UCL DEPARTMENT OF COMPUTER SCIENCE

Research Note

RN/17/03

No Good Reason to Remove Features: Expert Users Value Useful Apps over Secure Ones

February 13, 2017

Steve Dodier-Lazaro, Ingolf Becker, Jens Krinke, Angela Sasse

Abstract

Application sandboxes are an essential security mechanism to contain malware. Yet, they are seldom used on Desktops. We hypothesise this is because sandboxes are incompatible with plugins, and with APIs used to implement a wide variety of Desktop features. To verify this, we interviewed 13 expert users about their app appropriation decisions, and illustrate how they recruit values like usefulness, productivity or reliability in their decisions. We found that *(a)* security is an unimportant factor for appropriation; *(b)* plugins considerably support productivity needs and *(c)* users may abandon apps that remove a feature, especially for feature removals justified by security. Productivity-oriented expert Desktop users place more value in a stable user experience and in having flexible apps than in security benefits. Sandboxing thus conflicts with their values. We conclude that for sandboxed apps to be systematically adoptable by expert users, sandboxes must no longer require the sacrifice of plugins and features found in Desktop apps.

INTRODUCTION

Sandboxes are security mechanisms that execute processes in an entirely controlled and isolated environment [14]. They are typically used to isolate apps from one another on operating systems (OSs). They protect users both against malicious apps and against exploits targeting vulnerabilities in legitimate apps. Sandboxes have become an essential building block of modern OSs [2, 3, 15, 23]. However, they prevent the implementation of a large number of features in apps, because the methods used to implement those features are also useful for malware writing. This results in sandboxed apps being more secure, but less featureful, than unsandboxed ones.

The security benefits of sandboxes are tangible. On Mobile OSs, all apps are sandboxed, which prevents malware-ridden and malicious apps from affecting other apps on the system. On Desktop OSs, however, sandboxes are only partially deployed. Desktop developers struggle to make their apps compatible with sandboxing without sacrificing important features and plugins. Many ultimately opt out from supporting this security feature [16, 9, 18, 24, 34]. Plugin infrastructures (which allow third-party developers to augment an app with additional features or user experience improvements) and features such as emulating keyboard input, screen sharing, audio recording, inter-process communication and bulk file processing are forbidden in sandboxes to prevent malicious behaviours, but they are sometimes too critical for apps to abandon [26, 39]. These incompatibilities are not, *per se*, technological constraints that cannot be overcome. They are design decisions made by sandbox designers. Instead, designers could have chosen to complicate sandboxed apps' security policies to support those potentially dangerous features.

On Windows, many popular apps like Dropbox, Steam, iTunes, Google Drive, VLC, Office, Photoshop, etc. are not sandboxed, or only in rudimentary versions with missing features [17, 29]. Tech reporters argued that sandboxed apps are rarely downloaded and used on Windows, as they lack critical features and degrade productivity [6]. After five years, the adoption of sandboxing stagnates on Windows, and even dwindles on OS X where developers have publicly announced abandoning the Mac App Store [24, 39, 34]. On Linux Desktops, sandboxed app stores exist [10, 12, 8], but to the best of our knowledge, none have a substantial user base.

Consequently, Desktop users are not currently taking advantage of the security benefits of sandboxes, despite being exposed to phishing attacks, malware, ransomware, etc. Still, many productive activities such as software development, complex information work, data science, film or music making, etc. require the use of Desktop OSs.

Moreover, assuming sandboxing meets usability requirements, users still need to either abandon their current apps in favour of new, sandboxed apps, or accept updates to their current apps that introduce sandboxing. In other words, sandboxed apps must not only be usable, but also appropriated, which involves adoption, adaptation and retainment decisions – there might be unknown challenges to appropriation, since such decisions were not examined in past research [28, 32].

We hypothesise that developers refuse to support sandboxing because it would degrade what makes their apps valuable to their users. Our analysis of developer discussions on sandboxing revealed two main issues: certain types of features cannot be implemented in sandboxed apps, and sandboxed apps cannot have plugins. We also investigated other potential tensions between security and the users' desires and preferences. If the consequences of sandboxing upset users or make apps useless, it would explain why developers are reluctant to support it.

To answer these questions, we performed interviews with 13 expert users to explore the values they seek to fulfil when they make choices about apps. We aim to unveil the *de facto* requirements that sandboxed apps must meet in order to entice user adoption, support users' needs of app adaptation, and prevent app abandonment. Besides, we analysed how our participants react to feature loss, in order to help sandbox designers recognise the extent of the gulf between sandbox capabilities and user demands.

Contributions

We show that our users struggle with explaining and accepting feature loss, and may choose to abandon apps that remove features – especially for security reasons. We show that plugins are useful and valuable to expert users, and are a crucial way to improve their productivity. We also show our participants do not consider security as a prime factor in their decisions related to app appropriation.

We also make the following minor contributions: we perform a value-sensitive analysis of app adoption, adaptation via plugins and abandonment. We find that different values underpin each of these processes, and that the values recruited to think about content consumption and production apps differ. We identify shortcomings in past usable security research: temporal aspects of appropriation (e.g. use of plugins, which address issues that were experienced in use and reflected upon by users) can only be studied in-the-wild; and participants' appreciation of security must not be distorted by priming.

Outline

We first present relevant research. Next, we explain our study design and research questions. Then, we present our value analysis of three aspects of app appropriation. We continue with a detailed analysis of participants' reactions to feature loss. We finish with a list of limitations, and conclude with a summary of our findings and open problems.

BACKGROUND & RELATED WORK

Usability evaluations of security mechanisms are mostly restrained to their user interfaces. We argue there is more to technology adoption than usable interfaces. If a tool does not perform a function that is useful to users, or if this function conflicts with other valued artefacts, the tool may be ignored. This is why Smetters and Grinter [35] have called for usable security to ensure that designed systems are useful.

Mathiasen and Bødker [21] examine usable security from the lens of experience-driven design [22]. They "concern [themselves] with how, on the one hand, the use experience is determining the security technology, while on the other hand,

the security technology resists, constrains and directs the use experience". This approach calls for security technologies that build upon users' past experiences and that fit with the use experiences they confront themselves to. By framing sandboxing as an appropriation problem, we can focus on the compositional and spatio-temporal aspects of user experience, which are usually ignored in usable security research.

The Usability of Sandboxes

Research on sandboxing focuses on technical aspects like what policies can be enforced [19], or how efficiently [11]. Only two usability studies of sandboxes exist [28, 32]. Both had participants perform scripted scenarios in a lab, emulating basic app interactions. These studies are not representative of the complexity of apps in the wild. Expert users may rely on features that are more demanding on security policies, or sometimes not possible to formulate safely with current app sandbox models. These differences in technological needs are masked by seemingly successful usability studies.

On Windows and OS X, sandboxing is only offered to applications that are distributed via the official app store, and which must respond to other criteria in terms of content, UI and libraries used. This makes it impossible to infer knowledge about the impact of sandboxing specifically from usage statistics of store apps on these platforms.

Value-Sensitive Design

We did not want to just document our participants' preferences, but understand *why* they held such preferences. Value-Sensitive Design (VSD) [13] is a methodology that reveals values involved in user behaviours and the frictions between them. It combines three forms of analysis. Conceptual analysis is used to identify stakeholders, their goals and potential value tensions between them. Empirical analysis reveals tensions in studied environments where technologies are deployed. Technical analysis probes how artefact designs position themselves with regards to values and value conflicts. We used a VSD conceptual analysis to design the interview we report on, and an empirical study to model the values involved in app appropriation and relate them to security.

Research on App Updates

Vaniea et al. [42] examined how novice Windows 7 users decide to update their apps. They found novice users avoid updates by fear of losing functionality or having to adapt to UI changes. We study expert users rather than novice users, and our investigation pushes beyond app updates to include other facets of appropriation. However, we observed similar behaviours between our cohort and the authors'. Neither group actively considered security when thinking about apps that do not interact with the Internet. However, our participants were less reluctant to modify their productivity apps, which we attribute to the higher complexity of their needs. They also preferred and used automatic updates, which we credit to the app store they used distinguishing security updates from major functionality updates which are more disruptive.

Rejecting updates exposes users to accrued security risks. The interplay between (non-security-related) technology ecosystems and user activities hence directly mediates users' ability to engage in secure experiences, without them realising. Security hygiene has been shown to emerge from holistic experiences that were designed to support security, rather than solely from the design of standalone security artefacts [21, 27].

