Simulating Perceptions of Security

Paul Wernick[1], Bruce Christianson[1] and Joseph Spring[1]

University of Hertfordshire

Abstract. Systems complicated enough to have ongoing security issues are difficult to understand, and hard to model. The models are hard to understand, even when they are right (another reason they are usually wrong), and too complicated to use to make decisions.
Instead attackers, developers, and users make security decisions based on their *perceptions* of the system, and not on properties that the system actually has. These perceptions differ between communities, causing decisions made by one community to appear irrational to another.
Attempting to predict such irrational behaviour by basing a model of perception on a model of the system is even more complicated than the original modelling problem we can't solve. Ockham's razor says to model the perceptions directly, since these will be simpler than the system itself.

1 Introduction and Rationale

The theme of SPW 2017 is "Multi-objective security". In this position paper we consider security mechanisms in the light of the different perceptions, by different communities, of the degree of vulnerability of a computer system[1] to various attacks. We portray these differences by means of a graphical model, which is intended to capture quantitative changes in these perceptions over time.

We identify three distinct communities involved in the development and use of a system. These communities are likely to have very different opinions on the vulnerabilities of the system. The communities are, first, people planning to attack systems or computer-based devices, or who have already done so (intruders); second, those developing or augmenting the defence mechanisms of those systems (developers); and third, people who lack deep technical knowledge, and simply use the systems (users). A person can belong to one community with respect to one system, and to others for other systems; for example, a person can be an attacker for one system whilst relying for their own security on a tool they are using such as an operating system or language compiler.

Incomplete knowledge means that no community has platonic access to an "objective" reality about the security of the system with which it interacts, and so communities perforce make decisions on the basis of *their* perceptions,[2] along

[1] The term 'system' as employed here includes a computer's operating system, as well as any countermeasures such as antivirus and other programs, and mobile and other inter-connected devices. Our argument applies to all of these.

[2] cf. [1].

with any internal models in which *they* have trust. It is a brute fact that divergences between the beliefs held by the communities cause them to act in ways that other communities perceive as being "irrational". So, rather than trying to ignore these differences as symptoms of a "mistake" on the part of one or another stakeholder, we seek to model them as essentially rational from their community's perspective.

2 Perceptions of Vulnerability

The three communities to which we have referred have very different viewpoints of the security of a system at any particular time:

1. a potential or actual *intruder*, who will attempt to subvert the use or gain control of that device for some reason, and who may be aware of weaknesses unknown to the developers. We include within this community knowledgeable hackers, hacking tool developers, and unsophisticated 'script kiddies'.
2. the *developers* of the software in the device, who may be aware of some weaknesses not yet exploited by intruders but may also be unaware of other vulnerabilities undiscovered by, available to, or already exploited by those intruders.
3. the *users*, whether commercial organisations or individual owners, of these devices, who must rely on the *publicising* of weaknesses to support any decision to update the operating software on their devices.

It is the user community's perception of safety in device or system use which particularly concerns us. Given the lack of reliable information – and the presence of misinformation – available to them, particularly online, they are often unable to make decisions as to how safe their devices or systems are that are as rational as they might make if they were informed better and/or earlier. Consequently, they are vulnerable not only to attacks on their technology but also to a false sense of security in using it [4].

The tensions between the different perceptions of risk implicit in the use of devices with insecure software to access sensitive services such as online banking is heightened when users are unaware of known risks embedded in their equipment. This situation is made even worse when we note that one of the most active areas of growth in mobile banking – Africa – is where it is likely that for cost reasons many people will be using previously-owned devices exported from richer areas of the world.

One impetus for the work described here arises from a paper presented at SPW 23 [8] in which the authors appear to suggest that users of mobile technology are likely to upgrade their devices' operating software at a sufficient speed to ensure that it is sufficiently up-to-date to minimise security risks. We believe this assumption cannot be sustained in an environment in which security holes are not revealed to users; security fixes are not provided to users on an ongoing basis (Android devices are updated at the whim of the manufacturer); and older devices (which may have an increasing number of security issues as support tails

off [2] are not destroyed but passed on to subsequent users: see, for example "Once you've been paid, your phone may be refurbished and sold to insurance channels or sent abroad to be sold on there."[3]

