendors' applications.

To address these challenges, we have developed a system called TALC (for Threat Awareness, Learning, and Control) that aims to augment users' awareness of vulnerabilities posed by unpatched software through unobtrusive yet persistent visual reminders, persuade them to remedy those vulnerabilities, and provide easier mechanisms for patch installation across a range of applications.

In this paper, we describe TALC, its architecture, and a deployment-based user study that we performed to determine its overall utility. We conclude with a discussion of our approach and directions for future research.

## RELATED WORK

Most of the current work on easier patch management is either vendor-specific, or has focused on managed solutions for the enterprise environment.

In the vendor-specific category, tools such as Windows Update and Mac OS X Software Update perform automatic detection and download of new patches, and single-click installation. However, these tools only work for operating system components and, as noted above, unpatched *application* software now represents the major source of vulnerabilities. Of course, many application vendors provide mechanisms for their own products (such as Adobe Online's tools for update of their Creative Suite products). However, there is no unified vendor-supported mechanism for simple updates of all software on a user's system, requiring users to deal with these on a piecemeal basis, when such systems exist at all.

In the enterprise space, a number of companies have begun to focus specifically on patch installation in managed networks, as a way for centralized IT organizations to protect the corporate network. Enterprise management solutions like Marimba Patch Management from BMC Software [3], for example, enable deployment of security patches on all devices across the enterprise. While powerful, these systems are not designed for use by home users; they require, for example, a centralized administrator who manages patch releases to the corporate network, and rely on homogeneous software installations on client devices.

There is a tension between tools like Marimba, which are proactive and aim to shield end-users from direct involvement with patching, and other tools such as Windows Update that take a more interactive approach, involving the user in the patch decision process and demanding their attention [15].

While proactive tools are, on the surface, easier to use since they do not require direct user involvement, they also do not contribute to the user's learning process: awareness of threats is a critical component in managing software vulnerabilities given the diversity each user's individual software usage patterns. Further—and perhaps more importantly—unless potential software version conflicts can be reliably determined in advance, there is a risk that an automatically installed patch will break other software on the user's computer. Such a hypothetical, fully-automated tool for managing software updates across applications is difficult to achieve outside the homogeneity of the managed corporate network, meaning that users will likely have to be involved in at least some aspects of patch decision making for the foreseeable future [10].

Given these practical realities of automated patch management, it is imperative that users be kept informed about the potential dangers of an unpatched system, as well as the benefits and risks of installing a given patch, if they are going to be involved in making patch decisions.

A challenge, of course, is that highly interactive tools can potentially annoy users to the point that they turn off such tools completely. This problem has been especially evident in security software; most common firewalls, for instance, display pop up messages about threats such as port scans. Bailey, Konstan, and Carlis [1] report that such interruptions increase task completion times, as well as user anxiety and frustration. However the suggestion from [1] of an "attention manager" that predicts opportunities for engaging with the user may not be an optimal solution for a security task like patch management that does not require an instantaneous allow/deny decision in the way that antivirus or firewall alerts do. This suggests that different, more subtle and less intrusive approaches than interrupting the user may be employed, which allow the user to interact

1 Even with systems that include an auto-update mechanism, the response window between the public disclosure of an exploit and the availability of a software patch is sufficient for a worm to exploit the vulnerability and achieve significant spread. Usually an advisory on working around the vulnerable software is released before the actual patch and educating users with these advisories can be effective in stopping exploits.


Figure 1: TALC showing graffiti on the user's desktop along with a popup description of the threat.

with the patch management system as a secondary task, but with sufficient persuasion that users do not ignore it completely.

There have also been a number of research efforts intended to address the problem of excessive dependence on user interaction for security. For example, systems such as the Chameleon System for Desktop Security [23] attempt to categorize software into activity roles in an effort to reduce impinging on the user's attention. However, most such tools are incomplete, or focus on a narrow range of threats. For example, Chameleon is a low-fi, paper-based prototype intended to address only the threat of malware.

TALC is designed in response to the need for better patch management on end-user systems. It aims to strike a balance between proactive and interactive support, in order to provide users with awareness and control over security risks without excessive attention costs or disruption to their workflow. TALC uses "calm" notifications, rather than intrusive techniques such as popups, to motivate specific user behaviors, and to provide awareness of overall system risk from software vulnerabilities. TALC also provides a holistic approach to patch management, by assisting with patch management across the heterogeneous variety of applications and software components that may be installed on a user's machine.

## DESCRIPTION

In this section, we describe how the user sees TALC, as well as how TALC detects and assists in repairing software vulnerabilities.