Vaniea and Rashidi [41] also collected records of positive and negative experiences of app updates. They found that user are concerned about four types of externalities with updates: features being removed, and reliability issues being introduced including apps becoming slower, buggier, or requiring more system resources. We found our participants had similar reasons for disliking and abandoning apps after an update.

STUDY DESIGN

We aim to identify how sandboxes clash with the needs of expert, productivity-oriented users. We performed semi-structured interviews with 13 users about the apps they use.

Research Questions

Feature loss and plugin loss are externalities of sandboxing that developers expect and dislike, and thus focus most of our investigation on these aspects. However, other tensions might yet have to be uncovered. We hence explore the relationship between users and their apps more thoroughly, including situations like app adoption and abandonment which are have been ignored in past studies. We treat plugin usage as acts of app adaptation, and thus include their use in our value analysis. If the presence of features emerges as an important value for users, and if plugins play a distinct and important rules in users' practices, it would corroborate developers' worries about these two aspects of apps that conflict with sandboxing.

We first investigate what users *value* and prefer in their apps, and the relation between these values and security. Our research questions are:

RQ1: Which values drive app appropriation behaviours? Is security one such value?

RQ2: How much do expert users rely on plugins? What value do plugins provide to expert users?

After that, we turn to how users relate to and *react to* feature removal in their apps. We discuss their own experiences and beliefs, and then explore how they make sense of feature removals motivated by security reasons.

RQ3: Is feature loss acceptable? How does it impact users' choices of apps and practices?

RQ4: How does security-motivated feature loss differ from other types of loss with regard to acceptance and reaction?

Data collection and coding

We performed semi-structured interviews centred around participants' usage of apps, how they manage and value their apps, and about their information management and security strategies. The interviews lasted 40 minutes to 1:50 hour (median 1:14 hour), and we collected 81 to 227 statements per participant (median 140). The interviews concluded a period of five weeks where we observed our participants' practices, and during which they completed a diary of their activities. We used the diary data in the interviews to help participants

recall choices about specific apps and to choose which apps to talk about when querying participants about feature loss.

We coded our data separately for the value analysis and questions about feature loss. In the next section on value analysis, we allocated all participant's statements for each topic to characteristics of the apps that they relate to (we call those *app traits* in this paper), e.g. : apps being slow or costly, or the fact that an app offers new features. We regularly re-coded previous answers and refined app traits as we went along, until all participants answered could be unambiguously classified. We then mapped these app traits to the value they support, to enable a value-sensitive empirical analysis of participants' behaviours. In the section on feature loss, we used Grounded Theory's open coding [40] to identify themes in participants' answers, e.g. how they made sense of feature loss statements or the expected compensations for feature loss.

Self-reported data suffer from accuracy issues. To eliminate potential demand traits biases [25], we only retained strong statements – which participants justified or supported with prior experiences. We eliminated 18 hypothetical, vague or contradictory statements, and used 201 in our findings.

Recruitment and Demographics

We advertised our study on a Reddit community dedicated to Linux. We used Linux users because participants were recruited as part of a larger field study, parts of which include deploying software components that cannot be written for closed-source OSs. Linux is for this reason the *de facto* standard OS for systems research. We paid participants £20 for participating to the interview this paper is based on, out of a total of £100 for participating to the whole project.

We recruited 13 Xubuntu users from 7 EU countries and from the USA, aged between 18 and 54, representative of Desktop Linux users for age, occupation, gender and degree of Linux proficiency. Most describe themselves as expert users, except P6 and P12 (beginners), and P3 and P10 (IT professionals). P10 and P13 are security experts, and P12 attends security classes. Our participants include a Web developer, two (adult) high school students, two tech support representatives, a musician, a consumer retail employee, a student teacher, a sales engineer and four computer science students. 8 of them write code, 7 perform information work, and 7 produce media content (e.g. graphics, audio, video, scores, photos).

Use of deception

We told participants the study focused on their multitasking habits, to avoid non-respondent bias from participants with limited motivation to engage with security, and social desirability biases and demand trait biases during the study. We chose multitasking to attract participants who have a need for productivity, as opposed to leisure users of computers. We revealed the deception to participants near the end of the interview. Unless mentioned otherwise, all the data we use in this paper was obtained before we revealed the deception.

Our institution's Research Ethics Committee approved this study.

VALUE-SENSITIVE ANALYSIS OF APP APPROPRIATION

Sandboxes can make an impact in terms of everyday security only if they are *used*, rather than merely *usable*. To this end, we aim to determine how sandboxing interplays with three aspects of app appropriation: adoption, adaptation and retainment. Sandboxes may conflict with users' ability to obtain features and may incur a performance penalty. If users' adoption and abandonment behaviours are driven by the presence or absence of features and by performance considerations, then sandboxing will conflict with users' main decision factors. This could lead to sandboxed apps being adopted less often, or apps being abandoned after they become sandbox-compatible.

Besides, sandboxes prevent apps from providing plugins. Plugins are an integral part of how apps can be adapted to better suit workflows. Users of plugins must compare the benefits afforded by plugins with the sandbox's benefits and decide whether to adopt or circumvent the sandbox, a process described by the 'compliance budget' model [5]. We aim to find out where plugins are used, and what value they provide.

Method

We classified participants' statements on how they appropriate apps and on the plugins they use, based on the app traits they relate to (e.g. "Ad-blocking" or "Access to content" for plugins; "Unresponsive UI" or "Privacy Issues" for app abandonment). For plugins, we paid attention to their *reported purpose*, e.g. P11 uses a VPN service to access foreign media rather than for security. When participants added or replaced components of their Desktop Environment (DE), we recorded those events as DE plugins.

Next, we categorised traits into values: *usefulness*, *security & privacy*, *usability*, *productivity*, *credibility*, *affordability*, *mobility*, *stability* and *flexibility*.

We chose values to highlight known tensions in the usable security literature (*security* vs. *usability* [1], *usefulness* [35] and *productivity* [4]). We also captured the concern of involved stakeholders (e.g. the risk of feature removals affecting *credibility* for developers and *usefulness* for users). To have a more fine-grained view of value tensions, we defined UI usability as learnability, memorability and subjective satisfaction. We included absence of errors in *reliability*, and UI efficiency in *productivity*. We did not categorise the 'ad blocking' trait because participants did not always tell us *why* they used ad blockers (e.g. for privacy, performance, or subjective satisfaction). In most figures in this section, we use colours to help readers identify which traits relate to which values.

We classified apps into categories: browsers, communication apps (email and messaging), file sharing apps (cloud storage and torrent), media consumption apps (e.g. music and video players, news aggregators, etc.), media and document editors (e.g. Office, audio, video, image editors), code editors, DEs and security apps. When a statement refers to an app's feature or to a past experience with an app, we assign it to the category that fits the app.

We first discuss app adoption and app abandonment, and then summarise our findings on plugins. We continue with values

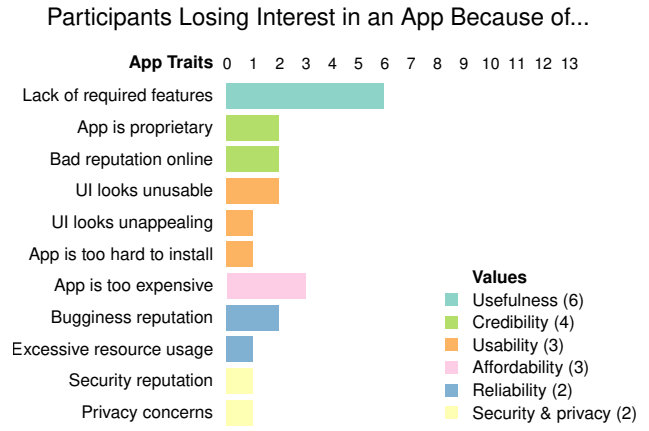

Figure 1. Participants decided not to install potential new apps primarily because they lacked a required feature. Other reasons revolve around *Credibility* and alleged *Usability* and *Reliability*.

