Systematic Decision Making in Security Management Modelling Password Usage and Support

Simon Arnell, Adam Beautement, Philip Inglesant, Brian Monahan, David Pym, Angela Sasse

HP Laboratories

HPL-2011-36

Keyword(s):

security analytics, security management, economics, password

Abstract:

We demonstrate the use of a systematic decision-making methodology to support an informed choice of a password policy. Our approach uses an executable system model, grounded with empirical data, to compare, using simulations, two options. The basis of the comparison is a notion of organizational utility. Using our results, we are able to explore trade-offs between breaches of system security, users' productivity, and investment in support operations.

External Posting Date: March 21, 2011 [Fulltext] Approved for External Publication
Internal Posting Date: March 21, 2011 [Fulltext]

SYSTEMATIC DECISION MAKING IN SECURITY MANAGEMENT: MODELLING PASSWORD USAGE AND SUPPORT

SIMON ARNELL, ADAM BEAUTEMENT, PHILIP INGLESANT, BRIAN MONAHAN, DAVID PYM, AND ANGELA SASSE

ABSTRACT. We demonstrate the use of a systematic decision-making methodology to support an informed choice of a password policy. Our approach uses an executable system model, grounded with empirical data, to compare, using simulations, two options. The basis of the comparison is a notion of organizational utility. Using our results, we are able to explore trade-offs between breaches of system security, users' productivity, and investment in support operations.

This is an interim report of research originally conducted during February 2010; it is anticipated that a fuller, expanded version will appear in print in due course.

1. INTRODUCTION

Security managers in organizations are routinely called upon to make choices about the technologies and policies they deploy in order to protect the business-critical infrastructure of the organization. These choices are constrained by both economic and regulatory circumstances and managers necessarily must make trade-offs. When faced with analysing these trade-offs security managers have very limited tools to aid them in systematic decision making (for example, [7]). Consequently managers rely on their own experience and intuition when selecting their implementation choices. While it is not being suggested that this results in poor decision making there are inherent drawbacks with this approach, including:

(1) Decisions made in this fashion may be optimal but cannot be shown to be optimal. Their relative merit will also be opaque to the manager who will have no rigorous means of comparison with other options;
(2) Any decisions taken by the manager cannot be shared in a meaningful with other stakeholders, in particular business and finance managers.

A recent article [22] indicates that the rise of cloud computing could lead to passwords comprised of fewer than 12 characters being insecure against a brute force attack. We envisage a scenario in which a senior business manager — concerned about the implications of this news — asks the organization's security manager to change the organizations security policy to mandate a minimum password length of 12 characters. Any experienced security manager would be wary of the impact of such a change on the organization's workforce and infrastructure, but how should the manager test the validity of this intuition and communicate the result to the business manager in a manner that he can appreciate? The language of finance speaks to all sections of an organization whatever their primary remit, and so we approach this challenge from an economic viewpoint.

Building on our previous work in this area [3, 18, 20], we use the economic concept of utility, capturing the managers' objectives and preferences, to motivate the creation of an executable mathematical systems model representing the key processes, locations, and resources of the organization as well as the environment within which they sit. Critically, we keep the model grounded by supporting its construction and parametrization with empirical data gathered for the purpose. The combination of these approaches is at the core of our proposed methodology for systematic decision making. Using this model, we can compare the expected utility of the organization under two password policy conditions; the first mandating a minimum of 6 characters and the second a minimum of 12 characters.

The remainder of the paper is organized as follows:

- In Section 2, we outline the economic concepts behind the development of the utility function;
- In Section 3, the design and implementation of the empirical data gathering process is discussed;
- Section 4 contains an explanation of our systems modelling approach;

Date: 22nd February, 2010.

- Section 5 provides a conceptual explanation of the executable mathematical model used in this scenario;
- In Section 6, we analyze the results and draw conclusions.

2. Utility (Loss) and Information Security

The fundamental concepts of information security are confidentiality, integrity, and availability. These concepts are *declarative*; that is, they express properties[1] that systems may possess. Alongside these declarative, organizing concepts, sit various *operational* notions — such as access control, authentication, and possession of keys, and so on — that are used in order to establish the declarative properties.

Over the years, many authors (for an excellent distillation of the essentials, see [27]) have suggested that confidentiality, integrity, and availability provide an incomplete basis for information security analyses, and have suggested that various concepts — as above, access control, authentication, and possession of keys, and so on — be added to them. In doing so, they confuse the declarative and the operational.

This confusion becomes problematic in the context of investment decision-making, where the concept of utility — also sometimes confused with the declarative and operational concepts mentioned above — is critical. Utility theory (see, for example, [21, 32]), particularly as developed in the contexts of macroeconomics and financial economics, provides a highly expressive framework for representing the preferences of the managers of a system.

For example (e.g., [30]), in the macroeconomic management of market economies, central banks play a key rôle. The managers of a central bank are given, by their national governments, targets for certain key economic indicators, such as unemployment (u_t) and inflation (π_t) at time t (time can be either discrete or continuous here). Their task is to set a (e.g., monthly) sequence of controls, such as their base (interest) rates (i_t) so that the key indicators are sufficiently close to their targets, $\overline{u_t}$ and $\overline{\pi_t}$, respectively. Typically, using this example, the managers' policy is expressed as a utility function

$$U_t \quad = \quad w_1 f_1(u_t - \overline{u_t}) + w_2 f_2(\pi_t - \overline{\pi_t}) \tag{1}$$

together with system equations, $u_t = s_1(i_t)$ and $\pi_t = s_2(i_t)$, expressing the dependency (among other things) of u and π on interest rates in terms of functions s_1 and s_2 that describe the (macro) dynamics of the economy. Two key components of this set-up are the following:

- The weights w_1 and w_2 (typically, values between 0 and 1) that express the managers' preference between the components of the utility function — that is, which they care about more; and
- The functions f_1 and f_2 that express how utility depends on deviation from target. A simple version of this set-up would take the f_is to be quadratic. Quadratics conveniently express diminishing marginal returns as the indicators approach target, but make utility symmetric around target. More realistically, Linex functions [33, 34, 30], usually expressed in the form $g(z) = (exp(\alpha z) - \alpha z - 1)/\alpha^2$ are used to capture a degree of asymmetry that is parametrized by α.

The managers' task, then, is to set a sequence of interest rates i_t such that the *expected* utility, $E[U_t]$, remains with an acceptable range, as u_t and π_t vary, and trade-off against each other, as the sequence of rates i_t evolves. In general, there can of course be as many components as required in a utility function.

This economic framework can be deployed in the context of information security (see, for example, [3, 20, 19, 5]), where concepts — such as confidentiality, integrity, and availability — that lie within competing declarative categories can be seen to trade-off against one another as the relevant controls — such as system configurations or investments in people, process, and technology system configurations — vary.

In this paper — in which we are concerned with the use of passwords to access a system and the issues associated with the need to reset them — the declarative information security concepts of interest are the following:

- *Breaches*, B, which may be understood as a particular aspect of confidentiality: passwords become known to unauthorized individuals;
- *Productivity*, P, which may be understood as a particular aspect of availability: the user's ability to access the system and perform work-tasks; and
- *Investment*, K, which here is simplified to be the provision of IT Help Desk effort.

[1] Logically, this is made precise in *property theory* [2].

We thus end up with a utility (loss) function of the form

$$U(k,l) \quad = \quad w_1 f_1(B-\overline{B}) + w_2 f_2(P-\overline{P}) + w_3 f_3(K-\overline{K}) \tag{2}$$

where the parameters (αs) of the Linex functions f_i and the weights w_i are determined by the managers' preferences, and the control variables,

- k, the level of investment in help-desk staff.
- l, the length of password (here we explore six and twelve characters).

are the parameters that are explored experimentally using an executable mathematical model of the system (as explained in Section 5).

Thus the executable mathematical system model replaces the rôle performed by the system functions, s_1 and s_2, in the discussion above.

We can summarize the approach as follows:

- The organization that deploys information security measures exists in an economic and/or regulatory environment. This environment places constraints upon the systems and security architectures available to the organization's managers;
- The managers formulate a utility function that expresses their policy preferences, which will depend upon the nature of their organization. For example, state intelligence agencies and online retailers will have quite different priorities among confidentiality, integrity, and availability; see, for example, [20];
- In a highly complex situation, such as a security architecture, it will typically not be possible to formulate system equations (in terms of functions s_1 and s_2) in the way that is usually possible in, for example, macroeconomic modelling. Typically, though, the key control variables, such as system interconnectivity or investment in various aspects (people, process, and technology) of security operations, will be identifiable;
- Instead, however, an executable system model [11], using the key control variables, can used in order to simulate the the dynamics of the utility function.

Finally, note that for the purposes of our results, as presented in Section 6, we work *not* with utility, which one seeks to maximize, but rather with its dual, *loss*, which one seeks to *minimize*.

3. EMPIRICAL DATA GATHERING

3.1. **Populating the model from empirical password studies.** To obtain an empirical basis for our model for the case study described in Section 1, we researched the impact of user authentication through passwords on organizational productivity. We chose password authentication because it is the most widely deployed security mechanism in commercial and public sector systems today. Previous research on password practices has established that users prefer simple passwords that are not very secure (e.g., [13]). Most organizations try to counteract this preference through security policies and mechanisms that force users to make passwords more secure — by manating a minimum length and complexity of passwords, and regular password expiry. However, most users cannot cope with the cognitive demands that results from those policies, leading to an increase in:

(1) The number of forgotten passwords [31]. Secure resetting of passwords consumes resources in the organization — helpdesk time and user time;
(2) The number of users who employ workarounds, such as writing passwords down [1]. These workarounds often create new vulnerabilities, and these vulnerabilities increase the likelihood of an organizational security breach.

Previous research did not, however, provide a set of data that would allow us to determine the impact of different password policies on organizational productivity in a systematic way. For instance, we were not able to predict by what proportion helpdesk calls would increase if an organization forces users to change from a simple 6 character password to a complex 12-character one. Similarly, determining the risks resulting from workarounds depend very much on the type of users, and the context in which they and the organization operate. The security implications of a password being captured depend on what the user has access to — some users have access to sensitive systems and data, others do no. The risk of a password being captured is very different for users who only access systems from the relatively controlled environment inside an organization's perimeter, compared to 'road warriors' who spend much of their time accessing systems while in public areas, and using 3rd-party wireless access networks.

To create an empirical basis for our model, we conducted a set of studies to obtain data on the impact of different password policies:

(1) We estimate the time to generate a password of specific length and complexity, based on observations of a group of participants in a laboratory study;
(2) We estimate the time to enter a password of a specific length and complexity, and the number of failed attempts, based on a set performance test under laboratory conditions;
(3) We estimate the frequency of forgetting a password a password, based on a password diary study [18] conducted with employees in two organizations, supplemented by reset statistics from two organizations.

3.1.1. *Study 1: Time to generate passwords.* We measured the time it took the 19 participants (those in cohort G1 in Table 1 below) to generate passwords in response to a specific password policy: the password had to be 7 or 8 characters long, contain 3 out of 4 character types (lower case letter, upper case letters, numbers and symbols), and could not consist of a dictionary words or simple derivatives. Participants took on average 1 min 34 seconds, (standard deviation 86.4) to produce a password that complied with the policy.

3.1.2. *Study 2: Time to type in passwords.* We carried out a set of laboratory-based experiments, in which we measured the time it took participants to enter passwords of varying length and complexity (see Table 1). The passwords tested comprised passwords that were generated by participants in response to a specific security policy, and passwords that corresponded to the requirements of the policy, but were generated for them.