TALC paints graffiti on the user's desktop to indicate the presence of unpatched software on the user's system (see Figure 1). Unlike intrusive techniques such as popups, this is meant to be a "low-distraction" technique, designed to make users aware of potential problems, while allowing them to act on them in their own time. In contrast to warning dialogs that interrupt users' activities ("Your patches are out of date!"), this awareness function is intended to serve as a constant but gentle reminder, allowing users to finish their primary tasks without letting them forget about the security maintenance tasks that need their attention.

For each threat found on the user's machine by TALC a single graffiti image is chosen out of a corpus of images, and composited into a randomly selected area of the screen. TALC uses the size of the graffiti image to convey the relative severity of the threat: the graffiti is shown larger for severe threats and smaller for more mild threats.

### Why Graffiti?

We chose the graffiti visualization of software vulnerabilities to convey a general sense of "decay" or "threat" to the user, suggesting that their machine has entered a state of risk. Such notions appear to be broadly associated with graffiti in physical environments for many people. Numerous studies have confirmed this association across a number of cultures and communities; see, for example Morin et al.'s study of US public health nursing students' perception of threat in their communities [22], Bowling et al.'s study of risk perception in Britain [4] as well as others [2, 13].

We realize that this association may not hold across all cultures, or even across individuals within a given culture. (See, for instance, sources that reflect the artistic value in graffiti such as Susan Farrell's Art Crimes site, http://www.graffiti.org, as well as academic work exploring the appropriation of graffiti by various subcultures [11].) However, even for those users that may not have negative associations with graffiti, we hoped that their personal inclinations would be outweighed by the minor annoyance of having part of their desktop covered by the graffiti (covering a personally selected photo for instance), and therefore would still provide motivation to deal with the software vulnerabilities.

Our choice of a real world metaphor for visualizing security threats stems from the observations made by Redstrom, Skog and Hallnas in their work on informative art [27]. We explored a number of other, non-graffiti visualization approaches during prototyping, which we also believed might convey a sense of risk to the user. These included one that used bullet holes in the user's background image (deemed both to be too violent, and to inappropriately convey a sense of active attack rather than simple decay), and one that rendered increasingly large piles of garbage and other debris along the bottom of the user's screen (deemed to appropriately convey a sense of decay but perhaps be too easy to ignore). We believe using graffiti walks the line between the ambient media and diversion categories as described by McCrickard, et.al. in their model for notification systems [18].

### Determining and Presenting Vulnerabilities

TALC determines potential vulnerabilities through a multi-step process. First, we perform a periodic system-wide audit to identify software versions installed on the user's machine. Next, this data is compared against the online NIST National Vulnerability Database (NVD) [24], resulting in an up-to-date list of installed software for which patches exist.

When the user's cursor hovers over a graffiti area, a tooltip displays the name of the software that is vulnerable, as well as the threats posed by this vulnerability (see Figure 1).

We parse the NVD at connection time to retrieve patch information, as well as the descriptions presented to users, as shown in Figure 1 above. The language used in the descriptions in the National Vulnerability Database is often highly technical, and may be confusing to home users. To make the descriptions more palatable, we use a set of heuristics to simplify the explanations. For example, an NVD threat description such as the following:

> *The do_change_cipher_spec function in OpenSSL 0.9.6c to 0.9.6k, and 0.9.7a to 0.9.7c, allows remote attackers to cause a denial of service (crash) via a crafted SSL/TLS handshake that triggers a null dereference*

would be presented by TALC as:

> *Denial of service vulnerability that lets a remote attacker slow down/crash your computer.*

The descriptions are scanned for a small set of keywords, and predefined descriptions of the problems are presented to the user. This provides a more readable description for a large number of common classes of vulnerabilities; other descriptions that do not match our heuristics are explained with a generic message: "Other vulnerability."

We did not completely eliminate all information about the vulnerability to allow users to learn about common types of threats, and—if necessary—communicate such information to any people or organizations they trust to help them keep their computer safe. Thus, following Zurko [30], TALC places emphasis on helping users understand these security concepts through its use.

### Repairing Vulnerabilities

In addition to supporting threat awareness, TALC also allows users to take actions that mitigate threats. When the user clicks the right mouse button on graffiti, a popup context menu appears that allows them to repair the threats posed by a vulnerable program. When the user chooses to fix the program, TALC downloads and displays the webpage that contain patches or workarounds for vulnerabilities, and displays the control window shown in Figure 2. TALC also shows system information and the name and version number of the program with the vulnerability.

Unfortunately, different vendors require different processes to acquire patches: some may require that users log in, while others require a click-through license agreement, and others may simply provide direct access to the patch itself. Thus, while TALC automates the process of *finding* a patch for the detected vulnerabilities, it leaves the task of actually *installing* patches to individual users. This is not only because of the difficulty involved in automatically dealing with multiple vendors' web sites, but also because users must often be involved in the process of deciding whether a particular patch is appropriate for them. Security is not

users' only concern; they must make security related decisions in context, such as knowing whether a new software version will break compatibility with other tools. For example, Windows XP SP2—while providing important security features—broke the functionality of a number of network-based tools [21]. Simply installing such updates automatically without considering the context of other software in use can often lead to such problems.