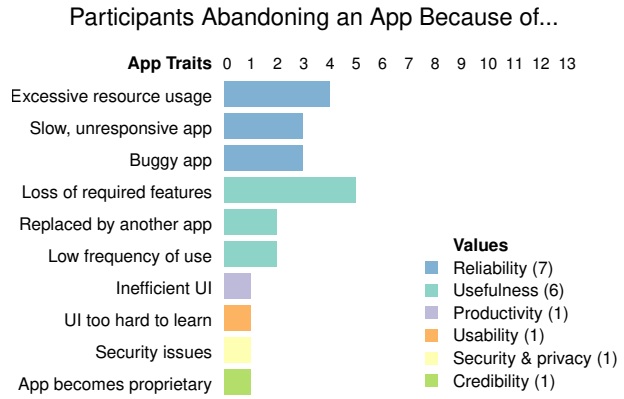

Figure 2. Participants stopped using applications primarily because of *Reliability* issues: bloated apps, unresponsive or buggy UIs. Apps also fell out of use, or lost required features after an update.

that transcend the whole of appropriation, and we finish by comparing productivity and media consumption apps.

App Adoption and Abandonment

We look at the values governing app adoption and app abandonment, in order to discover potential challenges during the transition to sandboxed apps. When developers port their apps to a sandbox, externalities can include features being incompatible, loss of plugins or performance degradation. They must decide if those changes will put users off from adopting or continuing to use their app. Hence, we asked participants what would convince them not to try a new app, and what would convince them to abandon an app they are using.

We classified participants' answers by matching their justifications to app traits (e.g. uninstalling an app that is "buggy"). When participants weave multiple reasons together (e.g. P5 ignoring an app that is neither "pretty" nor "ergonomic"), we counted half an answer for each corresponding trait. When they report multiple examples of past decisions for the same app trait, we count multiple answers. When they repeat the same example, we count a single answer.

Losing Interest in Potential Apps

We recorded 20 statements of interest loss. P4 gave no answer, and P2's answers were too weak to be included.

As Figure 1 shows, half of our 12 respondents stopped considering an app because it lacked a feature. Feature loss is a possibility when porting an app to a sandbox, either because the feature relied on privileged operations (e.g. bulk file processing, access to hardware, IPC) or on libraries that are themselves not compatible with the sandbox. Thus, if an app developer removes a key feature because of sandboxing, fewer users will adopt their app in the future.

P10 mentioned avoiding apps that have a reputation for "breaking other programs somehow" or "security stuff". He also avoids apps that are hard to install. Apps with such a reputation might benefit from being sandboxed owing to the benefits of app stores. Ultimately however, sandboxes appear more detrimental than beneficial to adoption for our cohort.

Abandoning a Current App

We also analysed what reasons participants have to stop using their current applications, to identify the impact of sandbox introduction for the current users of an app. 11 participants provided 21 statements on app abandonment. P2's data was again removed.

Figure 2 shows that *Reliability* is the primary factor for app abandonment: participants stopped using apps because they became too slow, buggy, or used too much RAM.

Usefulness follows in users' reasons for app abandonment. It is by changes in apps or in user needs. Two participants no longer needed an app, and two had a better replacement available. Five abandoned an app because it was missing a feature (in four cases, it was lost to an update; in one case, it was only partially implemented).

Security was mentioned only once spontaneously as a good reason to abandon an app. Two other participants stated security was a good reason after we accidentally primed them.

Using Plugins to Customise Apps

Expert users commonly install plugins on their apps to improve them. Plugins are routinely found on browsers, but also code editors, media editors, information work apps, communication apps, media players, etc. They are written by third-party developers, and they complicate the tasks of code verification and code signing for OS distributors who provide app stores. They are banned from the Windows App Store and on Mobile platforms, and partially banned on the OS X App Store [33]. Browsers, to retain the ability to provide third-party plugins, run unsandboxed both on Windows and OS X.

Our participants reported using 73 plugins (2 to 9, average 5), for all categories of apps except media consumption apps (see Figure 4). When asked, seven participants mentioned 11 additional plugins they would like to have. Participants plausibly had more plugins installed than they recalled, as many Linux productivity apps and media players are distributed with some plugins enabled by default. If all Linux apps were sandboxed, our participants would miss out on a significant part of their user experience due to the unavailability of plugins. In this

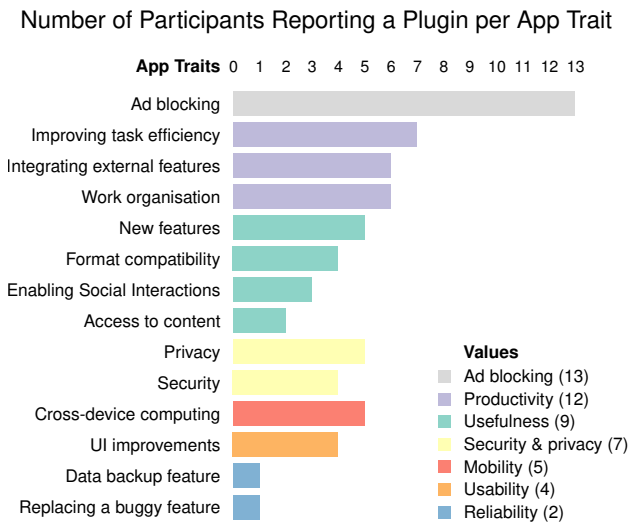

Figure 3. The plugins installed and wanted by our participants primarily support *Ad-blocking*, *Productivity* (task efficiency, external features, work organisation) and *Usefulness* (new features, format compatibility, access to content, social interactions).

section, we document what plugins are used for to understand how users would be affected if they chose to adopt sandboxed apps. This informs us on the values that security mechanisms compete against when they compromise the ability to have plugins.

Desired Plugins and Features

We asked participants to imagine an additional feature or plugin they would like to have, to check if specific types of features are in demand, or if plugins are wanted for specific app categories. We found that the 73 installed plugins and 11 desired plugins and features were similar in terms of the values they support. Consequently and for space reasons, we discuss 'installed plugins' and 'desired plugins' together in this paper.

Two differences stand out. Firstly, all participants already had ad-blocking plugins, so none asked for additional ad-blocking features. Excluding ad-blocking, the values embodied in installed and desired plugins overlap by 79.9%.

Secondly, plugins were slightly less in demand for browsers and code editors, and more for DEs, as shown by Figure 4. Our participants wanted plugins for all sorts of apps, not just browsers. Thus, a regime of exception for browsers (e.g. allowing unsandboxed browsers with plugins on app stores where all other apps are sandboxed) will not suffice to satisfy user demands for plugins.

The Role of Plugins

Plugins were predominantly used for browsers, but also for content production apps such as code or image editors and for communication apps. The features provided by plugins supported a variety of app traits, e.g. making an app compatible with a new format. Our classification aims to show what exactly participants would lose if plugins were removed. Some types of users or some apps' userbases may be more affected than others. We highlight the app traits for which sandboxes

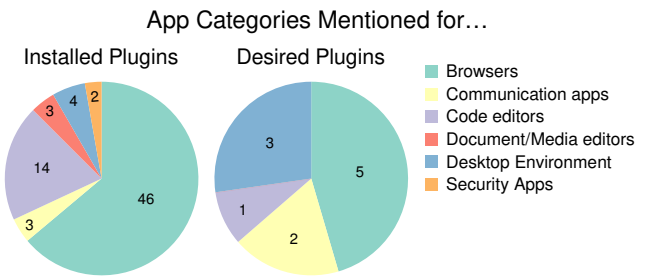

Figure 4. Browsers and code editors dominate installed plugins and desired plugins. Demand was high for DE plugins, too. Participants did not use or want plugins for media consumption or file sharing apps.

may be able to replace plugins with other techniques. We counted how many participants mentioned each trait and assigned traits to values, as shows Figure 3. The Ad-blocking trait was mentioned by all participants and not classified into a value due to its unique nature.

Plugins mostly support the *productivity* value, with three traits relating to it. Firstly, plugins help participants perform small tasks faster, e.g. code snippets or tools to accelerate browsing through Web pages. Secondly, they integrate features normally found in other apps to make them faster to access, e.g. image processing features in browsers or source version control in code editors. Thirdly, plugins help participants organise their work sessions in browsers, DEs and code editors, e.g. tools to manage tabs or improve window placement.

Plugins also support *Usefulness*, with traits such as the compatibility with new document formats, enabling new social interactions, granting access to copyrighted content, and with the introduction of new features. *Security* plugins consisted of script and Flash blockers, HTTPS Everywhere, and a password manager. *Privacy* plugins comprised end-to-end encryption for instant messaging and email apps and of plugins to prevent user tracking on the Web and on Facebook. Sandboxes can partially emulate some features of network security plugins, albeit without proper integration into apps' UIs. They cannot compensate for the loss of plugins in the *Usefulness* category.

Only the second type of productivity plugins could be replaced with methods like Inter-App Communication or UI embedding [30] by OS designers to allow apps to share and reuse features. The other types of productivity plugins, and the usefulness plugins, cannot be replaced. Most of the value provided by plugins would be lost if they were removed.