TABLE 1. Password Allocations

Cohort	**No. of Participants**	**Password given or chosen by participants**	**Policies used by the cohort**		**Mean time per attempt**	**Success rate on first attempt**
C1	22	Passwords chosen	Policy 1	Min 6 chars, 3 of 4 character sets	3.74s	219/220 = 99.55%
			Policy 2	Min 8 chars, at least 1 numeric, 1 capital	4.62s	215/220 = 97.72%
G1	19	Passwords given	Policy 1	7 letters, numbers, and non-alphanumerics	3.94s	185/190 = 97.37%
			Policy 2	8 letters and numbers	3.45s	186/190 = 97.89%
C2	21, of whom 2 failed	Passwords chosen	Policy 1	Min 12 letters, numbers and non-alphanumerics	6.16s	178/190 = 93.68%
			Policy 2	Min 10 letters, numbers and non-alphanumerics	5.40s	182/190 = 95.79%

Participants entered passwords corresponding to two different policies 10 times each. In total, we measured 1200 password entries by 62 participants; 2 of these participants were unable to complete the experiment because they had completely forgotten the password, and 3 others required password reminders. The mean time overall for a single password attempt was 4.53 seconds. In 1165 of these entries, the password was successful on the first attempt.

3.1.3. *Study 3: Frequency of forgetting passwords.* We captured the likelihood of forgetting passwords through diary studies with 32 participants in two organizations — a university and an investment bank — who kept a diary of their password interactions for one week. In total, 144 unique passwords were recorded in the study, used in 982 password entries, of which users reported 5 forgotten passwords, resulting in the need for a new password or helpdesk call. We also obtained statistics for password resets as recorded by helpdesks (the more secure but expensive option) and self-service resets through portals (cheaper, arguably less secure) in these organizations. These are shown in Table 2; in comparison, Florencio and Herley [13] report that only 1.5% monthly of Yahoo users' passwords have to be reset every month: unlike the organizations in our study, Yahoo does not mandate strong passwords, and most users had weak passwords. The password diary study also revealed that, in addition to time and effort required to carry out the reset, there are further implications for productivity because of the time for it can take for a password reset to propagate through the organization's infrastructure — in one of our organizations, it took on average 2 hours. This means users were 'locked out' from many services for that period of time, and had to re-arrange their primary tasks.

TABLE 2. Actual password resets in two organizations

Organization	**Type of password**	**Period of data**	**Mean password resets per headcount per month**
Investment bank	Single sign-on	January - October 2008	11.4%
University	Single password but not full SSO	December 2009 - January 2010	13.4%

3.2. **From empirical data to the model.** The model (see Section 5) makes the simplifying assumptions of two different password lengths, fitting three categories of character class. The model assumes probability distributions for mis-typing, related to the length, character set, and newness of the password, and for time to enter a password from a normal distribution with fixed mean and standard deviation. A user can become locked out on password expiry, by exceeding policy-defined limits, or as a random event. Becoming locked out incurs the time and expense of a password reset, followed by a password change. We model two kinds of reset: a 'light' self-reset through a portal or a 'heavy' reset involving the helpdesk. Whichever reset method occurs, a password reset (for a discussion of costs, see [24]) should be considered to include the non-trivial time and effort of generating a new password and, possibly, the time for the password to propagate around multiple servers. For the purposes of our model, however, these are complexities to be added as later refinements.

Against these costs, the model balances the risks of a partial or complete password capture. The model assumes that these risks parameters are related directly to the location of the user, and for this purpose we modelled users as three categories of user in four locations, making the assumption that the proportion of time in each type of location is related to the user category.

We model a partial capture as observation — through shoulder-surfing, for example — and, for simplicity, we assume that three characters are captured; the number of partial captures required to fully capture the password therefore depends on the password length.

Finally, we validated our assumptions about the three types of users in four locations as part of in-depth interviews with three senior information security managers.

4. SYSTEM MODELLING

Our approach, the mathematical basis of which is presented in [12, 9, 11], is grounded firmly in mathematical logic, computation theory, and probability theory, but employs well-developed, implemented tools. Our approach views a system as having the following key conceptual components:

- Environment: All systems exist within an external environment. We may seek to model the structure of the environment, in which case we treat the environment as a system of interest in itself; typically, however, we treat the environment as a source of events that are incident upon the system of interest according to given probability distributions [6]. For example, a queue at a post office is subject to arrivals of people from the street;
- Location: The components (i.e., resources; see below) of a system of interest are typically, distributed around a collection of places. Different places are connected by (directed) links. Locations may be abstracted and refined provided the connectivity of the links and the placement of resources is respected. Mathematically, various graphical structures capture this notion [9];
- Resource: The notion of resource captures the components of the system that are manipulated by its processes (see below). Resources include things like the components used by a production line, the tools on a production line, computer memory, system operating staff, or system users, as well as money. Mathematically, we capture this notion using certain algebraic structures [26, 29, 28];
- Process: The notion of process captures the (operational) dynamics of the system. Processes manipulate resources in order to deliver the system's intended services. Mathematically, we use algebraic representation of processes based on the ideas in [25], integrated with the notions of resource and location [12, 9].

This framework is explored in detail in [11, 10, 15], with related work in, for example, [16, 17, 14]. The Gnosis modelling toolset, which implements this framework, supporting Monte Carlo-style experiments, is described in [11, 15, 8]. The basic idea is to implement the process component as (concurrent combinations of) sequences of actions, together with a range of control constructs. Resources are declared at locations, and may be shared by processes, which may manipulate the quantity and location of a resource. Events incident upon the system from the environment are represented by the results of the sampling of declared probability distributions by processes. Here the environment is intended to include not only things outwith the system of interest, but also any internal parts of the system that we do not wish to model structurally or in detail.

The experimental methodology deployed in a study of the kind described in this paper must be tailored to the constraints of the experimental environment. The basic starting point, however, is the classic mathematical modelling cycle illustrated in Figure 1 [23], below.

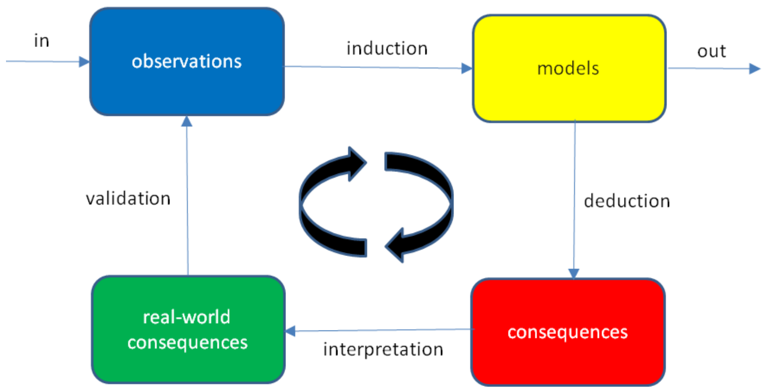


FIGURE 1. The classical (mathematical) modelling cycle

In the context of the study of password usage, as described in this paper, some methodological details are, owing to the relative sparsity of the data, important. In the presence of sufficiently large data sets, classical statistical clustering techniques could be applied. Our approach is to examine the individual executions of the system model and compute the utility function of the resulting clusters of points. Thus we are sampling the utility function rather than analyzing it statistically, witnessing the formation of clusters, and observing local utility maxima. A thorough statistical analysis is further work.

4.1. **From empirical data to the model (revisited).** The password data we gathered is used to provide support underlying our choice of the *transfer function* we use in the model. This function essentially defines a way of mediating from various characteristics (e.g., password length, number of retries) into a success probability — see the initial comments to Section 5 (and also 5.5) explaining the role of the transfer function in our model. While there is a substantial volume of data collected from our password studies, not all the information gathered within the study was in the end relevant to the estimation of the transfer function: for instance, data was gathered for more password lengths than the ones we actually used in the simulation experiments. It is difficult to provide sufficient data to be able to completely determine and thus characterize a functional proxy of this kind.

As an example of how we used the empirical data in constraining the choice of transfer function, consider the following table:

Percentages of passwords accepted on first attempt									
Password Length	6	7	8	9	10	11	12	13	14
Percentages (%)	98.0	97.9	98.1	99.0	98.2	96.7	96.7	100.0	100.0
Probability	49/50	235/240	353/360	99/100	167/170	29/30	116/120	40/40	20/20

Clearly, we can assume that, for 6 and 12 characters at least, most attempts succeeded first time ($>98\%$) – making further password attempts relatively rare events to observe experimentally. Thus, our data constrains the choice of transfer function, without forcing it completely. We conjecture that this issue will be a persistent feature of future studies of this kind.

5. The Conceptual Model

In this section, we explain our executable mathematical model, constructed according to the general methodology outlined in Section 4. The model itself is constructed in the Gnosis modelling language [11, 15], which embodies the structure explained in Section 4. Rather than try to explain the Gnosis code in detail, we instead explain conceptually its various components and their interconnection. The code is deferred to Appendix A.

We begin by imagining our scenario consists of a corporate work situation with a number of types of corporate user: ROAD_WARRIOR, OFFICE_WORKER, and EXECUTIVE, in a variety of different types of location: HOME, WORK, PUBLIC, and IN_TRANSIT.

All of our users need to log-in daily and access information from the corporate intranet system via their desktop/laptop systems (we don't model the infrastructure or their connection state). In making these accesses, they need to 'prove' and test their credentials by typing their password (at least one — there could be more, depending on the number of corporate services users need to access as part of their working day). They typically need to do this authentication several times per day — we only consider those accesses where an authentication challenge is required and we otherwise ignore other tasks where no challenge is necessary.

From the systems modelling perspective, we consider the life-cycle of a user (of a given type) for whom authentication challenges are events that randomly occur from the environment. The key conceptual components of the model are then *authentication*, *password resets*, and the *threat environment*. Note that, critically, all of these components exist at the operational level, and that their dynamics influence the utility (loss) calculation of the levels of the associated declarative components (here breaches, productivity and investment). Investment is represented here by the number of *help desk staff* employed as a part of the model. The utility (loss) function (of the remaining two components) can then be evaluated post-facto from the outcomes produced experimentally by running our model.

Finally, a critical conceptual component of the model is the *transfer function*, explained in detail in (Section 5.5). The key idea is to represent an interaction between the users as represented in the model and the environment. More specifically, the transfer function characterizes the relationship between the number of attempts to enter correctly a password and the probability of successfully entering a password. The use of a transfer function — as initiated in a related previous paper on USB memory stick security [3] — is a convenient way to capture a very complex situation, the exploration of which is beyond the scope of this paper. The form of the function taken (see Figure 2, page 11) is consistent with the empirical data obtained. It is not, however, determined by it. Such a determination would require extensive and delicate experimental work, beyond our present scope.

5.1. **Authentications.** The authentication itself involves typing a user name (we assume perfect recall here - and hence username plays no further role here) and then typing a password of some length. Note that the probability of failure to correctly type the password increases with password length and complexity - via the above transfer function. The following represents the *state* of the user as seen by the authentication system:

UnAuthenticated: User is not currently authenticated: no productive work is being done, but also no productive work is lost;

Authenticated: User has currently authenticated and proceeds to complete the task (this is where productive work could then be performed);

Failed: This is a temporary state in which the user failed to authenticate on this occasion, but may retry again. Retries may resume after a short delay. If the user does not retry after a longer delay, their state returns to **UnAuthenticated** (i.e., this means that the failure state is 'forgetfully forgiving');

Disabled: The user is explicitly prohibited from accessing the system - they will need a complete credentials reset in order to restore their capability to do productive work. However, such a reset is not actively waiting for action - the reset will be initiated at some other time. Importantly, however, the user is taken to be currently unavailable for productive work - and therefore cannot be regarded as enduring a period of productivity loss due to password reset;

For this reason, we legitimately ignore this entirely as it is tantamount to starting over afresh — that is, starting as a new user. We assume a stable population of users: no user joins or leaves;

Expired: Superficially similar to the Disabled state, except that it arises because users failed to comply with the 90 day password renewal policy. Once a password expires, the password is automagically reset - in other words, the user does not have to *initiate* the reset process (in effect, it just happens). At this point the user doesn't actually know their new password and (in effect) there is a further delay while they acquire knowledge of it in some prescribed manner (i.e. from their supervisor, their manager, an admin, their voicemail, or via the internal mail etc.). In this case, the entire elapsed time is an accountable period of productivity loss;

Locked-Out: User is locked out, but pending password reset: this is an accountable period of productivity loss.

Our model has a simulated duration of a single year, with time unit = 1 day. For each user type, we generate authentication attempts randomly at an average rate per day. These then proceed according to the flow described above.