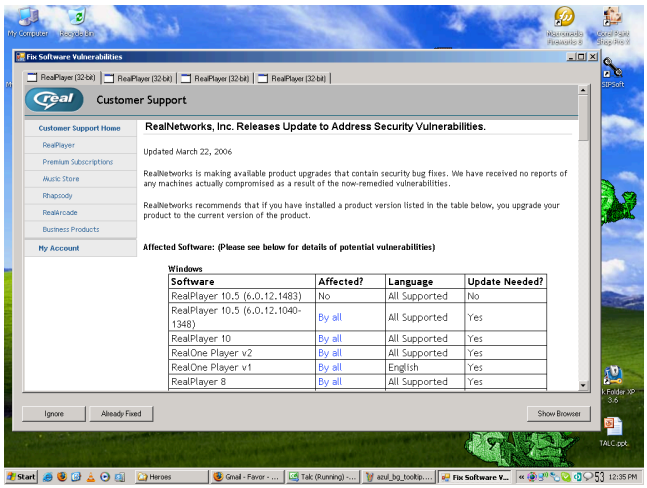


Figure 2: TALC Control window displaying the website with patch information.

## IMPLEMENTATION

TALC uses a modular implementation, with well-defined communication interfaces between modules to facilitate easy addition and replacement of components. This is exposed in the form of an API that allows developers to write pluggable modules for TALC. For example, these APIs have been used to create the software vulnerability detection system described in this paper, but can be extended to provide functionality beyond software patch management. The extensible nature of TALC is intended to be used to visualize and control a large set of security tools through a framework similar to the one described by Dourish and Redmiles in [9]. For example, information sources, sensors and aggregators can be created and plugged into TALC, allowing it to be extended to new visualizations and to detect new security threats. Our long-term goal is for TALC to ultimately serve as an integrated security suite along the lines of Internet Security suites from Symantec, McAfee and an advanced form of the Windows Security Center [20].

The TALC system is composed of four modules, *Information Source Module*, *Correlation Module*, *Visualization Module* and *Control Module* linked together by a *Communication Manager* that allows modules to pass messages to each other. Each module exposes hooks and registers callback functions with the other modules for communication. The Information Source module detects events from the host and the network and processes them into XML data that can be exchanged with other modules. For example, the software vulnerability detection features described in this paper are implemented as a custom Information Source module, which generates data by using Hijackthis [19] (a tool that scans the registry for references to installed programs) and a series of other system scans (such as programs on the Start Menu, or on the user's desktop) to identify installed software.

The Correlation module interprets this information by aggregating the data from the different Information Sources. For the software vulnerability detection incarnation of TALC, the Correlation module correlates the information from the system scans with data pulled from the NVD. The Correlator performs a version match with the list of vulnerable software from the NVD database, ascertains the severity of the threat (Mild, Medium or Severe), and records the website indicated by the NVD to contain information necessary to resolve the vulnerability. This is fed to the Visualization and Control modules.

The Visualization module is responsible for information presentation. As described earlier, our current visualization module presents software vulnerabilities as graffiti rendered onto the user's desktop. Other visualizations are possible; for example, one visualization we have explored renders vulnerabilities as pieces of garbage piling up along the bottom of the user's screen; one could also create visualization modules that use standard pop-up dialog boxes to notify the user of threats.

Finally, the Control module provides the means by which the user can act upon the information presented by the Visualization module. In our current system the Control module opens up websites that lets the user download a patch to fix the software vulnerability. The Control module can be easily extended to give a user control of different security software such as their firewall.

## EVALUATION

We performed a deployment-based study of TALC to determine its efficacy in providing better awareness of software vulnerabilities, and in incenting users to rectify those vulnerabilities. Our study structure consisted of a two-week deployment period during which TALC was in operation on users' primary machines. Logging features in TALC reported on users' use of the system so that we could collect quantitative data on their actions. We also collected data from pre- and post-study questionnaires to get qualitative data about users' perceptions of the software.

Participants were recruited from a non-university context. After consent was obtained (but before any other participation in the study), users were sent a link to an online questionnaire that tested them on their awareness of computer security concepts and threats, as well as their expertise in general computer usage.

Once users completed the pre-test questionnaire they were emailed a link to the installation file and they were asked to download and install TALC. Our participants ran a specially instrumented version of TALC on their personal

computers for two weeks. Every week, TALC would upload the data it had collected on how users interacted with it.

When the participants had been running the program for two weeks, they were sent a link via email to a post-study questionnaire to allow us to get data about their subjective experiences of using the software.