Accounting for Productivity Apps

Our participants used plugins for code editors and document and media editors, as well as DEs and browsers. We call both editor categories 'production apps' – apps to produce value in productivity contexts. Browsers, DEs and communication apps are hybrid, relevant to all sorts of use contexts. Media consumption apps (music and media players, online social networks, news aggregators, etc.) are, themselves, rarely ever useful in productivity contexts. Even though plugins are available for most of the media consumption apps mentioned by our participants, none of them used plugins for this category. Thus, plugins are particularly in demand for production apps.

Moreover, some types of practices rely on plugins more than others. $2/7$ information workers used plugins for document and media editors, while $6/8$ coders used code editor plugins. Code editors were the second most popular recipient of plugins, with over twice as many plugins as DEs, the next app category. The *Productivity* value dominates especially in code editor plugins ($7/15$ mentions for this category) and DE modifications ($5/7$). Therefore, users of code editors are particularly dependent on plugins to boost their productivity. They would be more affected than others by plugin loss.

Other app categories use plugins, for other reasons. Participants mentioned security and privacy plugins, but only for browsers and communication apps. As Vaniea et al. [42] argue, users express security needs only for apps that visibly interact with the Internet. They may refuse to degrade their user experience in production apps in exchange for security, as they have no perceived security need for such apps.

Values Driving Appropriation over Time

We recorded other value statements made by participants that are not specific to adoption, abandonment or plugins. Two values were frequently mentioned: stability and flexibility.

6 participants expressed, in 8 statements, discontent when their user experience is disrupted by changes in apps, therefore preferring *stable* experiences. P7 and P5 expressed disbelief about feature removal. P5 said: "If there is a need and there something covering this need, if you remove it it's really hard to explain to your users that it's just not there any more". Three participants were attached particularly to a specific feature (e.g. the ability to browse books or albums by their cover for P5, or the reopening of documents at the page they were last closed for P10) while we discussed their work habits. Finally, P13 expressed not wanting to change the apps he was habituated to, and disliking when those apps' UI changed after an update.

4 participants also praised, in 6 statements, software that is *flexible* and can be adjusted to their needs. P4 and P12 told us how they take advantage of settings and plugins to speed up keyboard-driven workflows. P4, P5, P12 and P13 mentioned customising applications like their document editors or DE. P5, for instance, says "I have been able to basically make my own toolbars with everything that I use. That's really flexible. [...] And it's pretty much the same idea in all applications".

Summary of Findings

RQ1: Which values drive app appropriation behaviours? Is security one such value? We found apps are:

adopted if they are *useful*, appear *usable* and *affordable*, and have a reputation of *reliability*, *security* and *credibility*

adapted with plugins to boost *productivity* and *usefulness* and sometimes to provide *security* and *ad blocking* capabilities

abandoned when they lose their *usefulness* or *reliability*

Users also valued a *stable* user experience, and *flexible* apps that can be adjusted to their needs.

RQ2: How much do expert users rely on plugins? What value do plugins provide to expert users? All our participants used plugins – for browsers, DEs and all types of editors, but not for media consumption apps. Plugins mainly provide usefulness and productivity. They also provide ad-blocking in browsers, and security for Internet-facing apps. Few of the benefits provided by plugins could be replaced by other mechanisms, if plugins were to become unavailable.

Productivity plugins were more prevalent for productivity apps and DEs, and our participants were in demand for more productivity plugins than they already had. Thus, people who use computers for productive work, and specifically users of some types of apps, would see their productivity decrease if they no longer had access to plugins.

Implications for Sandboxing

Sandboxing threatens *usefulness* by preventing the implementation of some features, *reliability* by degrading performance and resource usage, and *stability* by causing developers to transform or drop some features. Sandboxes thus conflict with the values recruited by participants when they decide to adopt and abandon apps. Owing to their effects on plugins, sandboxes further threaten *productivity* and *usefulness*, the main values supported by the use of plugins. Developers who chose to drop features and plugins to support sandboxing will be confronted to loss of users and potential new users, according to our value-sensitive analysis.

Our participants' liking of *stability* suggests sandbox designers shouldn't expect user experience sacrifices as a prerequisite to sandbox adoption. Mobile OSs never had plugin infrastructures, and so their users have adopted what was available. Android and iOS are dominated by media consumption apps [37, 38], and since there is no plugin demand for consumption apps, plugins are not as crucial for Mobile OSs as they are for Desktops. Users might refuse to switch to sandboxed versions of Desktop apps if this means losing plugins they have already integrated into their work practices.

Plugin loss will particularly affect users with productivity goals, and some demographics e.g. users who write code (and expectedly, over demographics that were not represented in our cohort). When productivity is put in competition with security, users respond by implementing "shadow security" practices, which involve disengagement from sanctioned, verified security mechanisms, even if they do value security [20]. It is advisable that plugins be supported by sandboxes, especially since there is no technical barrier to distributing plugins on the Windows and Mac App Stores, just like standalone apps.

Moreover, sandboxes are known to impact app performance, e.g. disk throughput, RAM and CPU usage [19, 28, 36, 43]. Systematic sandboxing could be unacceptable for users who rely on e.g. high-performance computing, data science or video processing. Sandbox designers could choose to make different arbitrations between security and performance for CPU- or RAM-intensive apps, in exchange for e.g. a more stringent developer authentication process for said apps.

FEATURE LOSS

We've identified the importance of usefulness in appropriation decisions, and we know that sandboxes conflict with useful-

ness by forbidding some features. We now explore the value arbitrations made by participants when they are confronted with feature loss. We're interested in learning what characteristics of an app they consider to be more, or less justifiably important than the presence of a feature. We also query participants' reactions to feature loss in an app they use, when the cause of the loss is not explicited, and when "security reasons" motivate the loss. This informs us both on the values that users oppose to usefulness when rationalising feature loss, and on how they accept loss of usefulness.

Method

We asked participants, if a feature was removed from an application, what they would do and how it would affect them. We also asked them what good and bad reasons a developer could give to justify this change. We paid attention not to ask for a singular reason, and let participants express multiple ones. When possible, we asked participants about features they mentioned during the interview. Otherwise, we would ask about "a feature" or "the ability to have plugins" for an app they mentioned in their study diary. Most participants responded with hypothetical scenarios based on apps they used. We excluded previous answers to app abandonment questions as this would have constituted priming towards app abandonment with regard to reactions to feature loss.

We formulated the security question as such: we asked participants what they would think if a developer were to remove a feature or plugin (using the same criteria as above) "for security reasons". P12 spontaneously mentioned security as a valid reason for removing a feature, obviating the security question. P5 and P9 were mistakenly asked about justifications to feature removal after we had revealed the security deception.

We refer to answers based on participants' features as "own experiences", and answers to the security question as "security reasons". As the interviews were semi-structured, we missed some answers from participants, especially P3 and P11.

We verified afterwards if the features discussed in interviews were compatible with sandboxing: 4 concerned apps that are not sandboxable; 11 were compatible; 5 were partially incompatible; and 18 were incompatible. None of our participants demonstrated extensive knowledge of how sandboxes affect apps, later on in the interviews when we discussed security with them. None mentioned knowledge of design decisions or actual technical constraints that restrict features in sandboxes. Therefore, we do not believe that the compability of features discussed in the interviews could have influenced participants' attitudes and reactions.

Justifying Feature Removal

We wanted to know what determined whether users would accept the disappearance of a feature. If a specific reason makes sense to users, they will be less incredulous and suspicious when a feature is removed for that reason. Inversely, if users are told a feature is removed for a reason they do not understand, they might deplore the developer's decision and be more prone to switch apps.

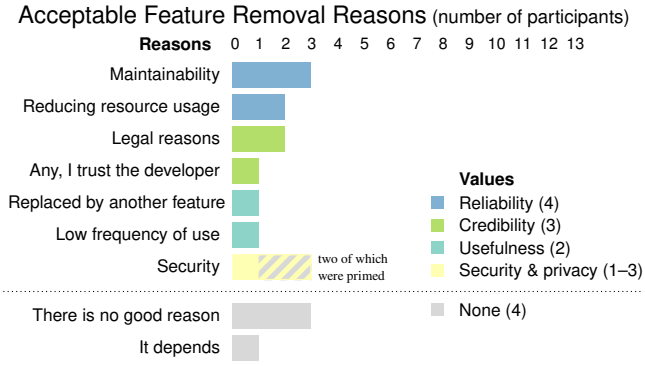

Figure 5. Number of participants citing a reason as acceptable to justify feature removal. Three participants could not come up with a way to justify feature removal. Security justifications include one non-primed and two primed answers. P4 mentioned maintainability twice.