Thomas et al. also note that there are security holes in Android devices still in current use but running older versions of the operating system. A growing number of these vulnerabilities will never be fixed due to the failure of the manufacturers responsible for providing users with updated versions of Android. These permanent weaknesses will inevitably include some of which those users remain unaware, resulting in a false feeling of confidence in the security and confidentiality of their activities. We feel that it is not desirable to *assume* that security is adequately maintained simply by older, insecure devices falling out of use. Nor does it suffice to leave users' security to the whim of commercial organisations with an interest in selling new equipment to people who already have working systems, as suppliers currently seem able to avoid responsibility for loss or damage to users who do not upgrade. A system provider such as an internet bank might seek to pass the risk on to their customer base. In the past card issuers have denied liability for card fraud by stating as fact that the victim must have told somebody their PIN, and tried to fight off their liability.

3 Deciding When to Update

The decisions of users when (or whether) to update are likely to be influenced by their *perceptions* of the *risks* to continued operation incurred by performing these updates, as well as of the benefits of doing so. The behaviour of these users can be influenced by statements by interested parties, such as antivirus authors whose concerns are often quoted in the media, and news media who rely on a stream of potentially-overstated stories of data loss or damage to attract people to their websites and increase advertising revenue.

An example of an issue which we are exploring is whether users should install updates published by software suppliers as they are issued, a common concern for users of technology. Users may be impelled to update their systems if they have been attacked themselves.[4] Under these circumstances they might adopt all future upgrades. They might buy new versions of software as they believe they need to – or as they are told they need to. Alternatively, if they are not currently aware that they have been attacked or are at risk, they might not update their systems, on the basis that they don't think they need to, or have been scared away from doing so, perhaps by potential attackers telling them not to upgrade because the upgrade will damage their systems.

If a user does decide to update their system this may not be a painless process for the non-technical user, and may also incur a subsequent risk to the continued operation of the device, even with modern updating procedures (e.g. [5]; [7]). So if the updating procedure is under the user's control then it must be

[3] http://www.comparemymobile.com/, accessed 11 October 2016.

[4] Many commercial organisations test updates received from suppliers before distributing them to their user base, a protection unavailable to the private user.

expected that a proportion of users will choose to defer updating, preferring to let others find the landmines, until it is imperative or even beyond. This may be one reason for Microsoft updating Windows 10 without user permission or control; there may also be commercial reasons for this policy relating to their move to a subscription model of revenue generation.

In future work we intend to modify the model to reflect the real-world circumstances of one or more specific systems and the media stories surrounding their security history, and calibrate the model to explore the dynamics of the three communities' perceptions of how secure those systems have been over time.

4 The Simulation Environment

Our model has been developed using the Vensim [9] system dynamics [6] (SD) environment. In this approach to simulation, which is based on an analogy with a hydraulic system, movements of tangible elements, and intangible aspects such as belief, are seen as equivalent to liquids moving from one state (represented by a rectangular box and referred to in SD as a 'level') to another via pipes (double lines) whose flow is constrained by valves ('flows'). Levels and flows are named variables in the quantified simulation; the equations which calculate their values are based in part on their type, the value of a level for example being the integration over time of its inputs minus its outputs, typically commencing with a given initial value at the start of the simulation run. The value of a variable is computed based on the values of other model variables, and the constant numbers and look-up tables which link the model variables to real-world values.

The benefit of this simulation approach is not limited to what can be found from running a quantified model; as Professor Lehman told the first author of this paper, "90% of the value of a simulation comes from building the model".[5] This is because the modeller is forced to eliminate ambiguities and resolve misunderstandings in their appreciation of the situation under consideration, and the resulting model structure diagram becomes a vehicle for discussion between the model builder and the different groups of stakeholders ehicvh can improve understanding of the situation being analysed.

5 The Model Described

Our intention in developing this model is to enable a comparison to be made between the relative degrees of confidence (whether well-justified or otherwise) of three groups of people who are stakeholders in the security of a software-based system. These are: people who are attacking the security of that system, or wish to be in a position to do so in the future; the people responsible for developing, maintaining and upgrading that system; and the end users of that system. Our current model reflects a generic situation; discussions at SPW have convinced

[5] Manny Lehman, conversation with Paul Wernick, 1997.

us that it needs to be refined to reflect the events and influences in the history of a specific system before it can be usefully calibrated.