### The Participants

Ten participants finished the study successfully. Seven more started the study, but dropped out for a variety of reasons, including changing their mind about participating in the study before downloading the software, and because of installation problems. One participant uninstalled the TALC software before the two weeks were completed; we include data from this subject in the results presented here: one of the things we hoped to discover was whether we had correctly adjusted TALC to motivate users without being annoying, and so results from users who ceased using the software had the potential to be especially illuminating.

The ages of the participants ranged from twenty-three to thirty-five, with an average age of twenty-six. Of those who completed the study, four were women and six were men.

Although the absolute number of participants is smaller than would be typical in a controlled lab study, the intent with our evaluation was specifically to engage users in a real-world deployment of the system over a sustained (half month) period of time, a style of evaluation that we believe is necessary for ecological validity. While lab-based studies can easily engage substantially larger numbers of users, these studies have problems in the security context, particularly around artificial experimental scenarios that are removed from users' day-to-day experiences, and also may overly prime subjects' orientation toward security. We therefore believed that a deployment study, on users' own computers, confronting unknown usage contexts and uncontrolled software vulnerabilities, was the only appropriate way to measure use.

### The Pre-test Questionnaire

The pre-test questionnaire was administered online, using a common online survey vendor. Participants were emailed requests to fill out the survey, and provided links to the survey site. The survey consisted of three demographic questions, eleven questions to determine how comfortable the participants felt using computers, and how confident they were in their computer's security. Finally there were eight questions in which the user was asked to define simple computer security related terms, to gauge their general knowledge of computer security concepts.

### The Study

There were two conditions in the study, to separate patching actions incented directly by TALC from other patches downloaded merely because users had an increased awareness of the security issues as a result of participating in the study itself. In all conditions TALC was downloaded and installed by the participants. In one condition, however, the tool was instrumented to not identify any vulnerabilities (and, hence, to not show any graffiti) during the first week; in the second condition the tool was instrumented so that it did not detect any vulnerabilities (nor show any graffiti) in the second week. Through these two conditions we hoped to isolate any potentially biasing novelty effects caused simply because subjects were participating in the study. In both cases, for the week it was active, TALC detected vulnerable software on all participants' computers, and thus presented graffiti to all users during the week it was activated. During both weeks, for both conditions, TALC continued to scan the users' systems and record vulnerabilities as well as how the participants interacted with it.

Participants were randomly assigned to a condition when they consented to be a part of the study. They were not given a link to the TALC installation file until they had completed the pre-test questionnaire. Participants were instructed to inform researchers if they had any problems installing the software packages; despite this, a number of participants did have trouble installing our prototype and yet did not contact us, which contributed significantly to the drop-out rate.

### The Post-test Questionnaire

The post-test questionnaire was administered online, again using a common online survey vendor, in the same way as the pretest questionnaire. Participants were emailed requests to fill out the survey, and provided links to the survey site, when they had run the TALC software for two weeks and their usage data had been uploaded. The survey had the same questions as the pretest questionnaire, along with the addition of thirteen questions to determine the participants' perceptions of using the TALC system over the deployment period.

## RESULTS

This section describes the results from our deployment study, and from our pre- and post-test questionnaires.

### Events Tracked

TALC kept logs of the users' interaction with it, over the two week period, and the data was uploaded twice to our server: once at the end of each week.

From the logs we categorized five types of events: *Awareness* events, *Learning* events, *Control (Fixed)* events, *Control (Ignore)* events, and *Reappear* events. An Awareness event was recorded when graffiti for a particular threat was shown to the user for the first time. Whenever our simplified description of the threat was shown to the user or when the user clicked on a graffiti, and the vendor website was displayed, it was recorded as a Learning event. Whenever users would indicate to the system that a vulnerability had been repaired and should be dismissed (through clicking the "Already Fixed" button in the TALC interface), Control (Fixed) events were recorded. If the vulnerability hadn't *actually* been fixed, a Reappear event would be recorded when the vulnerability was re-detected. Finally, if a user chose to not fix a vulnerability by clicking

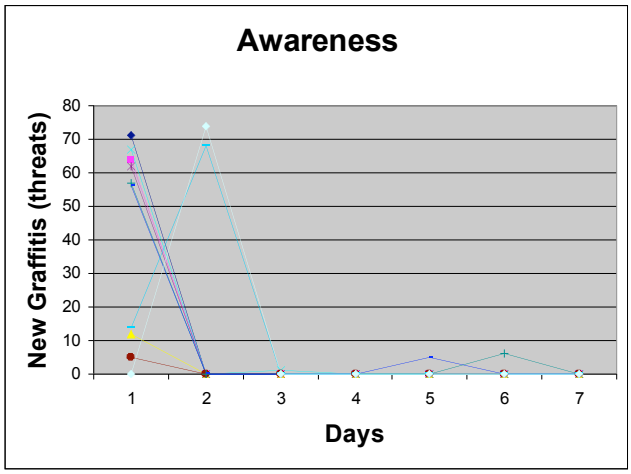


Figure 3: Awareness of new threats reported by TALC to the test participants.