We collected 18 reasons which participants (except P3 and P11) thought were acceptable (see Figure 5) and 8 unacceptable (Figure 6) to justify feature removals. 5 participants recalled actual experiences of feature loss, showing it is a commonplace experience, though overall participants did not find it easy to answer those questions.

Maintainability was seen as the most valid reason to remove features, by 3 participants, with 2 mentions from P4. This included removing code that was too difficult to maintain or not stable enough, or making plugins temporarily unavailable after an update. However, one of the "feature loss" app abandonment reasons we discussed in the previous section was justified with maintainability: P4 abandoned the GNOME DE because its plugins would often stop working after an update. So the reason is not unanimously accepted.

Security was mentioned thrice, albeit two times by participants whom we accidentally primed to think about security beforehand, as we forgot to ask the question about feature removal until right after revealing the security topic of the study and before discussing security practices.

Legal reasons were mentioned both as a good and as a bad justification. So was reliability, with participants claiming that excessive CPU or RAM usage were valid reasons, but excessive disk usage would not warrant removing a feature. Likewise for usefulness: P6 mentioned not caring about a feature he did not use, whereas P12 strongly opined that developers should not remove a feature used only by a fraction of their user base that he has a use for.

Participants can conceptualise why feature are removed (maintainability, legal issues, reliability, and security), but none of the enumerated reasons seem to be *always* justified. Besides, three participants thought feature removal to be inexcusable, no matter the reason. Therefore, there is no *blanket rationale* that developers can invoke to explain away a decision to remove a feature. They will invariably need to convince a majority of their expert users why a feature removal was warranted.

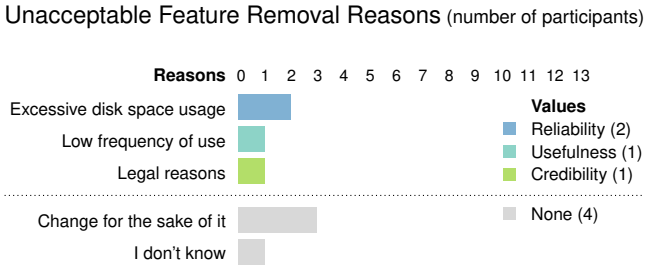

Figure 6. Number of participants citing a reason as *not* acceptable to justify feature removal. Participants disliked when developers were unclear about the reasons for a change and it appeared as being "for the sake of it".

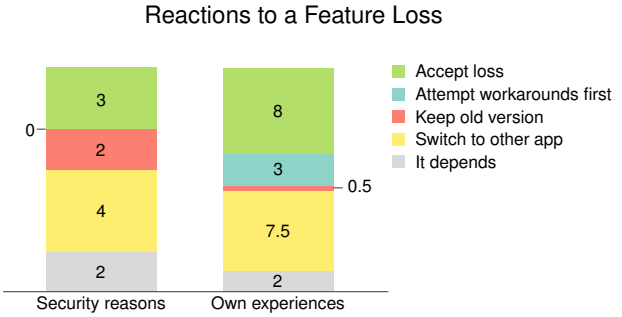

Figure 7. Participants are more likely to accept an update that induces feature loss for reasons other than security. Some will deploy workarounds to emulate or replace the lost feature, before seeking a replacement app. Over a third of participants would abandon an app that lost a feature and seek another one with an equivalent feature either way.

Security reasons

We asked eleven participants (except P3 and P11) what they would think of a scenario where a feature is removed for "security reasons".

Are security reasons really accepted?

Eight participants considered security to be a good reason to remove a feature, once we asked them. The three others did not answer the question, but described how they would analyse the feature loss instead. None found it explicitly unacceptable. Yet, only P12 mentioned security spontaneously – as well as P5 and P9 right after we primed them. Security might be a positive value to our participants, but it is not something they think about when features are affected by updates.

As we will see in the next section, participants are even less likely to go through with a feature-removing update that is motivated by security than with a generic one. This contradicts participants' apparent acceptance of security. If they truly thought security to be a valid reason, more of them would have mentioned it spontaneously, and their intended actions would match their stated belief.

Making sense of "security reasons"

Even though participants agreed security was an acceptable justification, they sounded noticeably negative about it. We had expected them to state that they would no longer use the insecure software. Instead, they showed us they would attempt to understand the announcement and to decide for themselves if they should be concerned and adjust their practice.

Participants were mostly defiant because of how they made sense of "security reasons". They understood security as *incident response*, rather than the *anticipation* of risks that have not yet materialised, or *compliance* with external constrains. Yet, sandbox feature constraints derive from risk management considerations rather than security vulnerabilities.

Three participants clearly expressed the idea that the security risk had resulted in exploitation, using words such as "malware", "breach" or "security exploit". Three more talked of a "vulnerability" or "security hole" and wondered if their data could be compromised as a result. Only P8 pondered that the feature itself might have represented a danger, without mentioning the existence of a fault attributable to the developer.

Deciding What to do About Feature Removals

How many users would abandon an app if its developers decided to remove an important feature from it? The answer to this question is relevant to developers who must decide whether to adopt feature-degrading sandboxes or not. We thus asked our participants how they would react to the loss of a feature they had previously mentioned to us, or to the loss of plugins. We sometimes asked participants about more than one feature. Figure 7 presents the 20 reactions we collected from 11 participants for feature loss in general (some participants answered for several features, P9 gave weak answers, P11 was not asked). It also shows the 11 reactions collected for security-induced feature loss from 9 participants (P1 gave two answers, and P3, P9, P11 and P13 gave none).

For updates motivated by security reasons, participants decided to stay on the old, insecure version of the app in $2/10$ cases. In $4/10$ cases, they preferred switching to another app. $2/10$ said their reaction would depend on the feature or the developer's attitude. This leaves only $3/10$ cases where participants would accept the update. This reaction contradicts our finding that nearly all participants agreed security is a valid reason to remove features. We hypothesise this discrepancy is due to usefulness taking precedence over security in driving participants' choices. Another possible conjecture is that our expert users have become *prejudiced* against security announcements, owing to dissonance between alleged and perceived security benefits in past security experiences.

The cost of feature loss was viewed as higher than the security benefits in our security question. In contrast, when we asked about feature removal in a generic update, participants valued the imagined benefits of the update more often than the feature to be removed: they would use the new version in $11/21(52\%)$ cases – including $3/21(11\%)$ cases where they would attempt to emulate the lost feature with the new version, but would switch back to the old one or to a new app if their coping mechanism fails to satisfy them. Security is, after all, a secondary goal [31, 44], so it is valued less than features which support a primary goal. Our value analysis corroborates this: factors like usefulness, productivity or reliability trump security in participants' decisions. P10 would either switch to another app or stay on the old version. In $7.5/21(36\%)$

cases in total, participants would switch to another app. In $2/21(10\%)$ cases, participants said it depends on the feature.

Getting Something out of the Loss

In both conditions, three distinct participants expected lost features to be re-introduced after some time. When a disruption is temporary, participants might tolerate it as a necessary evil. P1, P10 and P13 also expected the app to be improved in some way (e.g. reducing RAM usage, speeding up the UI, or integrating popular plugins directly into the app) in the general case. This desire for *compensation* was not seen in the security condition, as a security benefit was already communicated.

P5, P10 and P13 expected developers to explain what the vulnerability was that had been fixed. Other participants sought to convince themselves of the well-foundedness of the security reason. P7 stated he expected to be told "how much time has there been a security breach, why have they not warned me beforehand, and what happens now". P12 said "they'd have to justify it pretty well". P2, P8 and P10 said they would look into the issue to decide if they should feel concerned or not. Those participants would not trust developers' words and would confirm the existence of a security benefit before choosing what to do about the feature loss.

Summary of Findings

RQ3: Is feature loss acceptable? How does it impact users' choices of apps and practices? Feature removal has a substantial impact on users: over a third may abandon an app when a feature they used disappears. Half won't consider updating an app with a missing feature, and they may also abandon an app that loses a feature. A forth of participants expected feature loss to be temporary, and a forth also expected it to be compensated with improvements.

There is no consensus among participants over what constitutes good reasons to remove a feature. Maintainability, reliability and legal issues were mentioned, although sometimes as bad reasons too. Security was mentioned spontaneously by one participant, and after security priming by two more participants. Given the prevalence of *stability* in user values, we find feature loss hard to justify to users overall.