Productive work is assumed to be any task that requires a successful authentication to be made - we ignore all those tasks that don't need authentication in the first place. Productivity loss arises when a user could attempt to perform work but could not for whatever reason. If a user is locked out or is needing a credentials reset, this must be counted as lost productivity (that is, a form of 'lost opportunity cost'). Our proxy statistic for a user's productivity loss is taken to be *the total elapsed time whilst inhibited from working due to password resets*.

This statistic neatly accounts for both the number of resets and usefully encodes how much delay there is in performing a password reset. This is analogous to the concept of computer downtime arising in classic performance analysis.

We could naturally encode various password policies imposed by the organization such as:

(1) Passwords are required to have at least n characters (for some suitable n);
(2) After 5 failed attempts, a password reset is always required;
(3) Every user must have changed their password some time in the last 90 days; equivalently, passwords older than 90 days will expire.

All of the policies above are represented in the current model.

5.2. Because of users respective roles and their locations, they are all subject to different levels of password capture risk. These are represented in the model by a location-indexed point distributions; see Table 3.

For instance, we simply note that ROAD_WARRIOR users are more likely to be working in risky places (i.e., IN_TRANSIT and PUBLIC) and are thus more exposed to potential breaches. We assume the standard fiction that all breaches are perfectly and accurately observable only after the fact and never before. This says that breach detection is essentially perfect, while acknowledging that any breaches that do happen to occur do so only because they are executed extremely well and therefore could not have been anticipated. We do not model the details of the password administration process or its infrastructure as such. All we capture in the model are the *service characteristics*, such as the times taken from initial request to resolution, the number of staff required and so on.

5.3. Resets. For verisimilitude and general credibility, we split the password reset/renewal service into two levels of service, reflecting a common practice in service industries:

Light reset: Essentially a mitigating quick-response, automated, password renewal system, involving zero administration overhead. Although typically a cheaper and easy alternative, it may have some security issues (such as potential for malicious resetting, replays or interception) and hence some unwelcome security drawbacks — that is, it is imperfect. It still takes some time to perform (yielding productivity loss) — but doesn't require expensive resources such as an administrators attention;

Heavy reset: This is a more involved process and requires a password administrator to check credentials and appropriately reset the password. Password requests go into a queue and could potentially take a long time (i.e., more than a day) to reach resolution.

Note that this allows us to neatly capture the issue of escalation: some password resets require escalated responses (heavy reset) while others can be safely resolved automatically via light reset.

5.4. Threat Environment. An organization faces numerous threats, some of which are due to the capture of a user's credentials. Such an adversary may casually shoulder surf a user entering their password or may gain access to the full password through some other means, perhaps placing a user under duress.

A number of abuse cases exist for the leakage of a user's password, from corporate espionage to a jealous partner seeking revenge. A system model however does not need to capture such details of a user's life, we instead abstract away from this to an understanding that, when passwords happen to be captured, they are either fully or partially captured.

We parametrize the global threat environment by allowing the customer to choose a threat level, allowing them to model hypothetical scenarios for changes in their threat environment. The threat level defines how many attacks hit the system.

We extend this idea of a threat environment and consider it not just at a global level but allow the customer to define threat levels at a local level. As the model is located we allow each of the locations a user may authenticate to have probabilities assigned to the type of threats that may exist. Table 3 gives an example definition of a threat environment.

Location	Full	Partial
Home	$1/3$	$2/3$
Work	$2/5$	$3/5$
In transit	$2/5$	$3/5$
Public	$1/2$	$1/2$

TABLE 3. Example probabilities for types of password captures in locations within the model

As previously described, we allow users to self reset using a web portal, the light reset process is quicker i.e. incurs less lost productivity than the heavy process. It has, however, a security trade-off: it is prone to malicious users resetting other users' passwords.

As we offer the decision-maker control over the proportion of light and heavy resets, we can study friction points that exist in the trade-offs between productivity loss, breaches and investment required in the help desk to fulfil resets.

5.5. Transfer function. We abstract a user's authentication process into a transfer function that takes the following parameters: password age, policy defined required length, policy defined required complexity,policy defined maximum number of attempts, number of attempts, policy defined maximum time allowed to authenticate and time taken to enter. The function provides us with a probability that a user was successfully logged into the particular system.

We assume the function is the product of a number of dependencies that take parameters as defined below:

$$P(X) = \prod_{i=1}^{5} x_i \tag{3}$$

These dependencies are based on modelling intuition and empirical data. The empirical data collected allows modellers to attune their intuitions on the morphology of the transfer function. If a purely data-driven transfer function were desired then one must collect a much larger data-set to improve confidence levels, thus our current dataset provides only an indication.

We now discuss the assumptions that have been made to model each of the dependencies.

Password length: Passwords under 6 characters are very easy to recall, we therefore assume passwords of this length do not reduce the users chances to successfully authenticate. Therefore the probability of successfully authentication is inversely proportional relationship to the length of a password, i.e. as $length \to \infty$, $P(success) \to 0$.

$$x_1 = \begin{cases} 1 & \text{if } length \leq 5 \\ 1 - \frac{length-6}{5000} & \text{if } length > 5 \end{cases} \tag{4}$$

Number of tries: The number of tries affects the probability only in so far as if the maximum number of tries permitted by policy is met we do not allow the user to authenticate, otherwise it has no effect.

$$x_2 = \begin{cases} 1 & \text{if } tries \leq maxTries \\ 0 & \text{if } tries > maxTries \end{cases} \tag{5}$$

Password complexity: All password complexities have a $P(success) < 1$, simple, moderate and complex passwords are increasingly harder to recall and/or correctly type. We form this assumption based on simple passwords could be all lower case characters [a-z], moderate lower case, upper case, and numbers [a-zA-Z0-9], we could therefore assume the use of modifier keys such as caps lock and shift reduce the probability of a successful authentication.

$$x_3 = \begin{cases} 996/1000 & \text{if } complexity = complex \\ 998/1000 & \text{if } complexity = moderate \\ 999/1000 & \text{if } complexity = simple \end{cases} \tag{6}$$

Time taken to authenticate: Using the empirical data we assume that correctly typed passwords are typed within a window, those that exceed this policy defined window are not allowed to authenticate, otherwise it has no effect.

$$x_4 = \begin{cases} 1 & \text{if } timing \leq maxTime \\ 0 & \text{if } timing > maxTime \end{cases} \tag{7}$$

Password age: We assume a password requires a training period before the probability of a successful recall increases (the point at which a user has 'learnt' their password), after this point as $age \to \infty$, $P(success) \to 1$. Thus, the probability of a successful authentication is directly proportional to the password's age.

$$x_5 = \begin{cases} 0.999 & \text{if } age \leq 2; \\ 1 - \frac{1}{1000(age-1)} & \text{if } age > 2 \end{cases} \tag{8}$$

For the purposes of illustration, we fix a number of parameters for the transfer function and plot in Figure 2 the resultant probability (y-axis) with respect to password length (x-axis) and password age (z-axis). It is worth noting the very small difference in probability between 0 days (≈ 0.995) and 90 days (≈ 0.998), in terms of a model with 100 users who authenticate with a mean of every 20 minutes, results in roughly an order of magnitude difference in the number of daily password failures.

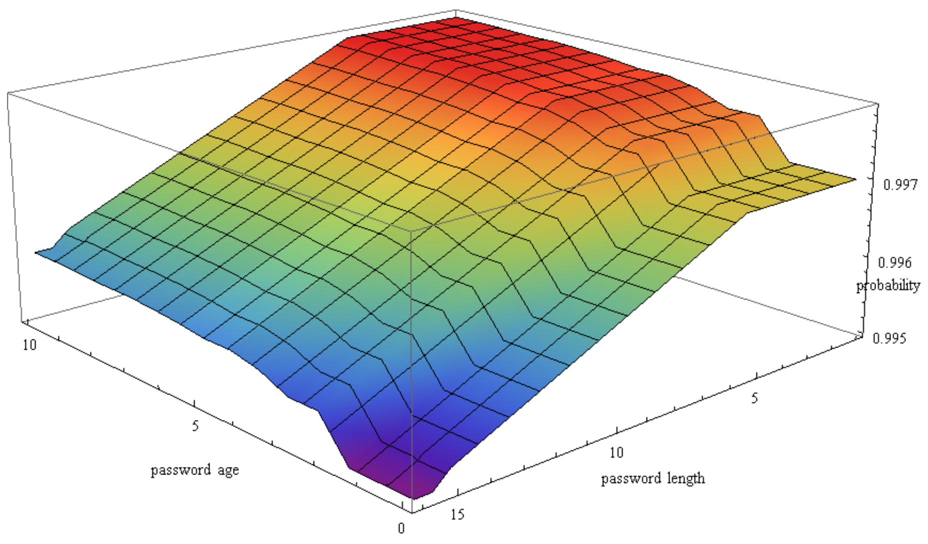


FIGURE 2. Plot of successful authentication transfer function where $maxTries = 5$, $tries = 1$, $maxTime = 25$, $time = 1$, $complexity = moderate$

6. RESULTS, ANALYSIS, AND CONCLUSIONS

In § 2, we considered the example of a security manager needing to determine the impact on utility (loss) — as represented by the cost of breaches, productivity loss, and investment in support services — of switching to a password policy mandating a minimum of 12 characters from one requiring only 6. To investigate this decision, we executed the model described in Section 5 (code in Appendix A) using the combinations of parameters in Table 4. This clearly results in 6 combinations in total.

Parameters	Values
Password Length	6; 12
Number of Help Desk Staff	1; 2; 3

TABLE 4. Experimental parameter combinations

We ran each of these experimental combinations 48 times, for a simulated duration of 1 year and with an independently randomised random-number generator seed to ensure novelty of choices (e.g. Monte-Carlo simulation). The results obtained by performing these runs yields the following data:

- Number of breaches seen;
- Productivity loss due to password resets for OFFICE_WORKER, ROAD_WARRIOR and EXECUTIVE.

With the results generated we can make use of the utility (loss) function as described in Section 2. As the utility (loss) function is not required in the simulation runs itself, the manager can investigate the effects of different trade-offs and target preferences using the same underlying data.

For the purposes of this paper we have taken the following targets and weightings in Table 5 where the

	Targets	Loss Function Weighting	Linex (α)
Breaches	1.25	3	0.4
Performance Loss	0.65	6	3

TABLE 5. Preferences for Targets and Loss Function Weightings

perfomance loss is measured in terms of the total salary paid out while staff are unable to work effectively due to password reset.

Both values are scaled so that a value of 1 means the same as 1 basic yearly salary. The above targets mean that annual costs due to performance loss should be at most 65% of a basic annual salary and, likewise, annual costs due to breaches should be at most 125% of a basic annual salary.

The loss function weightings we took are a value of 1 to account for help desk costs (e.g., salary), 3 for breaches and 6 for performance loss. This choice makes performance loss twice as important as breaches, expressing a preference for minimising performance loss over breaches. For example, such a preference might represent a corporate bias towards penalising anything that immediately causes a decrease in staff productivity. This preference can be adjusted by changing the loss function weighting and thus it forms one of the parameters through which a manager can adjust the model to represent better his organization.

The Linex values are both positive (to penalize any overshooting of loss targets) and their different relative strengths further emphasises the security manager's preference to penalise performance loss more heavily than breaches.

6.1. Displaying Outcomes from Simulation Experiments. In this section, we now turn to our visual display of outcomes from our simulation experiments, using the types of charts below. Each of these chart types are given for each of the 6 experimental parameter combinations that we took.

Scatter Plots: This is a 2D plot of a set of points (in a distinctive colour):

- **Cost of Breaches:** This is a value lying within the range $(0, 2.0)$ and represents the total annual breach costs to the organization;
- **Cost of Performance Loss:** This is a value lying within the range $(0, 4.0)$, and represents the total annual costs of performance loss (due to password resets) for the organization.

Each 2D point therefore represents values of the number of breaches and the total performance loss, computed as a weighted sum of the performance losses as seen by each type of user: OFFICE_STAFF, ROAD_WARRIOR and EXECUTIVE, weighted to their relative salaries).