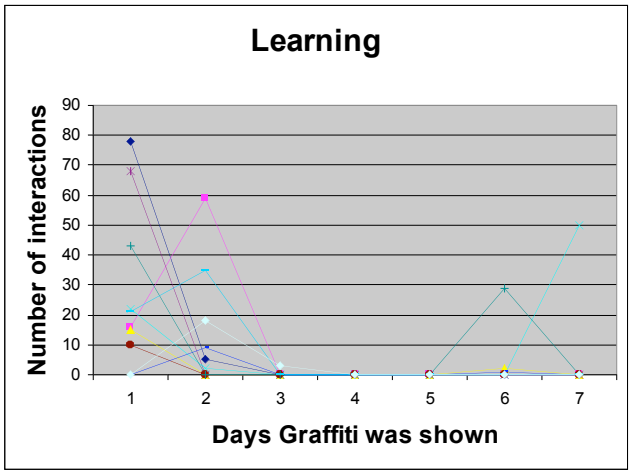


Figure 4: Learning events logged by TALC.

the Ignore button while viewing a patch website, a Control (Ignore) event was written to the log.

Each line on these graphs represents a single test participant. Figure 3 records the distribution of Awareness events—each representing a newly detected vulnerability—over the TALC active week period. The smaller spike in awareness at later stages of test period is a composite of two factors—new vulnerabilities released by NVD and new program installs by users of the software.

### Usage Patterns

The majority of user actions were taken within two days after graffiti for a particular vulnerability first appeared on the desktop: 60% of all vulnerabilities were fixed within a two-day period. However, 39% of the remaining vulnerabilities that were fixed were patched in the last two days of the test, indicating that users were patching, ignoring graffiti for a couple of days, and then coming back and patching at a later time. This is illustrated in Figure 4 below.

Recall that Learning events represent visits by users to a patch website through TALC; this figure shows the distribution of such visits. Beyond simply exploring vulnerabilities through TALC, we believe that the effects of making users aware of software vulnerabilities on their systems may have resulted in greater sensitivity to patching in general: A number of the respondents to our post-test survey reported using regular web search engines to find out more details about the detected vulnerabilities. While we were encouraged by these findings, such events are beyond the scope of the instrumentation we had in place for TALC and so we only have self-reports of such activities.

The Fixed and Ignore types of Control events are a good reflection of the effectiveness of TALC. When a participant applied a software patch, TALC did not immediately remove the graffiti; rather, the graffiti was removed during the next periodic scan of the user's system. The TALC user interface, however, provided an option to manually invoke the scan to remove graffiti for threats that had been fixed. Another option, which incidentally most users adopted, was to use the Control window to mark a threat as "Fixed" so that TALC hides the graffiti. Figure 5 shows users' usage patterns of marking threats as fixed.

A common pattern across all participants is that they tried to fix a number of vulnerabilities initially, following which there was a lull period with little or no activity; finally, several days later, there were more attempts to fix vulnerabilities. We believe this pattern indicates favorable acceptance on the part of users: rather than becoming infuriated with the notifications provided by TALC, users were "living with" the notifications for a period of several days, and then fixing them at convenient intervals. Users were able to put off patching anything for a couple of days, but were not allowed to forget about the security task to which they needed to attend. This indicates that the graffiti notification system worked well in allowing users to push back their lower priority (but necessary) security tasks until they were convenient, but while still retaining awareness of the need to perform these tasks. Furthermore, we believe the sudden flurry of activity near the end of the active week

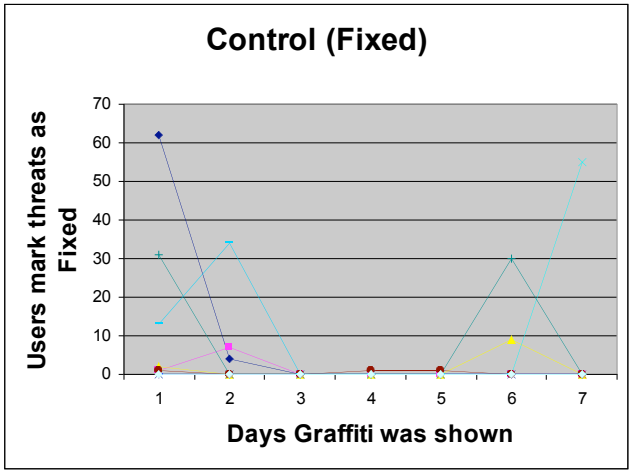


Figure 5: Participants marking threats as "Fixed".