We have included in our model different causes of perceptions of vulnerability for the communities which we have identified. This requires us to simulate the ability of an attacker to exploit a vulnerability of the software running on a system, the ability of the developers of its software to fix that vulnerability, and the ability and motivation of a user of that system to access the necessary security fixes. Our approach to modelling has been to ignore detailed differences between specific vulnerabilities or exploits, and attempt to improve our understanding of the situation from a high, more general, level; this approach is typical when using a continuous simulation environment such as system dynamics. Our current uncalibrated model shows how the differences in perception can arise; the final, quantified, model will also be able to quantify the degree to which these perceptions diverge over time.

Once this configuration and calibration of the model are complete, it will be possible to trace changes in these levels of belief over time. We expect this to show that attackers will be confident in their ability to attack such a system using techniques such as zero day exploits, that the developers will have some confidence that the current version is resilient, and that the confidence of end users – those most vulnerable to loss – in the security of the system may be considerably greater than that of the other two groups.

The current model consists of three main parts:

1. The flow of vulnerabilities from implicit through discovery by potential or actual intruders, the identification and fixing of these issues by software developers, and the releasing of these fixes to the field. This is modelled as an SD *aging chain*.
2. The developers' willingness to devote effort to fixing security holes, whether these holes have been identified in the field and fixed for release or identified within the development organisation and prepared for release but not fielded until the issue is subsequently exploited in the field.
3. The process which cause users of these devices to realise that their systems are vulnerable, identify relevant fixes made available to them, and adopt them. An important concern here is whether developers provide users with fixes to security weaknesses exploited in their model of device.

In some cases security and other updates are provided and installed automatically without user intervention, whilst at the other end of the scale there are many devices still in use which will never receive any further fixes to security weaknesses. These weaknesses, whether exploited on a specific device or not, will remain until the device is taken out of use. The users of these vulnerable devices may be aware of the problem, for instance people still using Windows XP for which discontinuance of support has been well publicised, or they may be unaware of the security holes in their devices, which we believe is the case for owners of mobile phones running older versions of Android. This issue is also likely to be relevant to users of many other software-based systems, including the security failures of the Internet of Things (cf. [3]).

At the same time it is likely that users will install updates for reasons other than security (for example additional or improved functionality, fixes to non-security-related bugs), and these updates may contain security fixes which improve user protection without users perceiving it. Of course, those fixes may also include new or revived vulnerabilities.

The structure of the model is shown in Figure 1.

6 Calibrating the Model

Following the identification of a specific system for consideration and the modification of the model structure to reflect that system, our intention is to calibrate our model with equations and values so as to produce quantitative results showing any divergent trends in perceptions of the security of the system. This will allow us to quantitatively assess the differences in perception of security of the three different communities and, we hope, explain trends in, for example, changes in the rates of successful attacks on examples of that system and the take-up of new more secure versions as they appear.

We have made the following tentative decisions on how to approach this calibration:

- The time step is in terms of elapsed time, to allow the significance of delays in, for instance, releasing fixes to known issues, to be considered
- Successive releases of the software under consideration are assumed to be a single sequence of releases. To avoid having to contend with the actual numbers given to releases and sub-releases, the sequence is indicated by a single integer value which increments with each release; this is the Release Sequence Number (RSN).

We will rely on expert opinion for input values which cannot be easily measured directly, such as numbers of undiscovered vulnerabilities; we have used this approach previously in earlier modelling activities [10].

7 Initial Results

Even before we adapt the model to a specific system and calibrate it to produce quantified results, we can derive some initial insights. Some of these are well-known, but it is reassuring that the model supports them, a situation which gives us more confidence in the utility at a high level, of the model structure as a representation of the current security situation. We expect that as we adapt the model structure it will be possible to draw further conclusions of this type.

An advantage of the Vensim environment is that a built-in analysis tool allows the modeller to see easily where feedback loops exist between variables; these loops will have a strong influence on the behaviour of the overall model and the perceptions which it seeks to represent. Even without a quantified model, we believe representing the situation in a graphical notation helps to clarify

Fig. 1. Model of Degree of Confidence in System Security

the relationships between different stakeholder groups. Examples of this include showing how users of a system can be left behind the current state of security mechanisms without realising until it is too late, and the sources of pressures on software developers to fix security flaws.