Naturally, all of these plots share common axes, to both illustrate, and facilitate comparison of, point clustering; the green box indicates the rectangular region that the associated Contour chart is given.

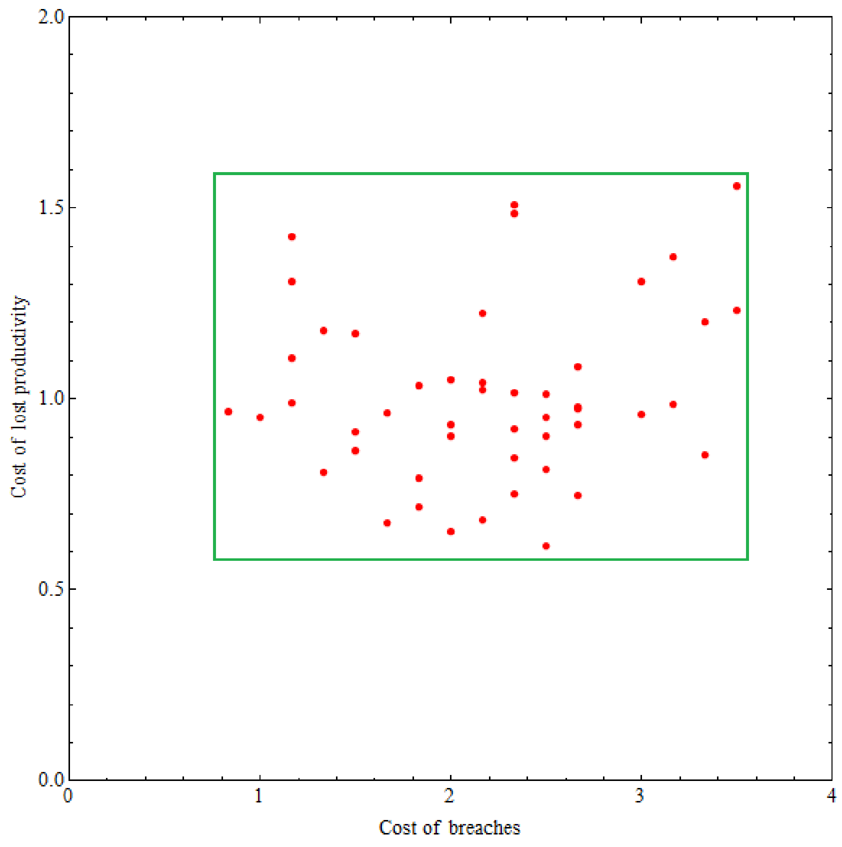
Additionally, we also give an aggregated Scatter Chart for all of the points (see Figure 9)

Scaled Contour Plots of Loss Values: This chart gives a contour plot of the loss function value for the points lying in the region indicated in the green box given in the *associated* Scatter Plot. To show the loss phenomena, a further scaling has been applied to both the XY axes and also to the value contours themselves. Such a scaling is necessary to show off both the different XY regions and the (potentially substantial) variation of the loss (calculated from the utility function). A legend is included stating the contour scaling used for each chart. Because of this scaling, care is needed to compare outcomes between contour charts.

Scaled 3D Stick Diagrams of Utility Values: Utility Values mentioned above. Each 'stick' represents the particular utility (loss) value (encoded by scaled lengths) for a particular point. Note that the XY axes are the same as those used in the associated contour plots. These are included to show an alternative method of representing the data but as they display the same information as the contour plots are not explicitly discussed in the analysis section.

These plots are grouped together by password length, as in Figure **??**, and Figure 11

The result charts for each experimental combination are shown below, grouped according to password length and help desk staff numbers. Figures 3, 4 and 5 give the results for 6 character passwords, and Figures 6, 7 and 8 give corresponding results for 12 character passwords.



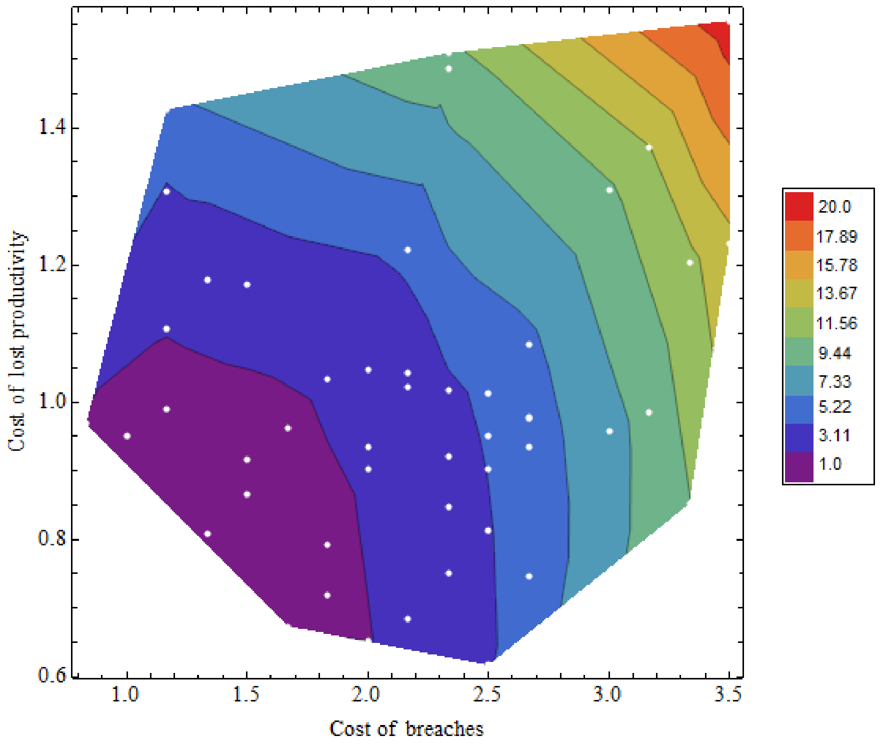
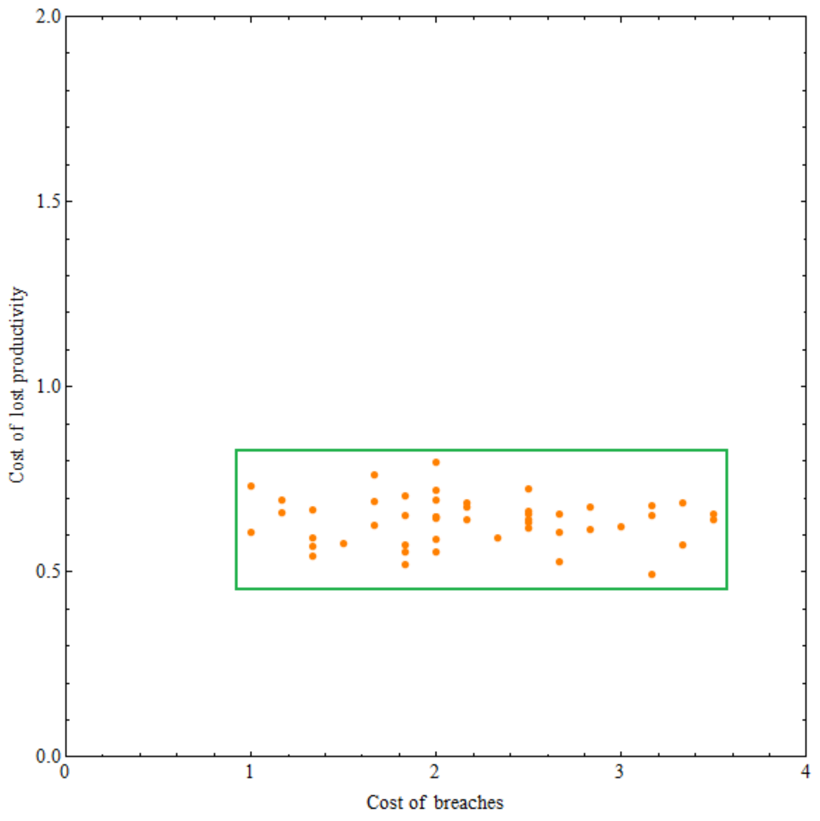


FIGURE 3. Scatter and Contour Plots: Password Length: 6 chars, Help Desk Staff: 1



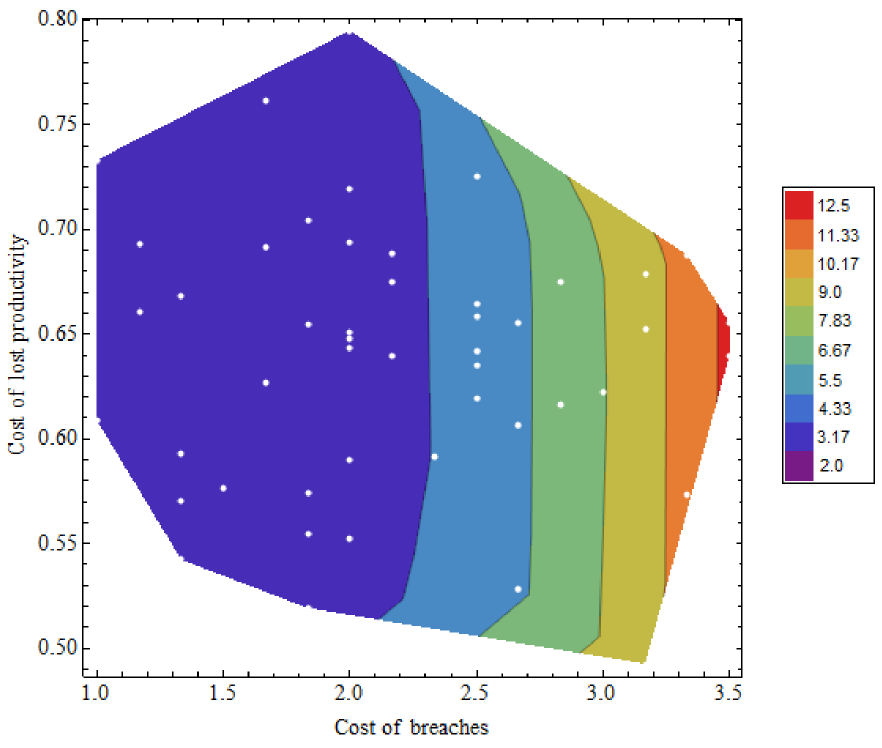
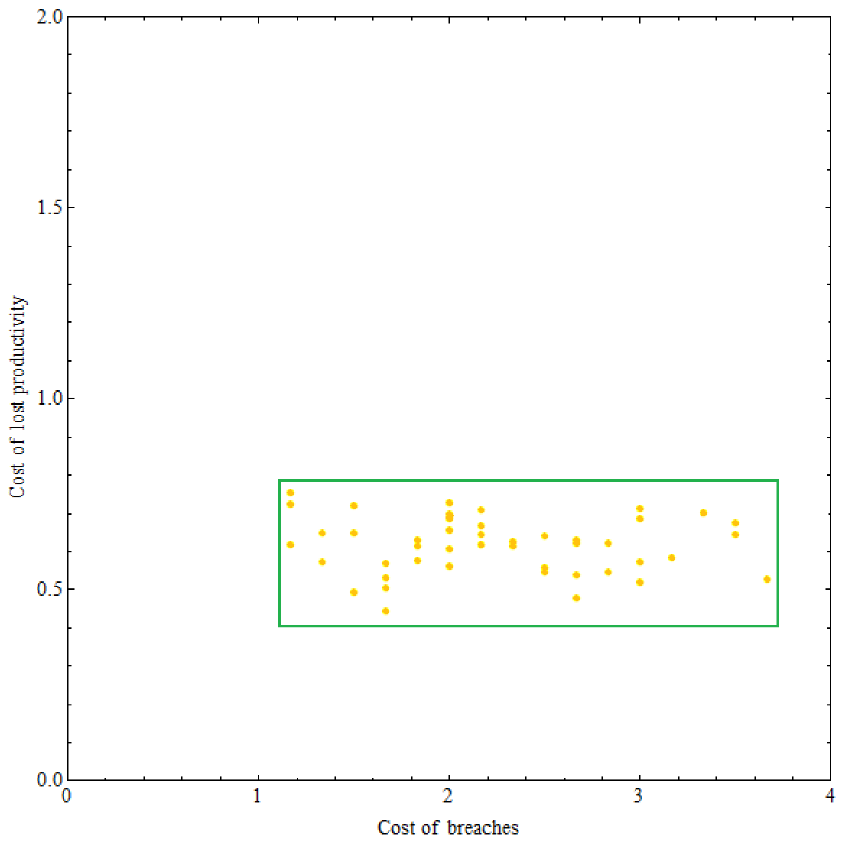