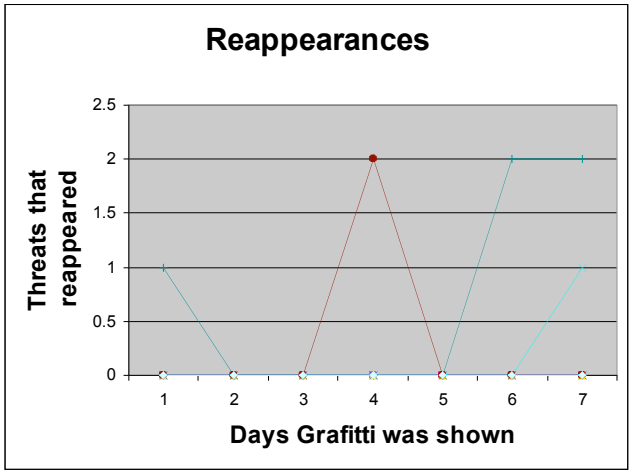


Figure 6: Threats that reappeared after being marked as "Fixed" by the participants.

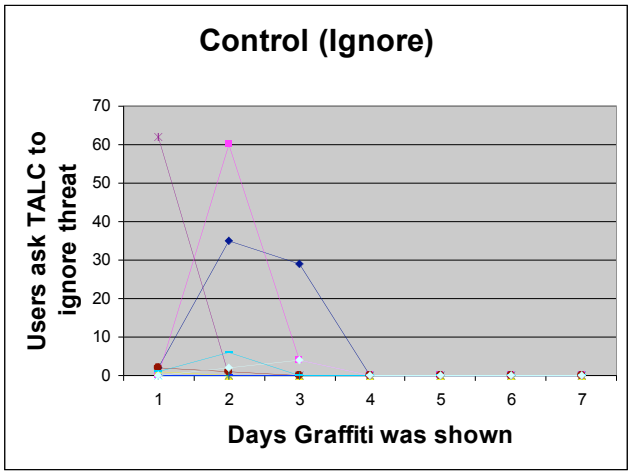


Figure 7: Participants asking TALC to Ignore a threat from subsequent scans

represents a periodic turn of attention toward patching, rather than a desire by participants to "wrap up" patching before the end of the study. Much of this activity, for example, came from the condition two participants, who still had another week to participate in the study.

In the subsequent scan cycles, TALC logged any of these fixed threats that still match the vulnerability description from NVD and logs them as a reappearance of a threat, shown in Figure 6.

We should note here that, in our current implementation, threats for which the NVD has only an advisory (meaning: for which no patch is available) are never detected as fixed by TALC. To handle such a scenario, TALC allows the user to optionally ignore these threats that have reappeared. Threats that have been ignored will not reappear in the subsequent scans unless the user explicitly asks TALC to include them in the scan. Occurrences of this event are plotted in Figure 7. The spike near the start of the test may be due to the large number of vulnerabilities detected by TALC (see below for details on the number of raw vulnerabilities found on users' systems). These Ignore events were also recorded when participants found the information provided by the patch website too daunting for them, and chose to leave the vulnerability unpatched. We discuss some suggestions for how to overcome both these shortcomings in the Future Work section.

## DISCUSSION

There are multiple useful metrics for determining what constitutes efficacy in a tool such as TALC. One such metric is whether the tool increases the *perceived* safety of users; the second is whether it increases their *actual* safety. While we evaluated for both metrics, we believe that the latter is actually the more important, since perceptions of increased safety are of little value without actually increased safety.

In the sections below we first report on TALC's efficacy in actually repairing system vulnerabilities, and then on users' perceptions of its efficacy.

### TALC Effectiveness

We were pleased to find that TALC provided a dramatic increase in the safety of users' systems during its deployment, and that TALC's notifications made a large and statistically significant difference in users' awareness and motivation to install patches.

In the weeks where TALC was dormant—meaning it was not drawing graffiti on the users' desktops—*none* of the users patched any vulnerabilities whatsoever. However, in the week when the graffiti was placed on the desktop, seventy percent of the users fixed at least one vulnerability, with an average of 24.3 vulnerabilities patched per user (averaged over all users), and an average of 34.7 vulnerabilities patched over users who patched at least one vulnerability. The number of vulnerabilities patched during the active period was found to be statistically significant ($t(9) = 2.78$, $p = 0.0216$) when compared to the dormant state when no vulnerabilities were patched.

The absolute number of vulnerabilities patched may seem artificially high, because it does not necessarily indicate the number of *fixes downloaded*, but rather the number of *vulnerabilities patched*: A single fix downloaded by the user may or may not patch multiple vulnerabilities.