As an example of the stories that the uncalibrated model structure can tell, here is one feedback loop in the model; this loop involves 11 variables:

New security fix releases
- Latest RSN
- New vulnerabilities
- Undetected Vulnerabilities
- Hacker-identified vulnerabilities
- Hacker-detected Vulnerabilities
- Attack rate
- Exploited Vulnerabilities
- Publication rate of attacks
- Publicised Unfixed Vulnerabilities
- Developer spend on publicised security fixes per unit time
- Average delay to find fixes

New security fix releases

This loop tells a story of newly-released security fixes themselves generating new vulnerabilities. After deployment of the new release, these vulnerabilities can be identified and exploited by hackers. The exploits are then either privately reported to the developers or publicised in the media, which causes the system's developers to deploy effort to close the security holes, an activity which inevitably takes time. The fixes are then released, causing a new set of vulnerabilities to arise or old vulnerabilities to re-emerge in new forms, in addition to existing problems, which are then identified by hackers, and so on.

It is unlikely that users, system developers and hackers will share the same appreciation of the current level of security in their systems, and of how vulnerable they are to attack. Some users' systems are no longer being evolved to fix security vulnerabilities; some users do not receive updates to security holes in a timely manner; and some will choose, if they have the option, to not update their systems. These groups of users are likely to be more vulnerable than they believe to be the case. Contrary to one possible inference from Thomas et al.'s [8] work, our model demonstrates how the failure of systems developers to maintain the security of their customers' systems can result in users being made more vulnerable to attack *without their realising that this is the case*. This is particularly true for users of systems which are no longer being supported with security fixes, and who are unaware of the seriousness of the threat, or who are otherwise persuaded not to replace their systems, or cannot afford to do so, or are buying old devices with operating software containing security holes. Even if they are aware of the risk, the situation is entirely outside the users' control unless they spend the money (if they have it), and take the time, to replace their systems.

8 Future Work

We intend to develop this work by identifying a specific long-lived software system with many security fixes issued over time. Following any necessary modifications to the model structure to capture this history we will complete its quantitative calibration. This will reflect trends in the rates of attacks and of the development, fielding and adoption of fixes. Further outputs will include estimates of the degree of confidence each group holds in the completeness and/or accuracy of their knowledge. We then hope to involve industrial collaborators such as antivirus software developers in helping to improve our understanding of how and why each of the groups we have identified acts as they do on the basis of their differing perceptions of the security of the software they use.

References

1. Christianson B (2013) Living in an Impossible World : Real-izing the Consequences of Intransitive Trust; Philosophy and Technology; 26 (4); pp. 411–429
2. Clark S, Blaze M and Smith J (2014) Blood in the Water: Are there Honeymoon Effects Outside Software?; Security Protocols 18; LNCS 7061; pp. 12–24
3. Leverett E, Clayton R and Anderson R (2017) Standardisation and Certification of the 'Internet of Things'; Proc WEIS 2017; to appear
4. Murayama Y et al. (2011) The Sense of Security and a Countermeasure for the False Sense, Security Protocols 19; LNCS 7114; pp. 205–222
5. Robinson D (2016) Windows 10 backlash: Which? demands compo for forced upgrades; The Register; www.theregister.co.uk/2016/09/22/windows_10_backlash_begins_which_calls_for_upgrade_compensation/; accessed 21 November 2016
6. System Dynamics Society (2016) What is System Dynamics?; http://www.systemdynamics.org/what-is-s/; accessed 7 December 2016
7. Tepper F (2016) Updating to iOS 10 is bricking some iPhones and iPads; TechCrunch; https://techcrunch.com/2016/09/13/updating-to-ios-10-is-bricking-some-iphones-and-ipads/; accessed 11 October 2016
8. Thomas DR, Beresford AR, Coudray T, Sutcliffe T and Taylor A (2015) The Lifetime of Android API vulnerabilities: case study on the JavaScript-to-Java interface ; Security Protocols 23; LNCS 9379; pp.126–144
9. Ventana Systems, Inc. (2016) https://vensim.com/; accessed 7 December 2016
10. Wernick P and Lehman MM (1999) Software Process Dynamic Modelling for FEAST/1; Journal of Systems and Software; 46; 193–201