FIGURE 4. Scatter and Contour Plots: Password Length: 6 chars, Help Desk Staff: 2



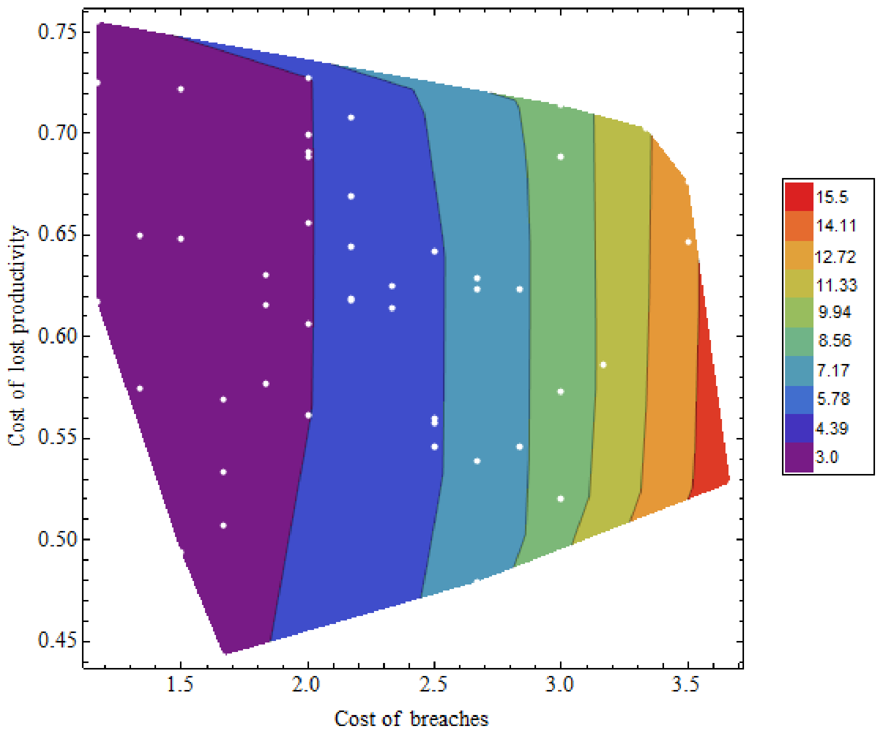
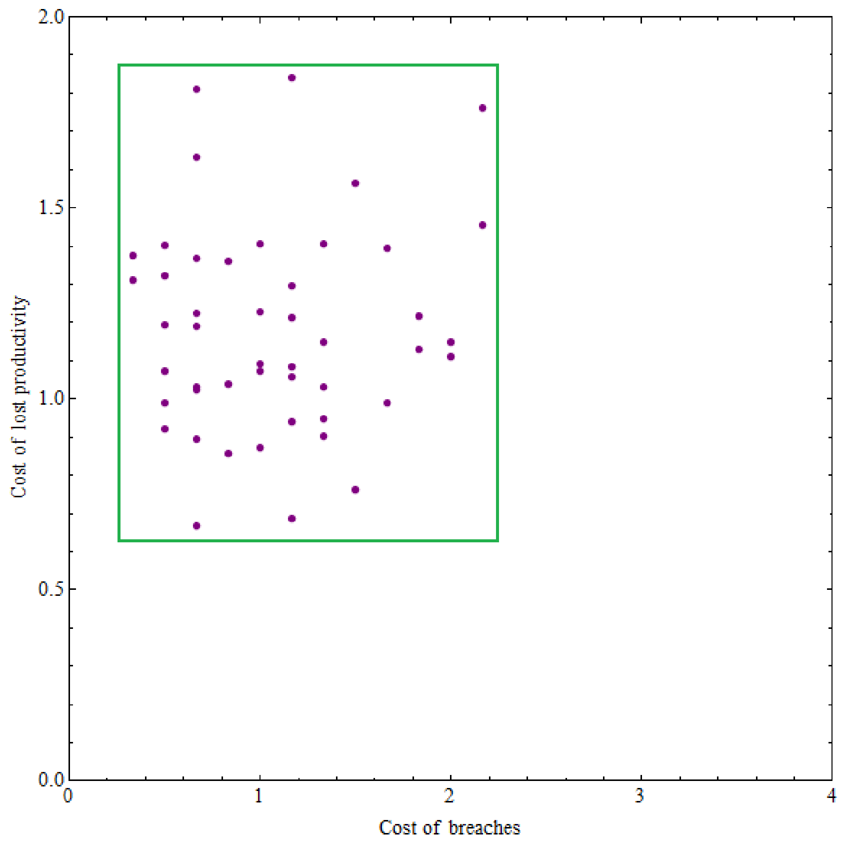


FIGURE 5. Scatter and Contour Plots: Password Length: 6 chars, Help Desk Staff: 3



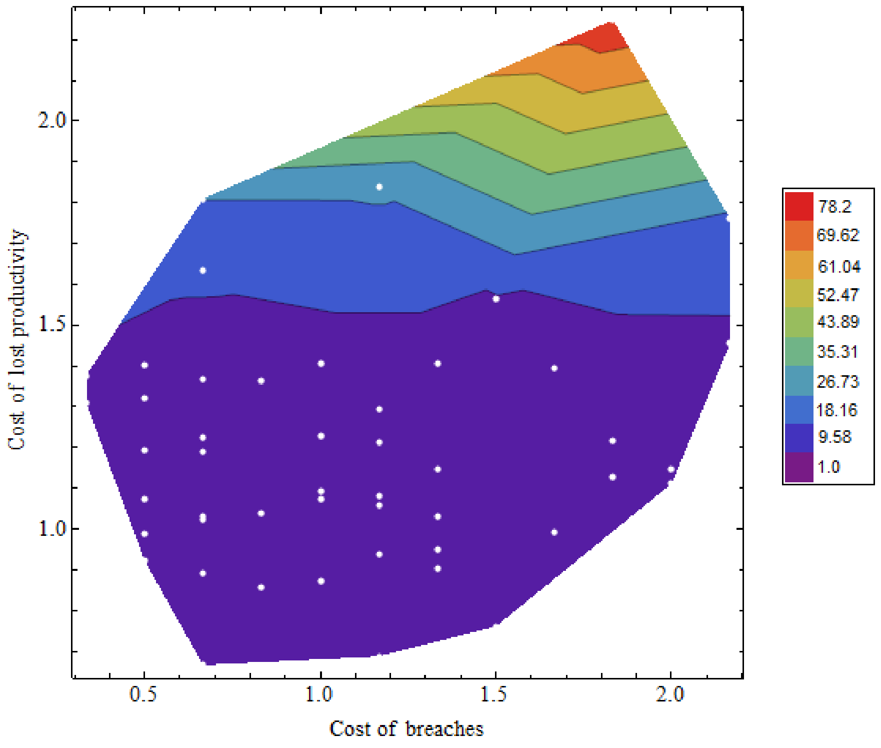


FIGURE 6. Scatter and Contour Plots: Password Length: 12 chars, Help Desk Staff: 1

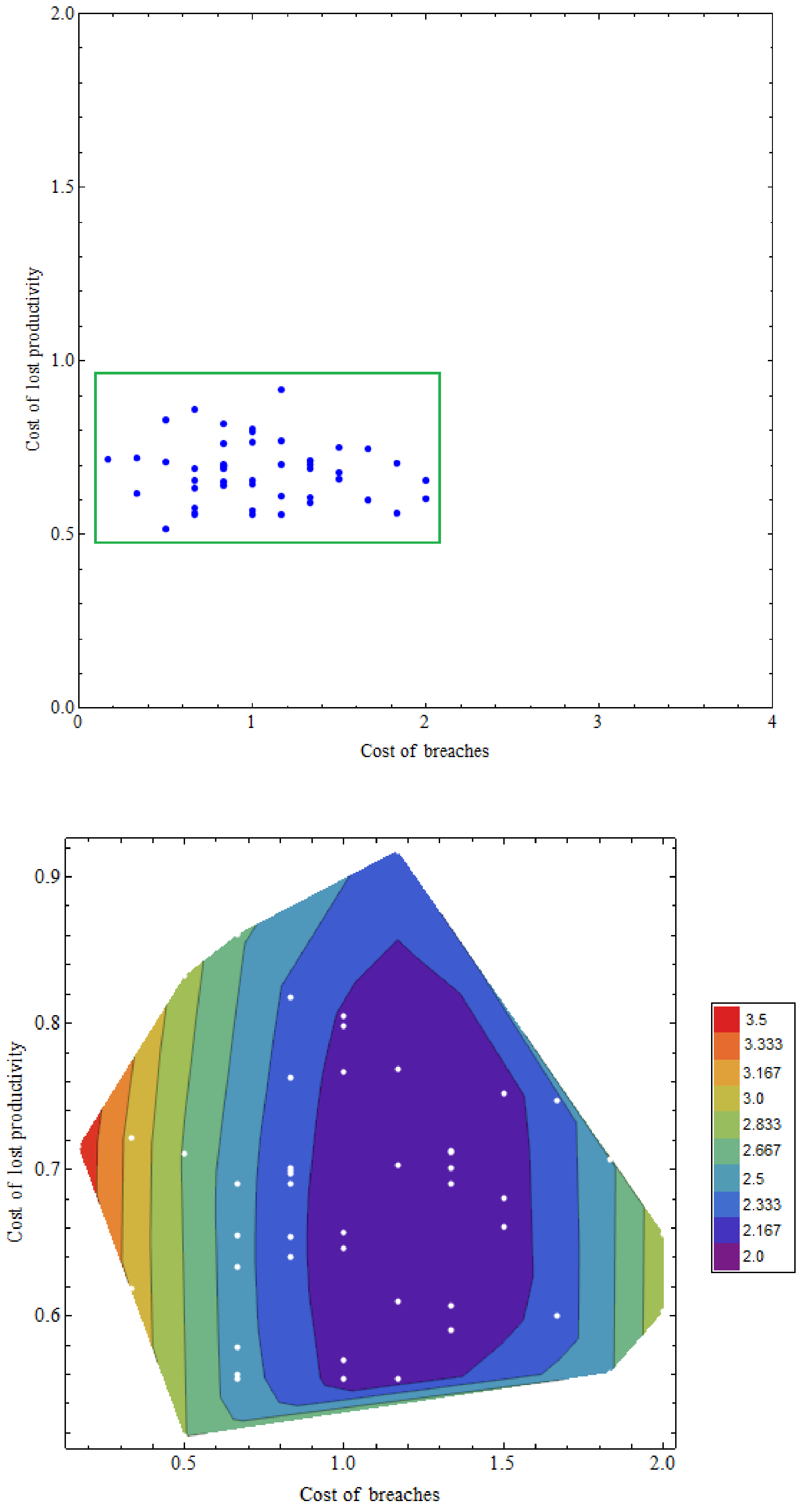
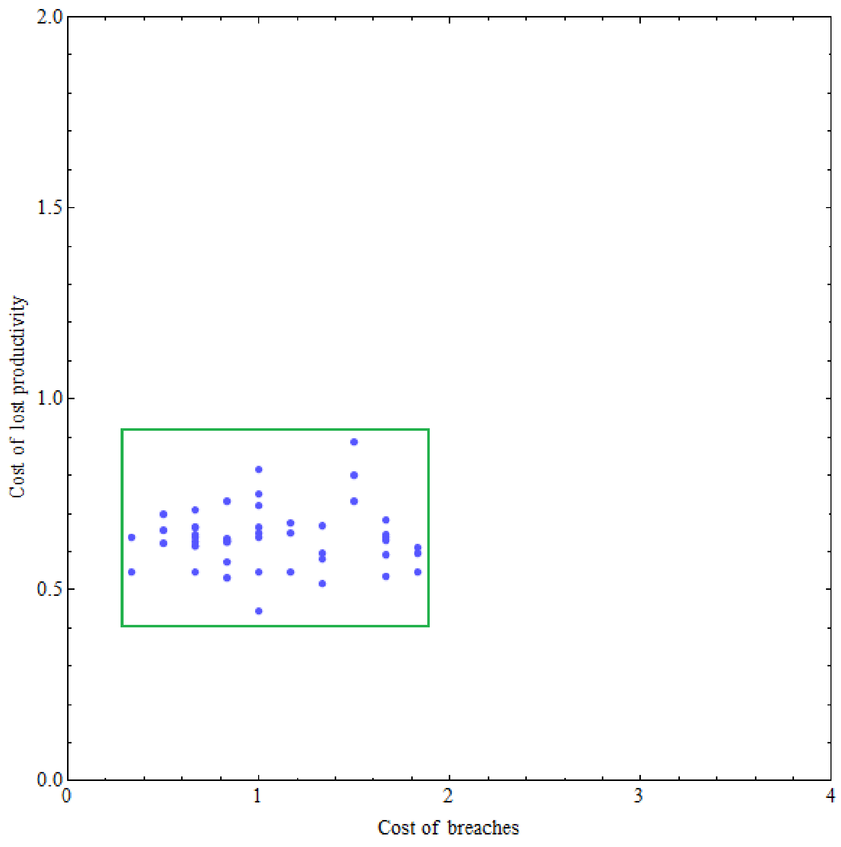