At the beginning of the test, TALC found an average of 47.6 vulnerabilities on our participants' machines. All participants' computers had at least five vulnerabilities, with the most having 64 vulnerabilities. These numbers not only confirm the threat posed by unpatched software, but also users' lack of awareness about vulnerabilities on their systems for which patches exist.

Although TALC was effective in getting users to fix some of the vulnerabilities in their systems, none of the users patched their machines completely. Of the 482 vulnerabilities left unpatched, 208 (43.15%) were considered serious threats by the NVD, 88 (18.25%) were considered moderate threats by the NVD, and 186 (38.59%) were considered mild threats. The total number of vulnerabilities increased over the test period in part due to

new software being installed by our test subjects, but mostly due to updates in the vulnerability listing from the NVD. 50% of the users responded that it was very difficult to correct the security vulnerabilities reported by TALC, which we attribute to the poor usability of many of the web pages supplied by the NVD. In addition, sometimes the links provided in the NVD data were not valid. Further discussion can be found in the Future Work section below.

### User Perceptions

One of the goals of TALC was to experiment with gentle reminder functions to motivate users to take security actions. In the post-test questionnaire, when asked to suggest improvements to the program, thirty-three percent of the participants suggested various solutions for making the graffiti less invasive. However, our goal was to make TALC as motivating as possible, without being overly annoying (which could have caused users to disable the program entirely). A majority (67%) of users raised no issue with intrusiveness; further, we find it to be telling that the only user who reported disabling the TALC software during the test did so because he felt it was slowing his system down too much. The implementation of our Visualization module uses the default .NET transparency effects, which run on the CPU on systems without modern graphics cards, so this may have contributed to the problem.

Finally, with regard to the effect of TALC on *perceived* user safety, we found that four of the seven users who responded to the post-study questionnaire felt that using TALC had improved their ability to protect their computer, and that their computer was safer as a result. Although this figure does not represent universal success in increasing perceived safety, we were delighted to demonstrate any increase, since our pre-test questionnaire showed users were generally unaware of *any* vulnerabilities. We believe that this sort of awareness of risk is essential for self-managed computers (at least given today's state-of-the-art in automated security systems), and points to necessary further work in the area of conveying an accurate sense of risk or safety to users.

### Future Work

Our near-term goals concern both expanding the capabilities of the TALC framework, as well as evaluating with a larger user base on how well our visualization and control features motivate and support users to mitigate threats.

One of the common problems encountered by users was the complexity of vendors' update websites. To address this issue, we intend to add another level of proactivity in the system, to allow TALC to automatically download and install patches from a set of "common" websites, the patch processes of which can be built into the tool itself. With this addition, TALC would only present instructional web pages for late-breaking advisories that do not include a direct download link to a software patch. This feature can be accompanied by an additional layer of abstraction over the patch websites that simply asks the user to update their software to the latest version without loading the entire website containing specific details of the threat.

To prevent display of "unfixable" problems (such as advisories that do not have valid URLs), we intend to filter the advisories for common problems, and never show graffiti for the vulnerabilities reported.

We also intend to add several more information sources to the program, to exercise the extensibility features described in the Implementation section. Most important among these are the creation of modules that integrate a number of existing security tools, such as NMAP [12] and Nessus [8], to extract data about other sorts of system vulnerabilities. Specifically, we plan to use NMAP (a port scanning tool) to independently audit the user's firewall, since properly configured firewalls are very effective at blocking many automated attacks. Nessus is a tool that performs security audits by running exploit code against a user's computer. It has an active development base that releases new exploits to be used while auditing.

## CONCLUSION

We have presented TALC, a software system that assists users in protecting their computers from some of the most serious threats on the Internet, software with known vulnerabilities. TALC uses a low-intrusion notification mechanism for presenting users with information about vulnerabilities. Through the use of automated system audits and correlation against online databases, we can detect potential software vulnerabilities and give users easier mechanisms to act to repair those vulnerabilities. The extensible architecture of TALC allows it to be used to detect, visualize, and mitigate against a wide range of threats.

More generally, we believe that the approach taken by TALC may be useful in cases where 1) user motivation for a task may be low (as is often the case with security tasks), 2) intrusive or disruptive notifications may actually incent users to disable the system, and 3) there is not the need for immediate action. This combination of factors makes this class of tasks somewhat different from others (such as firewalls or background email notification) that have been widely studied in our community. We believe that the strategy of background notifications that strike a balance between awareness and annoyance to gently incent the user can be profitably applied to this class of problems.