RQ4: How does security-motivated feature loss differ from other types of loss with regard to acceptance and reaction? When asked, our participants claim security is a valid reason to remove features. Yet, they are four times more likely to ignore a feature-removing security update than a feature-removing update with non-security motivations. Participants are two times less likely to accept a security-motivated feature loss. This illustrates well how security is a secondary goal to users.

Participants view security-motivated feature removals as incident response rather than a preventative measure. They expect developers to explain why a security risk existed and the consequences if it. Thus, developers' credibility may paradoxically suffer when they announce security improvements.

Implications for Sandboxing

Sandboxes restrain the ability to implement some features as a form of *risk management*, rather than because these features introduce systematic vulnerabilities. As our participants understand security as *incident response*, they are unlikely to attribute a sandbox-related feature loss to a fault on behalf of app developers. Besides, we've seen that there is no *blanket rationale* that developers can invoke to explain away a decision to remove a feature, since participants don't have a consistent mental model of valid justifications to feature loss. Therefore, the task of explaining a sandbox-motivated feature loss to users seems particularly strenuous and hazardous for developers.

Manifestly, feature removal can lead to significant user base attrition. As we've seen, this is more so the case when feature loss is justified by security. In competitive app ecosystems where many apps provide similar features, having to remove features from one's app might act as a strong deterrent for developers to consider sandboxing. We argue that the current restrictions on features and plugins place an unfair burden on app developers, and that sandbox designers must review those decisions rather than wait out for developers to finally 'get it' and adopt sandboxing. Presently, there are valid incentives in place for app developers to stay away from sandboxing.

LIMITATIONS

Cohort size

The field study we are running involves sustained interactions with participants, forcing us to keep a small cohort. We thus have too few participants to provide statistical significance for our results. We provide quantitative data as much as possible to allow for our results to be aggregated to future studies on this topic. Besides, we view the presentation of our method as a contribution in itself, relevant to security designers who need to study barriers to the adoption of security technologies in their app ecosystems.

Deception

We ensured the validity of our data via the use of deception. However, this means less data was available as we could not incite our participants to detail their mental models of security without drawing their attention to our actual topic of interest.

Method of report

App appropriation events are rare, and participants sometimes struggled to recall details of their past experiences. We helped them recall past events by using diary data to discuss the apps which we knew they used, and we eliminated statements where participants sounded hesitant or were inaccurate.

Linux users

We recruited Linux users. They are reflective about technology and have experience with multiple systems [7]. This is not a threat to validity, but reduces the scope of our findings to experienced and reflective practitioners. Many Windows and OS X users are experts, too – including developers, digital artists, researchers, etc. Linux users prefer software that is open-source. Thus, our data likely overstates the importance of the app traits related to proprietary licenses.

Reliability of analysis

Both our coding process and Value-Sensitive Design rely on researcher interpretation. Our choice of values reflects how we deconstructed usability into distinct actionable values (reliability, productivity, usability, flexibility). Other researchers may have used a different degree of granularity.

IMPLICATIONS FOR USABLE SECURITY RESEARCH

Some of our findings would not have been possible to make if we had stuck to the methods used in previous sandbox usability research [28, 32]. We derive methodological implications for future usability evaluations of security mediators.

Productive security is achieved over time, not in the lab

Beautement et al. [5] argue that the cost of security might be accepted during initial interactions, but rejected over time as users wear out their "compliance budget" – their ability to comply with security when the cost of it exceeds its benefits. When newly introduced security artefacts disrupt stability (e.g. with feature loss) or flexibility (e.g. by removing plugins), these artefacts cannot be declared usable solely on the basis of one-off interactions in a lab setting. Those values are fulfilled over time, and so the impact that changes in users' practices have on them must be studied over time too.

Previous usability studies of sandboxing [28, 32] failed to study how participants ultimately react to the cumulative frustrations caused by a degraded user experience, or how they can improve their productivity once sandboxes hinder apps' flexibility. Ergo, sandboxes must be introduced in-the-wild and their impact on practice monitored until they are completely appropriated or rejected by participants. Otherwise, researchers may falsely conclude that sandboxes are usable, when participants' compliance budget is exhausted in superficial interactions settings and their interaction would not have been sustained in-the-wild.

Deception is necessary to discover actual behaviour drivers

Participants overwhelmingly agreed that security is an acceptable reason to remove a feature, when we asked them. Yet, they would be less likely to continue using an app that lost a feature for security, rather than for other types of improvements. We conclude from that that querying participants directly about their attitude to security can mislead researchers into thinking that security is sufficiently valued to influence user behaviour. We've shown that explicit attitudes towards one value are not the proper measure for drivers of behaviour. Instead, researchers should focus on building value hierarchies and identifying the main values that users recruit in making decisions that impact security. This means that study designs must include deception to avoid non-respondant and social desirability biases, and to produce valid value hierarchies.

CONCLUSION

Sandboxes do not provide support for several types of features, and for plugins, resulting in second-class apps. Sandboxes also decrease app performance slightly. Sandbox adoption is low on Desktop OSs, and some developers even forsake sandboxed versions of their apps. We investigated how expert Desktop users arbitrate different values in apps, and how they cope with feature loss, to understand how they arbitrate between usefulness, productivity and security, and how likely they are to adopt or retain apps that sacrifice features for security improvements. If users are likely to abandon newly sandboxed apps, it would explain developers' reluctance to support sandboxing.

We built a model of values involved in three Desktop app appropriation processes: adoption, adaptation, and abandonment. We found that lack of features was the primary reason for users to reject a potential app, and one of two reasons (along with reliability) for users to abandon an app they're using. We also found that users like to adapt and customise their apps, primarily to meet productivity goals, especially for browsers and productivity apps like code editors. Besides, feature loss is a seldom understood phenomena that is poorly accepted by users. A non-negligible portion of our participants would abandon an app that removes a feature they use, especially if justified by security improvements.

Sandbox designers must identify the features threatened by the changes sandboxing brings about, and they must improve support for the relevant APIs so that these features survive sandboxing. They could support plugins by distributing them on app stores and subjecting them to the same security checks as apps. These corrections are essential to avoid putting security in competition with usefulness and security. Indeed, our value analysis clearly shows that security will not be privileged by expert users, and thus, that sandboxed apps are less likely to be adopted than their insecure counterparts.

In future work, we will continue to investigate how app sandboxing and our participants' digital lives fit together. We will assess the fitness of app sandboxing for the information management strategies of our participants using qualitative and quantitative data we collected, and we will investigate how many of the apps they used contain features typically threatened by sandboxing.

REFERENCES

1. Adams, A., and Sasse, M. A. Users are not the enemy. *Commun. ACM 42*, 12 (Dec. 1999), 40–46.
2. Apple Inc. *App Sandboxing*. Sept. 2016.
3. Apple Inc. *iOS Security – iOS 9.3 or later*. May 2016.
4. Beautement, A., Becker, I., Parkin, S., Krol, K., and Sasse, A. Productive Security: A Scalable Methodology for Analysing Employee Security Behaviours. In *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*, USENIX Association (Denver, CO, June 2016), 253–270.
5. Beautement, A., Sasse, M. A., and Wonham, M. The compliance budget: managing security behaviour in organisations. In *Proceedings of the 2008 workshop on New security paradigms*, NSPW '08, ACM (New York, NY, USA, 2008), 47–58.
6. Brad Chacos. *And the study says: Windows 8 users rarely touch Metro apps*. May 2013.