FIGURE 7. Scatter and Contour Plots: Password Length: 12 chars, Help Desk Staff: 2



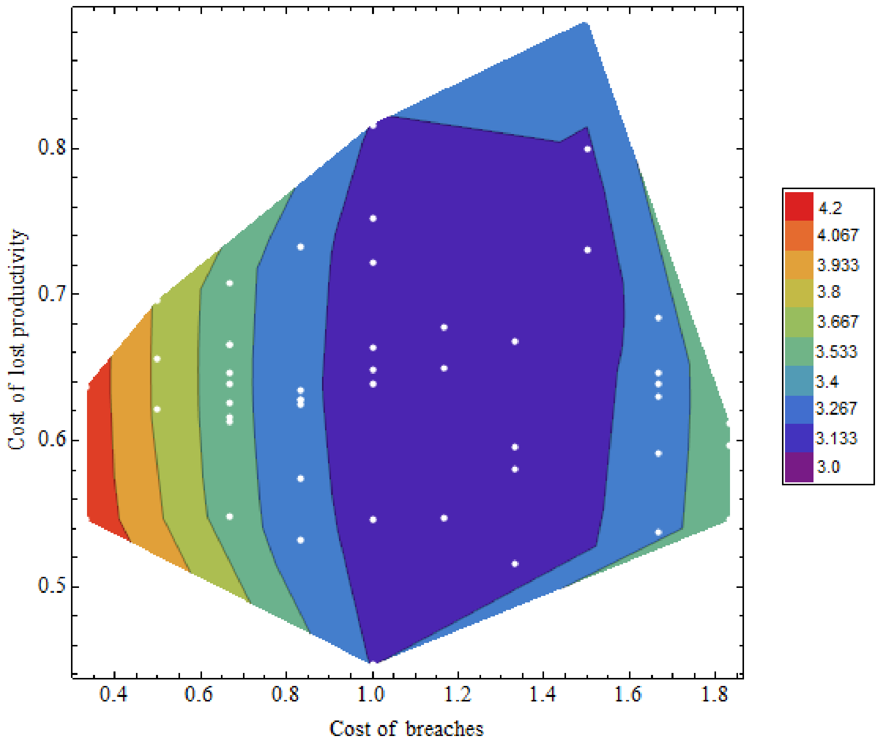
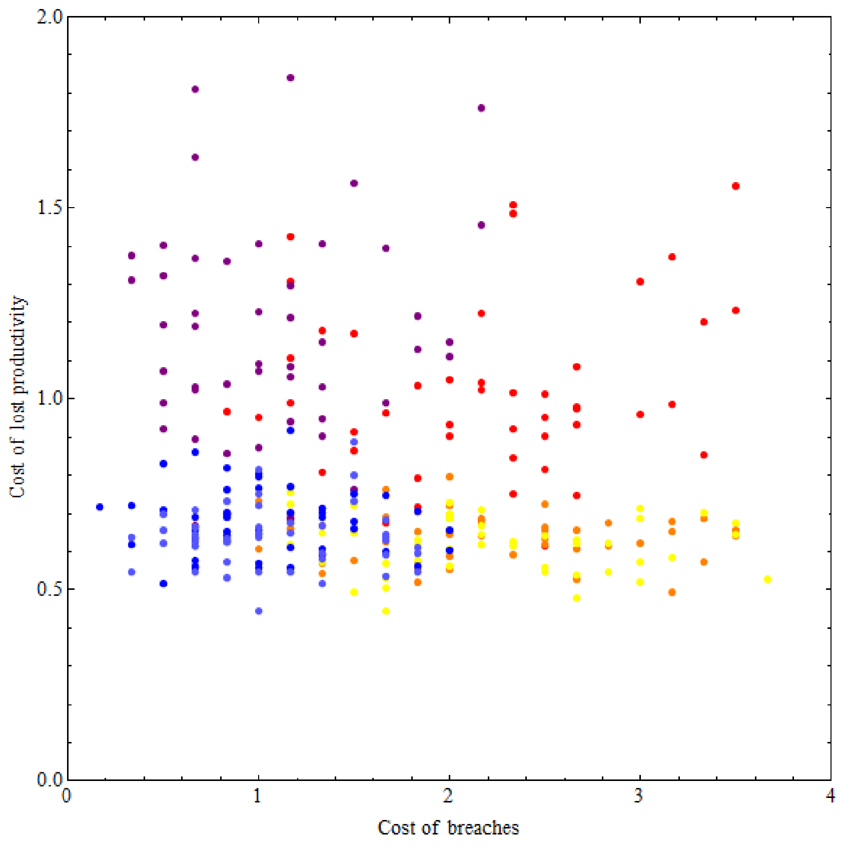


FIGURE 8. Scatter and Contour Plots: Password Length: 12 chars, Help Desk Staff: 3



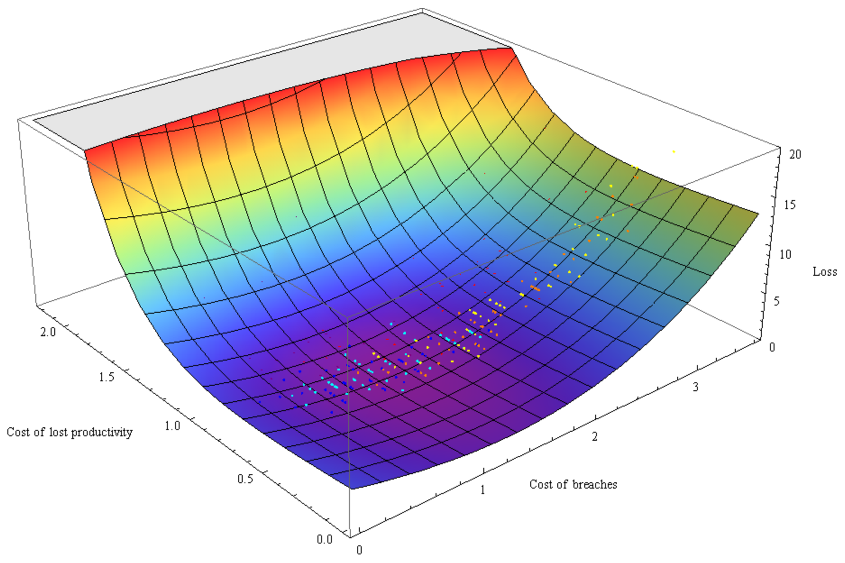
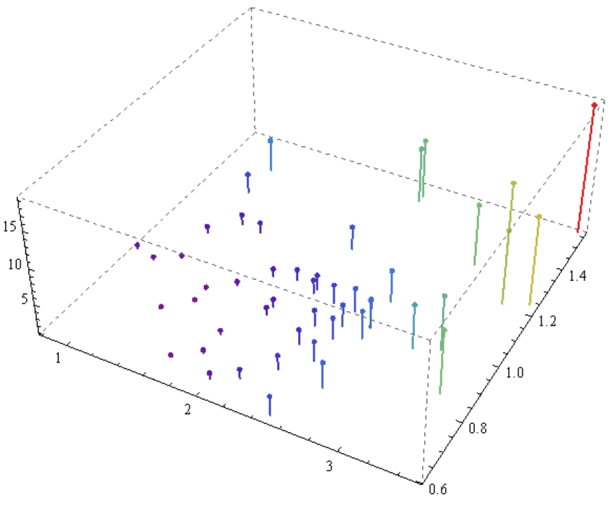
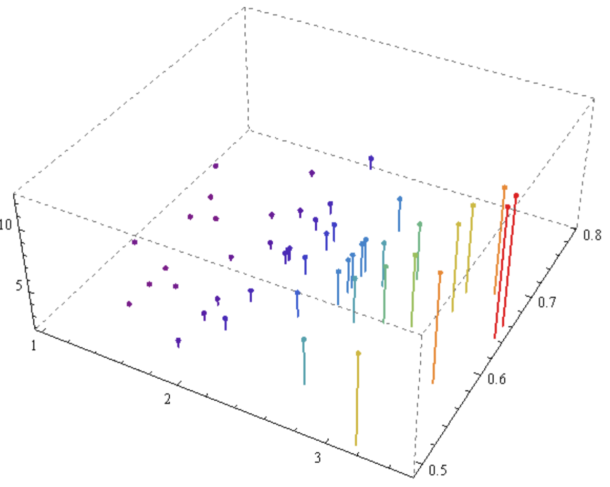