## REFERENCES

1. Bailey, B.P., Konstan, J.A. and Carlis, J. V. (2001) The effects of interruptions on task performance, annoyance, and anxiety in the user interface. *Proceedings of INTERACT '01*, pp. 593-601.
2. Bennett, R. and Flavin, J. "Determinants of Fear of Crime: The Effect of Cultural Setting." *Justice Quarterly,* 11:3, September 1994, pp. 357-381.
3. BMC Software. Marimba Patch Management Software, http://www.marimba.com/

4. Bowling, A., Barber, J., Morris, R., and Ebrahim, S. "Do Perceptions of Neighbourhood Environment Influence Health? Baseline Findings from a British Survey of Aging." *Journal of Epidemioogy and Community Health*, 60:476-483. 2006.
5. Computer Emergency Response Team (CERT), 2006. CERT/CC Statistics 1988-2006. http://www.cert.org/stats
6. Cooke, E., Jahanian, F., and McPherson, D. The Zombie Roundup: Understanding, Detecting and Disrupting Botnets, in *First Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2005.
7. Cowan, C., Wagle, P., and Pu, C. Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade, DARPA Information Survivability Conference and Expo, 1999.
8. Deraison, R. Nessus - A Comprehensive Vulnerability scanning program, http://www.nessus.org/, 1998.
9. Dourish, P., Redmiles, D. , An approach to usable security based on event monitoring and visualization. In *Proceedings of the New Security Paradigms Workshop (NSPW)*, 2002. pp. 75-81.
10. Edwards, W.K., Poole, E.S., and Stoll, J. Security Automation Considered Harmful? In *Proceedings of the New Security Paradigms Workshop (NSPW)*, White Mountain, New Hampshire. September 18-21, 2007.
11. Ferrell, J. *Crimes of Style: Urban Graffiti and the Politics of Criminality*. New York: Garland. 1993.
12. Fyodor. Nmap - Free Security Scanner for Network Exploration and Security Audits, Insecure.org, 1997.
13. Geason, S. "Preventing Graffiti and Vandalism." *Proceedings of Designing Out Crime: Crime Prevention through Environmental Design*, Sydney, Australia. June 16, 1989.
14. Ianelli, N., and Hackworth, A. Botnets as a Vehicle for Online Crime, CERT, Request for Comments (RFC) 1700, December 2005.
15. Isbell, C. and Pierce, J. An IP Continuum for Adaptive Interface Design. In *Proceedings of HCI International*, 2005.
16. LaMacchia, B.A. Security Attacks and Defenses, in 47th Meeting of IFIP WG 10.4. 2005.
17. McAfee AVERT Labs. *SAGE. Security Vision from McAfee AVERT Labs*, July 2006.
18. McCrickard, D. S., Chewar, C. M., Somervell, J. P., & Ndiwalana, A. A Model for Notification Systems Evaluation—Assessing User Goals for Multitasking Activity. *ACM Transactions on Computer-Human Interaction (TOCHI),* 10 (4), 2003.
19. Merijn. HijackThis . http://www.spywareinfo.com/~merijn/programs.php.
20. Microsoft. Manage Your Computer's Security Settings in One Place with Security Center, http://www.microsoft.com/windowsxp/using/security/internet/sp2_wscintro.mspx.
21. Microsoft. Programs that are known to experience a loss of functionality when they run on a Windows XP Service Pack 2-based computer, http://support.microsoft.com/?id=884130.
22. Morin, K., Hayes, E., Carroll, M., and Chamberlain, B. "Selected Factors Associated with Students' Perceptions of Threat in the Community." *Public Health Nursing*, 19:6, pp. 451-459, Nov. 2002
23. Moskowitz, C.L.a.C. Simple Desktop Security with Chameleon. in Lorrie Faith Cranor, S.G. ed. *Security and Usability*, O'Reilly, August 2005.
24. National Institute of Standards and Technology (NIST). National Vulnerability Database, http://nvd.nist.gov.
25. National Institute of Standards and Technology (NIST), 2002. The economic impacts of inadequate infrastructure for software testing. Technical Report 02-3, May 2002. This report estimates damage from attacks exploiting software vulnerabilities at $60 billion/year.
26. Rafail, J. Cross-Site Scripting Vulnerabilities, CERT Coordination Center, 2001.
27. Redstrom, J., Skog, T. and Hallnas, L. Informative Art: Using Amplified Artworks as Information Displays, in *Proceedings of the Designing Augmented Reality Environments Conference '00*, (Elsinore Denmark, 2000), 103 – 114.
28. Symantec Internet Security Threat Report, Volume IX. www.symantec.com/enterprise/threatreport/index.jsp.
29. US General Accounting Office (GAO), 2003. "Effective Patch Management is Critical to Mitigating Software Vulnerabilities." Testimony before the Subcommittee on Technology, Information Policy, Intergovernmental Relations, and the Census.
30. Zurko, M.E. User-Centered Security: Stepping Up to the Grand Challenge *ACSAC*, 2005.