7. Canonical. *Ubuntu User Surveys 2012 - Part 3*. 2012.
8. Canonical. Ubuntu Core Documentation – Security and Sandboxing, 2016.
9. Dan Counsell. *Not on the Mac App Store*. Nov. 2015.
10. Docker Inc. Overview of Docker Hub, 2016.
11. Drewry, W. Dynamic seccomp policies (using BPF filters), Jan. 2012.
12. Flatpak. Flatpak – the future of application distribution, 2016.
13. Friedman, B. Value-sensitive Design. *interactions 3*, 6 (Dec. 1996), 16–23.
14. Goldberg, I., Wagner, D., Thomas, R., and Brewer, E. A. A Secure Environment for Untrusted Helper Applications Confining the Wily Hacker. In *Proceedings of the 6th Conference on USENIX Security Symposium, Focusing on Applications of Cryptography - Volume 6*, SSYM'96, USENIX Association (Berkeley, CA, USA, 1996), 1–1.
15. Google. *android: Application security*. Sept. 2016.
16. Hoffman, C. *Why the Mac App Store Doesn't Have the Applications You Want*. Mar. 2015.
17. Hoffman, C. *Why Desktop Apps Aren't Available in the Windows Store (Yet)*. Mar. 2016.
18. Ian Paul. *The 10 most glaring Windows Store no-shows*. Apr. 2013.
19. Kim, T., and Zeldovich, N. Practical and Effective Sandboxing for Non-root Users. In *Proceedings of the 2013 USENIX Conference on Annual Technical Conference*, USENIX ATC'13, USENIX Association (Berkeley, CA, USA, 2013), 139–144.
20. Kirlappos, I., Parkin, S., and Sasse, M. Learning from "Shadow Security": Why understanding non-compliance provides the basis for effective security. In *Workshop on Usable Security* (San Diego, California, Feb. 2014).
21. Mathiasen, N. R., and Bødker, S. Threats or Threads: From Usable Security to Secure Experience? In *Proceedings of the 5th Nordic Conference on Human-computer Interaction: Building Bridges*, NordiCHI '08, ACM (New York, NY, USA, 2008), 283–289.
22. McCarthy, J. C., and Wright, P. *Technology as experience*. MIT Press, 2004.
23. Microsoft. *Windows 8 Security Overview*. June 2013.
24. Milen Dzhumerov. *Mac App Store: The Subtle Exodus*. Oct. 2014.
25. Nichols, A. L., and Maner, J. K. The Good-Subject Effect: Investigating Participant Demand Characteristics. *The Journal of General Psychology 135*, 2 (2008), 151–166.
26. Peter Cohen. *The Mac App Store and the trouble with sandboxing*. Apr. 2014.
27. Pfleeger, S. L., Sasse, A., and Furnham, A. From Weakest Link to Security Hero: Transforming Staff Security Behavior. *Journal of Homeland Security and Emergency Management 11*, 4 (2014), 489–510.
28. Potter, S., and Nieh, J. Apiary: Easy-to-use Desktop Application Fault Containment on Commodity Operating Systems. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'10, USENIX Association (Berkeley, CA, USA, 2010), 8–8.
29. Riley J. Dennis. *Desktop vs. Windows Store Apps: Which Should You Download?* June 2016.
30. Roesner, F., Kohno, T., Moshchuk, A., Parno, B., Wang, H. J., and Cowan, C. User-Driven Access Control: Rethinking Permission Granting in Modern Operating Systems. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, IEEE Computer Society (Washington, DC, USA, 2012), 224–238.
31. Sasse, M. A., Brostoff, S., and Weirich, D. Transforming the 'Weakest Link' — a Human/Computer Interaction Approach to Usable and Effective Security. *BT Technology Journal 19*, 3 (July 2001), 122–131.
32. Schreuders, Z. C., McGill, T., and Payne, C. Empowering End Users to Confine Their Own Applications: The Results of a Usability Study Comparing SELinux, AppArmor, and FBAC-LSM. *ACM Trans. Inf. Syst. Secur. 14*, 2 (Sept. 2011), 19:1–19:28.
33. Security StackOverflow. *Mac App Store and Plugins*. Mar. 2011.
34. Sketch. *Leaving the Mac App Store*. Dec. 2015.
35. Smetters, D. K., and Grinter, R. E. Moving from the design of usable security technologies to the design of useful secure applications. In *Proceedings of the 2002 workshop on New security paradigms*, NSPW '02, ACM (New York, NY, USA, 2002), 82–89.
36. Spencer, R., Smalley, S., Loscocco, P., Hibler, M., Andersen, D., and Lepreau, J. The Flask Security Architecture: System Support for Diverse Security Policies. In *Proceedings of the 8th Conference on USENIX Security Symposium - Volume 8*, SSYM'99, USENIX Association (Berkeley, CA, USA, 1999), 11–11.
37. Statista. *Most popular Google Play app categories in February 2014, by device installs*. Feb. 2014.
38. Statista. *Most popular Apple App Store categories in June 2016, by share of available apps*. June 2016.
39. Steve Streeting. *Between a rock and a hard place – our decision to abandon the Mac App Store*. Feb. 2012.
40. Strauss, A., and Corbin, J. *Basics of qualitative research: Techniques and procedures for developing grounded theory* . Sage Publications, Inc, 1998.
41. Vaniea, K., and Rashidi, Y. Tales of Software Updates: The Process of Updating Software. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, ACM (New York, NY, USA, 2016), 3215–3226.

42. Vaniea, K. E., Rader, E., and Wash, R. Betrayed by Updates: How Negative Experiences Affect Future Security. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, ACM (New York, NY, USA, 2014), 2671–2674.

43. Vasudevan, A., Parno, B., Qu, N., Gligor, V. D., and Perrig, A. Lockdown: Towards a Safe and Practical Architecture for Security Applications on Commodity Platforms. In *Proceedings of the 5th International Conference on Trust and Trustworthy Computing*, TRUST'12, Springer-Verlag (Berlin, Heidelberg, 2012), 34–54.

44. Yee, K.-P. Aligning security and usability. *Security Privacy, IEEE 2*, 5 (Oct. 2004), 48 –55.

UCL DEPARTMENT OF COMPUTER SCIENCE

Research Note

RN/17/03

ADDITIONAL MATERIALS

February 13, 2017

Steve Dodier-Lazaro, Ingolf Becker, Jens Krinke, Angela Sasse

Abstract

Application sandboxes are an essential security mechanism to contain malware. Yet, they are seldom used on Desktops. We hypothesise this is because sandboxes are incompatible with plugins, and with APIs used to implement a wide variety of Desktop features. To verify this, we interviewed 13 expert users about their app appropriation decisions, and illustrate how they recruit values like usefulness, productivity or reliability in their decisions. We found that *(a)* security is an unimportant factor for appropriation; *(b)* plugins considerably support productivity needs and *(c)* users may abandon apps that remove a feature, especially for feature removals justified by security. Productivity-oriented expert Desktop users place more value in a stable user experience and in having flexible apps than in security benefits. Sandboxing thus conflicts with their values. We conclude that for sandboxed apps to be systematically adoptable by expert users, sandboxes must no longer require the sacrifice of plugins and features found in Desktop apps.

LIST OF INSTALLED PLUGINS

See Tables 1 and 2 for the list of plugins our participants declared having installed.

LIST OF DESIRED PLUGINS

See Table 3 for a list of extra features or plugins our participants would like to have.

LIST OF REACTIONS TO FEATURE LOSS

See Table 4 for the features for which participants gave us a reaction to the removal of.

LIST OF JUSTIFICATIONS FOR FEATURE LOSS

See Table 5 for the features used to discuss good and bad reasons to remove a feature. P12 gave two bad reasons for J17, resulting in 8 bad reasons in the paper for 7 features.

LIST OF SECURITY-MOTIVATED LOSS SCENARIOS

See Table 6 for the features involved when we asked participants what they would think of a security-caused feature removal. There were 12 features. P3 and P9 did not tell us what they would do, and P10 gave us two different reactions based on how concerned he would be about the security reason at hand. This led to the 11 reactions to feature removal presented in the paper.