FIGURE 9. Scatter Plot and Loss Function Surface, covering all points





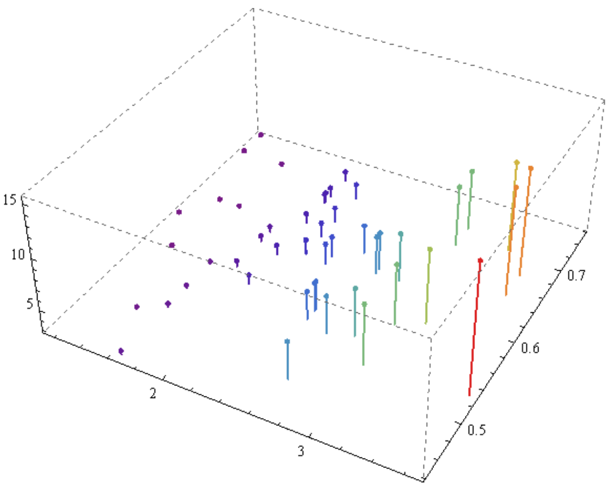


FIGURE 10. 3D Stick Plots: Password Length: 6 chars

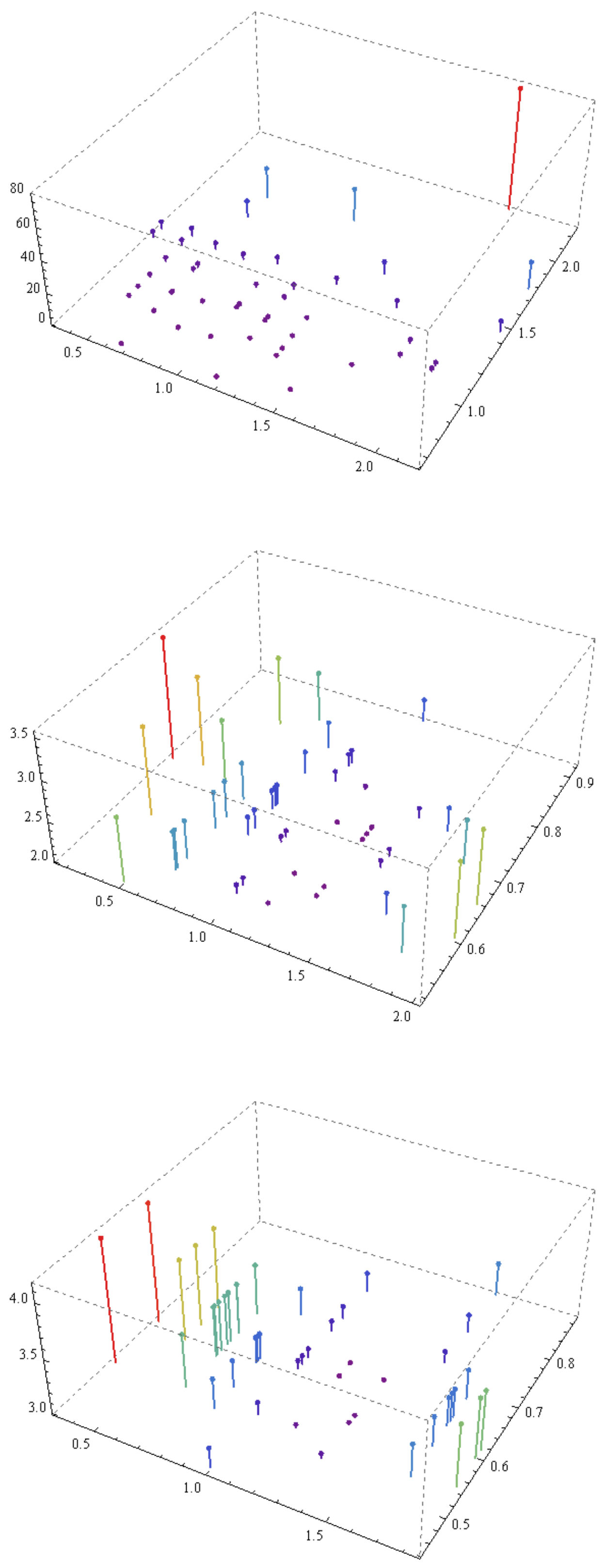


FIGURE 11. 3D Stick Plots: Password Length: 12 chars

6.2. Discussion of results. Inspecting the charts given above, we can visually see a number of important effects:

(1) Looking at the scatter plots, we can see that each policy alternative yields strong signs of forming clusters — of course, we naturally need more data and runs to confirm such formation. Nevertheless, the clustering we do see here naturally suggests that the length of password and the number of help staff does have a direct effect upon the various costs incurred;

(2) In the case of 6 character passwords, looking at the scatter plots we can see that the cost of breaches has a minimum value of 0.8 and rises to a maxium of 3.9. The cost of performance losses are well-distributed over the range starting from a value of 0.4, and reaching a maximum of 1.6;

(3) Comparing this with the results from the 12 character passwords shows us that the cost of breaches has been significantly reduced. In this case, the minimum value is 0.1 with the peak at 2.1. A difference can also be seen in the maximum productivity loss which has risen to 1.85. This accords with expectations in that a trade-off is being made between the security and usability of the password.

(4) Looking at the scatter plots for just one help desk staff, we can see that the clusters for both 6 and 12 character passwords are much larger than for clusters with two and three help desk staff. In fact the clusters are systematically greater for performance losses than for breach cost. Having at least two help desk staff reduces the range of performance loss costs dramatically (i.e., makes the clusters smaller in the performance loss direction). This applies to both 6 and 12 character passwords although the maximum productivity loss remains higher for 12 character passwords at all levels of help desk investment;

(5) Turning now to the contour plots of the loss function, we can see that the maximums of the overall losses incurred arises when 1 help desk staff is used, irrespective of password length (i.e., 20.0 for 6 chars and 78.2 for 12 characters). Note that the minimum losses incurred for either password length are identical (i.e., a value of i.0). This maximum loss is significantly reduced in both cases by investing in a second help desk operative.This suggests that using 1 help desk staff is inadequate and fails to reduce costs and overall losses sufficiently. The increased productivity loss for the 12 character passwords is not sufficient to demand additional staff to mitigate it;

(6) The clustering for either 2 or 3 help staff (separately under 6 or 12 passwords) are each very similar to the other - there appears to be similar costs incurred using either 2 or 3 help desk staff. This implies that there is little advantage to using either 2 or 3 help staff in terms of reducing costs indicating that 2 members of staff are sufficient in both cases to handle the majority of calls;

    That being said, the loss function contour plots for the 6 character passwords consistently generally show much higher loss (i.e., max. values of 12.5 and 15.5, for 2 and 3 help staff) against the smaller max values (3.5 and 4.2) for 12 characters and 2 or 3 help staff. This indicates that the trade-off between security and usability is in security's favour as we have managed to reduce our breach costs at a faster rate than we are incuring productivity costs.

From the above, we can conclude that:

- Using 1 help desk staff leads to greater costs of productivity loss in general, whereas either 2 or 3 help desk staff substantially reduces the cost of performance losses. This indicates using either 2 or 3 help desk staff - using 1 help desk staff is inadequate
- 6-character passwords generally have greater breach costs than 12-character passwords. While 12-character passwords increase productivity loss this value is lower than the savings through reduced breaches. Thus the overall utility score indicates that 12-character passwords should be used. .
- Comparing the outcomes using 12-character passwords and 2 or 3 help staff, it is clear from the loss function that using 2 help desk staff will incur the least losses overall, both in terms of minimum (2.0) and maximum (3.5) loss.

In the context of our original scenario, the security manager should therefore report back to the business manager that adopting a 12-character password is a feasible proposition. However, this will generate additional load on the help desk and that the study shows, via the large change from one to two help desk staff, a critical factor in maintaining productivity is adequate staffing in this area. In our study, the number of support calls generated was too low to add granularity to the staffing of the help desk and under both policies a single staff member was too low but two help desk staff were sufficient to handle the load. In a larger organization (or under a different reset policy) the optimal zone of staffing will likely differ between the policies. The 12-character policy did generate a higher number of reset calls (and thus productivity loss) but in this case the increase was not sufficient to require three staff members. Additionally, the model does not fully account for the impact on user effort of a 12-character password; this is discussed in more detail in the following section. Thus while we have made a good start in analyzing the impact of this decision there are several further refinements that could alter our current conclusion.

6.3. Future work. There are several key areas of our approach that we have targeted for revision and expansion in follow up work. User effort is currently under represented in the model and has the potential to significantly alter the conclusions drawn. Considering the notion of the Compliance Budget [4] we can see that the impact on the user of being forced to use 12-character passwords could push them to adopting insecure behaviours such as writing passwords down or sharing them with colleagues. The increased risk of breaches resulting from such behaviours has not been factored in and thus should be included in any future work.

Additionally there are two features that subsequent versions of the model will incorporate. These are multiple passwords per user and passwords with varying usage frequency. We can envisage that users may wish to access a range of services, some more frequently than others, each requiring a different password. Less frequently used passwords are more likely to be a cause of failed logins. Incorporating these changes will add significant richness to the model

Further changes may be included to allow a greater policy influence over the model behaviour. For example rather than having reset type controlled by a distribution we could allow this choice to be set by policy such that remote resets (at the PUBLIC or IN_TRANSIT locations) must always use the heavy reset. Again, this adds flexibility and richness to the model.

6.4. Conclusion. We have outlined a methodology that allows us to systematically explore security decision making through the use of a system model supported by empirical data. The conclusions we have drawn are based on the paramaterisation of the model and thus represent an outcome specifically related to the fictional organization we created for the purpose. The key point to understand here is not that 12-character passwords are better than 6-character, but that it is possible through a rigourous and repeatable process to investigate the impact of security decisions in a utility driven context. As the application of the utility function occurs after the dataset is generated it is possible for managers to dynamically explore different policies and preferences before making a decision and without repeatedly running the model. This allows security managers to efficiently make systematic decisions and, arguably more importantly, communicate them to other stakeholders in a familiar manner supported by data rather than intuition. The use of empirical data in this methodology greatly improves the strength of the results generated and adds confidence to the conclusions drawn.

7. Acknowledgements

We are grateful to many colleagues from the disciplines of the computing sciences, economics, and information security — including Simon Parkin, Martin Sadler, Simon Shiu, Aad van Moorsel, and Julian Williams — for their support of this work and their willingness to discuss it with us. The Systems Security Lab at HP Labs, Bristol and the HCS Group at UCL have provided an ideal environment within which to formulate these ideas.

References

[1] A. Adams and M. A. Sasse. Users are not the enemy: Why users compromise security mechanisms and how to take remedial measures. *Communications of the ACM*, 42(12):40–46, December 1999.

[2] George Bealer. Property theory: The type-free approach v. the church approach. *Journal of Philosophical Logic*, 23(2):139–171, 1994.

[3] A. Beautement, R. Coles, J. Griffin, C. Ioannidis, B. Monahan, D. Pym, A. Sasse, and M. Wonham. Modelling the Human and Technological Costs and Benefits of USB Memory Stick Security. In M. Eric Johnson, editor, *Managing Information Risk and the Economics of Security*, pages 141–163. Springer, 2008.

[4] A. Beautement, M.A. Sasse, and M. Wonham. The compliance budget: managing security behaviour in organisations. In *Proceedings of the 2008 Workshop on New Security Paradigms*, pages 47–58. ACM Digital Library, 2009.

[5] Y. Beres, D. Pym, and S. Shiu. Decision support for systems security investment. To appear, Proc. BDIM 2010, IEEE, 2010.

[6] G.M. Birtwistle. *Discrete Event Modelling on SIMULA*. Springer-Verlag, 1987.

[7] Robert Coles. Keynote Address, Eighth Workshop on the Economics of Information Security (WEIS 2009), University College London, England. 24–25, June 2009.
[8] M. Collinson, J. Griffin, and B. Monahan. The gnosis toolset. Forthcoming HP Labs Technical Report, 2009.
[9] M. Collinson, B. Monahan, and D. Pym. A logical and computational theory of located resource. *Journal of Logic and Computation*, 19(6):1207–1244, 2009. doi:10.1093/logcom/exp021.
[10] M. Collinson, B. Monahan, and D. Pym. A discipline of mathematical systems modelling. To appear: College Publications, London, 2010.
[11] M. Collinson, B. Monahan, and D. Pym. Semantics for structured systems modelling and simulation. In *Proc. Simutools 2010*. ICST: ACM Digital Library and EU Digital Library, 2010. ISBN: 78-963-9799-87-5.
[12] M. Collinson and D. Pym. Algebra and logic for resource-based systems modelling. *Mathematical Structures in Computer Science*, 19:959–1027, 2009. doi:10.1017/S0960129509990077.
[13] D. Florencio and C. Herley. A large-scale study of web password habits. In *Proceedings of the 16th international conference on the World Wide Web*. Association for Computing Machinery, May 2007.
[14] S. Gilmore and J. Hillston. The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. In *Proc. of the Seventh Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, number 794 in LNCS, pages 352–368. Springer-Verlag, 1994.
[15] Gnosis. `http://www.hpl.hp.com/research/systems_security/gnosis.html`.
[16] J. Hillston. Compositional Markovian modelling using a process algebra. In W. Stewart, editor, *Proceedings of the Second International Workshop on Numerical Solution of Markov Chains: Computations with Markov Chains*. Kluwer Academic Press, 1995.
[17] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
[18] P. Inglesant and M. A. Sasse. The True Cost of Unusable Password Policies: Password Use in the Wild. To be presented at CHI 2010: 28th ACM Conference on Human Factors in Computing Systems, April 10–15, Atlanta, GA, 2010.
[19] C. Ioannidis, D. Pym, and J. Williams. Information security trade-offs and optimal patching policies. Manuscript, 2009.
[20] C. Ioannidis, D. Pym, and J. Williams. Investments and trade-offs in the economics of information security. In Roger Dingledine and Philippe Golle, editors, *Proceedings of Financial Cryptography and Data Security '09*, volume 5628 of *LNCS*, pages 148–166. Springer, 2009. Preprint available at `http://www.cs.bath.ac.uk/~pym/IoannidisPymWilliams-FC09.pdf`.
[21] R.L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Wiley, 1976.
[22] Robert Lemos. Harnessing the cloud for hacking. *Technology Review*, December 2009.
[23] J. McColl. *Probability*. ButterworthHeinemann, 1995.
[24] Ellen Messmer. Data Breaches Get Costlier. *PC World, January 25, 2010*.
[25] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25(3):267–310, 1983.
[26] P.W. O'Hearn and D.J. Pym. The logic of bunched implications. *Bull. of Symb. Logic*, 5(2):215–244, 1999.
[27] Donn Parker. *Fighting Computer Crime: A New Framework for Protecting Information*. Wiley, 1992.
[28] D.J. Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*, volume 26 of *Applied Logic Series*. Kluwer Academic Publishers, 2002. Errata and Remarks maintained at: `http://www.cs.bath.ac.uk/~pym/BI-monograph-errata.pdf`.
[29] D.J. Pym, P.W. O'Hearn, and H. Yang. Possible worlds and resources: The semantics of BI. *Theoretical Computer Science*, 315(1):257–305, 2004. Erratum: p. 285, l. -12: ", for some $P', Q \equiv P; P'$ " should be "$P \vdash Q$".
[30] Francisco J. Ruge-Murcia. Inflation targeting under asymmetric preferences. *Journal of Money, Credit, and Banking*, 35(5), 2003.
[31] M.A. Sasse, S.Brostoff, and D. Weirich. Transforming the "weakest link": a human-computer interaction approach to usable and effective security. *BT Technology Journal*, 19(3):122–131, July 2001.
[32] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-theortic, Logical Foundations*. Cambridge University Press, 2009.
[33] H. Varian. A bayesian approach to real estate management. In S.E. Feinberg and A. Zellner, editors, *Studies in Bayesian Economics in Honour of L.J. Savage*, pages 195–208. North Holland, 1974.
[34] A. Zellner. Bayesian prediction and estimation using asymmetric loss functions. *Journal of the American Statistical Association*, 81:446–451, 1986.