P#	Cat.	App name	Plugin	App trait
P1	web	Chromium	AdBlock	ad blocking
P1	web	Chromium	OneTap	work organisation
P1	web	Chromium	Tool to calculate distances in pixels	integrating external features
P1	code	Sublime	Origami	work organisation
P1	code	Sublime	Code snippets plugin	improving task efficiency
P1	code	Sublime	Cooperative coding plugin	enabling social interactions
P1	code	Sublime	Themes	UI improvements
P2	web	Firefox	AdBlock	ad blocking
P2	web	Firefox	Torrent downloader	new features
P2	code	Vim	“reformatting and styling things” (name was inaudible, assumed to be a syntax highlighting plugin)	UI improvements
P3	web	Firefox	uBlock	ad blocking
P3	web	Firefox	NoScript	security
P3	code	Brackets	Colour palette	integrating external features
P3	code	Brackets	Python plugin	format compatibility
P4	web	Firefox	AdBlock	ad blocking
P4	web	Firefox	HTTPS everywhere	security
P4	web	Firefox	Pushbullet	cross-device computing
P4	web	Firefox	Vimperator	improving task efficiency
P4	DE	Xfce	DevilsPie	work organisation
P5	web	Chromium	AdBlock	ad blocking
P5	web	Chromium	HTTPS everywhere	security
P5	web	Chromium	Zotero	integrating external features
P5	web	Chromium	Hangout	enabling social interactions
P5	comm	Thunderbird	Enigmail	privacy
P5	docu	LibreOffice	Zotero	integrating external features
P5	DE	Xfce	Cairo	improving task efficiency
P6	web	Chromium	uBlock	ad blocking
P6	web	Chromium	TrackInhibitor	privacy
P6	web	Chromium	ZenMate	privacy
P6	sec	KeePass	A database backup plugin	data backup feature
P6	sec	KeePass	Favicon Downloader	UI improvements
P7	web	Chromium	uBlock	ad blocking
P7	web	Chromium	Shortcut to Calendar	integrating external features
P7	web	Chromium	Firebug	new features
P7	web	Chromium	MyTexts	cross-device computing
P7	web	Chromium	FacebookUnseen	privacy
P7	web	Chromium	NAS downloader	cross-device computing
P7	web	Chromium	Chromecast	access to content
P8	web	Chromium	AdBlock	ad blocking
P8	web	Chromium	Hoverzoom	improving task efficiency
P9	web	Chromium	AdBlock	ad blocking
P9	web	Chromium	Neutron IDE	new features
P9	web	Chromium	POSTman	new features
P9	web	Chromium	Telegram	enabling social interactions
P9	web	Chromium	MEGASync	cross-device computing
P9	web	Chromium	Hoverzoom	improving task efficiency
P9	web	Chromium	Shortcut to Google Doc	integrating external features
P9	code	Eclipse	SVN plugin	integrating external features
P9	code	Eclipse	Javadoc plugin	integrating external features
P9	code	Eclipse	UML plugin	format compatibility
P10	web	Chromium	AdBlock	ad blocking
P10	web	Chromium	Youtube repeater	improving task efficiency

Table 1. Plugins installed by participants (1/2).

P#	Cat.	App name	Plugin	App trait
P11	web	Firefox	AdBlock	ad blocking
P11	web	Firefox	FlashBlock	security
P11	web	Firefox	FoxyProxy	access to content
P11	web	Firefox	Youtube downloader	new features
P11	web	Firefox	Video downloader (for Vimeo)	new features
P11	web	Firefox	Roomy – bookmarks toolbar	UI improvements
P11	web	Firefox	Xmarks – bookmark synchronisation	cross-device computing
P11	web	Firefox	Foxtabs	work organisation
P11	comm	Pidgin	Off-the-Record	privacy
P12	web	Chromium	AdBlock	ad blocking
P12	docu	GIMP	Extra brushes and effects	new features
P12	docu	Blender	Tree modelling plugin	new features
P12	DE	Xfce	Conky	new features
P12	DE	Xfce	Kupfer	improving task efficiency
P13	web	Firefox	AdBlock	ad blocking
P13	web	Firefox	Tool to screenshot Web pages	integrating external features
P13	code	Emacs	Org mode	work organisation
P13	code	Emacs	Python mode	format compatibility
P13	code	Emacs	C mode	format compatibility
P13	code	Emacs	MarkDown mode	format compatibility
P13	comm	Thunderbird	Enigmail	privacy

Table 2. Plugins installed by participants (2/2).

P#	Cat.	App name	Plugin	App trait
P2	code	Vim	A mode to better organise tabs and simplify multitasking	work organisation
P3	web	Chromium	A plugin to enhance the usability of Facebook	enabling social interactions
P4	DE	Xfce	Faster access to recently used apps in app launchers	work organisation
P5	web	Chromium	A password manager plugin that doesn't lock you out when you forget your master password	security
P5	web	Chromium	Support for more types of Web pages in Zotero	format compatibility
P8	DE	Xfce	A way to suspend a task on the phone and resume it on the Desktop	cross-device computing
P9	comm	Skype	A more reliable video calls feature	replacing a buggy feature
P10	web	Chromium	A better way to display and organise browser tabs	work organisation
P10	web	Chromium	Something like Vimperator for Web keyboard navigation	improving task efficiency
P12	DE	File manager	A better way to navigate files with the keyboard	improving task efficiency
P13	comm	mutt	A GPG plugin	privacy

Table 3. Plugins or new features desired by participants.

ID	P#	Cat.	App name	Feature	Threatened by sandboxing?
R1	P1	code	Sublime	Syntax highlighting plugins	partly (some languages provided via plugins)
R2	P2	code	Vim	All plugins	yes
R3	P3	medi	Steam	Game mods in Steam	yes
R4		web	Chromium	Ad blocking plugin	yes
R5	P4	DE	GNOME DE	All extensions	yes
R6	P5	web	Chromium	HTTPS everywhere plugin	yes
R7		web	Chromium	Ad blocking plugin	yes
R8		IM	Thunderbird	Enigmail plugin	yes
R9		docu	LibreOffice	Zotero plugin	yes
R10	P6	docu	LibreOffice	Spell checking plugin	yes
R11		docu	LibreOffice	Compatibility with Office	no (but some plugins provide compatibility for other formats)
R12	P7	code	Eclipse	All plugins	yes
R13		code	Notepad++	Searching through multiple files	yes
R14		*	*generic*	*generic*	ø
R15	P8	web	*generic*	Built-in PDF reader	no
R16	P10	code	IntelliJ IDE	JAR decompression	no
R17		docu	SumatraPDF	Reopening PDFs on previously open page	no
R18	P12	DE	Xfce DE	All plugins	yes
R19	P13	web	Chromium	Bookmarks	no
R20		code	Emacs	All plugins	yes

Table 4. Features for which participants described their reaction to the features' removal.

ID	P#	Cat.	App name	Answer	G	B	Threatened by sandboxing?
J1	P1	code	Sublime	All plugins	✓		yes
J2	P2	code	Vim	All plugins	✓		yes
J3		code	Vim	Any feature or plugin		✓	ø
J4	P4	code	LaTeX	Macros provided by a LaTeX package	✓		no
J5		medi	MPV player	Whole user interface	✓		ø (Desktop sandboxes don't support CLI apps)
J6		web	Firefox	Whole user interface		✓	no
J7	P5	web	Chromium	Ad blocking plugin	✓		yes
J8		docu	LibreOffice	Spell checking	✓		partly (extra languages not packaged in the app)
J9	P6	docu	LibreOffice	Compatibility with Microsoft Office	✓		no
J10		docu	LibreOffice	Grammar checking	✓		partly (extra languages not packaged in the app)
J11	P7	*	*generic*	*generic*	✓	✓	ø
J12		code	Notepad++	Searching through multiple files	✓		yes (unless app granted special privilege to read specific folders or all files of a certain type)
J13	P8	web	*generic*	Built-in PDF reader	✓		no
J14	P9	web	Chromium	Cross-device bookmark synchronisation	✓	✓	no
J15	P10	docu	SumatraPDF	Reopening PDFs on previously open page	✓	✓	no (if sandboxed app is allowed to keep a cache)
J16		code	IntelliJ IDE	JAR decompression	✓		no
J17	P12	DE	Xfce DE	Keyboard shortcuts to launch apps	✓	✓	no
J18	P13	code	Emacs	Syntax highlighting	✓		partly (some languages provided via plugins)
J19		web	Chromium	Bookmarks	✓		no
J20		web	Chromium	All plugins	✓	✓	yes

Table 5. Features for which participants told me about what good (G) or bad (B) justification developers might have for removing them. Yellow rows correspond to primed answers in favour of security being a good justification.

ID	P#	Cat.	App name	Feature	Threatened by sandboxing?
S1	P1	code	Sublime	All plugins	yes
S2		code	Sublime	Syntax highlighting plugins	partly (some languages provided via plugins)
S3	P2	code	Vim	Any feature or plugin	ø
S4	P4	web	Firefox	Vimperator plugin	yes
S5	P5	docu	LibreOffice	Spell checking	partly (extra languages not packaged in the app)
S6	P6	docu	LibreOffice	Compatibility with Microsoft Office	no
S7	P7	*	*generic*	*generic*	ø
S8	P8	web	*generic*	Built-in PDF reader	no
S9	P9	web	Chromium	Cross-device bookmark synchronisation	no
S10	P10	docu	SumatraPDF	Reopening PDFs on previously open page	no (if sandboxed app is allowed to keep a cache)
S11	P12	DE	Xfce DE	Keyboard shortcuts to launch apps	no
S12	P13	web	Chromium	Any feature or plugin	ø

Table 6. Features for which we asked participants what they would think of a feature removal justified by security reasons.