APPENDIX A. THE GNOSIS CODE

```
-- Title     : Password model (pwd-sa-20100212a.gn)
-- Authors   : Simon Arnell, Brian Monahan

param day    = 1
param hour   = day / 8
param minute = hour / 60
param week   = day * 5
param year   = day * 365
param month  = year / 12


param runTime = year
param totalHelpDeskStaff = 2
param passwordLength_1 = 6

location HOME, WORK, PUBLIC, IN_TRANSIT

type time = num

type resetTypes       = LIGHT | HEAVY

type userTypes        = OFFICE_STAFF | ROAD_WARRIOR | EXECUTIVE

type captureTypes     = PARTIAL | FULL

type threatLevels     = LOW | MEDIUM | HIGH | HIGHEST

type complexityLevel  = SIMPLE | MODERATE | HARD

type authenticationStates = UN_AUTHENTICATED | AUTHENTICATED | FAILED | RESETTING | LOCKED_OUT

type helpdeskTicket = {| ticketID   :  int,                          // unique ID for the help desk job
                         submitted  :  time,                         // Time submitted
                         serviceID  :  int,                          // serviceID that the password is associated with
                         userID     :  int                           // user ID xof the user needing password reset
                      |}

type password = {| lastRenewal  : time,                              // time of last renewal
                   nextRenewal  : time,                              // time of next renewal
                   length       : int,                               // actual length of password
                   lockedOut    : logic,                             // is this "password" locked out ?
                   resetRequested : logic
                |}

type user   = {| userType          : userTypes,
                 currentLocation   : location,
                 timeLost          : num
              |}

type policy = {| minPasswordLength         : int,
                 complexity                : complexityLevel,
                 maxTries                  : int,
                 maxTime                   : time,
                 countermeasures           : logic,
                 notificationWindow        : int,
                 changeInterval            : int |}

bin newResetRequests       [] : helpdeskTickets
bin completedResetRequests [] : helpdeskTickets

array passwords[int, int] : password
array users[int] : user
array capturedPasswordData[int,int] : int = { 0 }

array passwordChangeHistogram[int] : int = { 0 }

param chooseResetType : resetTypes = point [(2/3, LIGHT), (1/3, HEAVY)]

param newPasswordThreshold : int = 2                                 //days
param threatLevel : threatLevels = MEDIUM

param randomReal   =  uniform (0, 1)
param timeTaken    =  normal (6, 2)

param userCount : int = 100

table tries[int] : int
tries[5] = point [(0.95,1),(0.025,2),(0.015,3),(0.0095,4),(0.0005,5)]
```

```
table policies[int] : policy
policies[1] = policy {| minPasswordLength = passwordLength_1,
                        complexity = MODERATE,
                        maxTries = 5,
                        maxTime  = 25,
                        countermeasures = true,
                        notificationWindow = 14,
                        changeInterval = 90 |}

table resetTypeTimes[resetTypes] : num
resetTypeTimes[LIGHT] = negexp(minute * 15)
resetTypeTimes[HEAVY] = negexp(hour * 4)

table userTypeCount[userTypes] : int                                   // number of each user_type to create
userTypeCount[EXECUTIVE] = floor(10 * userCount / 100)
userTypeCount[OFFICE_STAFF] = floor(70 * userCount / 100)
userTypeCount[ROAD_WARRIOR] = floor(20 * userCount / 100)

table threatEnvironmentMultiplier[threatLevels] : num
threatEnvironmentMultiplier[MEDIUM]    = 1/10000
threatEnvironmentMultiplier[LOW]       = threatEnvironmentMultiplier[MEDIUM] / 2
threatEnvironmentMultiplier[HIGH]      = threatEnvironmentMultiplier[MEDIUM] * 2
threatEnvironmentMultiplier[HIGHEST]   = threatEnvironmentMultiplier[HIGH] * 2

table locationCapture[location] : captureTypes
locationCapture[WORK]         =  point[(3/5,  PARTIAL), (2/5, FULL)]
locationCapture[PUBLIC]       =  point[(1/2,  PARTIAL), (1/2, FULL)]
locationCapture[IN_TRANSIT]   =  point[(3/5,  PARTIAL), (2/5, FULL)]
locationCapture[HOME]         =  point[(1/3,  PARTIAL), (2/3, FULL)]

table userLocation[userTypes] : location
userLocation[EXECUTIVE]       = point[(0.2, HOME), (0.7, WORK), (0.1, IN_TRANSIT)]
userLocation[OFFICE_STAFF]    = WORK
userLocation[ROAD_WARRIOR]    = point[(0.2, HOME), (0.2, WORK), (0.1, PUBLIC), (0.5, IN_TRANSIT)]

var newUserID : int = 0
var newTicketID : int = 0

param newPasswordMultiplier : num = 20
param taskInterarrival : num = negexp(20 * minute)
param waitForReset : num = negexp(10 * minute)
param numberOfServices : int = extent policies

param chooseService = 1

param portalBreach =  bernoulli(threatEnvironmentMultiplier[threatLevel])

array userTypeTimeLost[userTypes] : num = { 0.0 }
var userTypeTimeLost_OFFICESTAFF  : num = 0.0
var userTypeTimeLost_ROADWARRIOR  : num = 0.0
var userTypeTimeLost_EXECUTIVE    : num = 0.0

var breaches = 0

function complexityDependency (complexity : complexityLevel) : num =
   ( complexity == SIMPLE    → (9999/10000)
   | complexity == MODERATE  → (9998/10000)
   | complexity == HARD      → (9996/10000)
   )

function lengthDependency (length : int) : num =
   ( length > 5   → (1 - ((length - 6)/5000))
   | 1
   )

function triesDependency (maxTries : int, tries : int) : num =
   ( tries >= maxTries → 0
   | 1
   )

function timingDependency (overallTiming : time, maxTime : time) : num =
   ( overallTiming > maxTime → 0
   | 1
   )


function ageDependency (age : int) : num =
   ( age <= 2 → 0.999
   | 1 - (0.001 *(1/(age - 1)))
   )

function successProcPolicy
         (length : int,
          complexity : complexityLevel,
          maxTries : int,
          maxTime : time,
          entryTries : int,
          overallTiming : time,
          age : int) : num =
   complexityDependency(complexity)         *
   lengthDependency(length)                 *
   triesDependency(maxTries, entryTries)    *
   timingDependency(overallTiming, maxTime) *
   ageDependency(age)

function successProb
         (policy : policy,
          entryTries : int,
          overallTiming : time,
          age : int) : num =
   successProcPolicy(
     policy#minPasswordLength,
     policy#complexity,
     policy#maxTries,
     policy#maxTime,
     entryTries,
     overallTiming,
     age)

function getSuccessProbability (userID : int, serviceID: int, overallTiming : time, entryTries : int) : num =
   successProb (
     policies[serviceID],
     entryTries,
     overallTiming,
     getPasswordAge(userID, serviceID))

function getFailureProbability (userID : int, serviceID: int, overallTiming : time, entryTries : int) : num =
   1 - getSuccessProbability(userID, serviceID, overallTiming, entryTries)

function getPasswordAge (userID : int, serviceID: int) : int =
   (floor((NOW - passwords[userID, serviceID] # lastRenewal) / day))

function getNewRenewal (serviceID : int) : num =
   (floor
     ( NOW +
       (policies[serviceID] # changeInterval * day) -
       (discrete (0, policies[serviceID] # notificationWindow,
                     policies[serviceID] # notificationWindow - 2) * day
       )
     )
   )

function getNumberOfCharactersCaptured (minPasswordLength : int ) : int =
   (discrete(0,minPasswordLength,minPasswordLength - 2))


  // This represents the user's activities ...
process user (userType : userTypes) {
   newUserID += 1
   var userID : int = newUserID
   users[userID] := user {| userType = userType, currentLocation = none, timeLost = 0 |}

   for serviceID = 1 to numberOfServices {
     var firstRenewal = uniform(0, policies[serviceID] # changeInterval)
     passwords[userID,serviceID] :=
       password {| lastRenewal = NOW, nextRenewal = floor(firstRenewal), lockedOut = false, resetRequested = false |}
   }

   var serviceID : int = -1
   var user : user = users[userID]
   var password : password = none
   var authenticationState : authenticationStates = none
   var failureProbability : num = -1
   var captureType : captureTypes = none
   var random : num = 0

   repeat {
   hold(taskInterarrival)
```

```
    serviceID := chooseService
    password := passwords[userID, serviceID]
    users[userID] := user with {| currentLocation = userLocation[userType] |}
    authenticationState := UN_AUTHENTICATED

    while[password # lockedOut == true] {
    select [password # resetRequested == false] {
        authenticationState := RESETTING
        call startPasswordReset(userID, serviceID)
      } or else {
      hold(waitForReset)
      }
    }
    var overallTiming = timeTaken
    var entryTries = tries[policies[serviceID] # maxTries]
    failureProbability := getFailureProbability(userID, serviceID, overallTiming, entryTries)

    random := randomReal
  if[random <= failureProbability] {
    authenticationState := FAILED
      authenticationState := LOCKED_OUT
      passwords[userID, serviceID] := password with {| lockedOut = true |}
      authenticationState := RESETTING
      call startPasswordReset(userID, serviceID)
    }
    authenticationState := AUTHENTICATED
    random := randomReal
    if[random <= threatEnvironmentMultiplier[threatLevel]] {
      captureType := locationCapture[users[userID] # currentLocation]
      select[captureType == PARTIAL] {
        capturedPasswordData[userID,serviceID] :=
          capturedPasswordData[userID,serviceID] +
          getNumberOfCharactersCaptured(policies[serviceID] # minPasswordLength)
      } or [captureType == FULL] {
        capturedPasswordData[userID,serviceID] :=
          capturedPasswordData[userID,serviceID] + policies[serviceID] # minPasswordLength
      }
    }
  }
}


routine startPasswordReset (userID : int, serviceID : int) {
  var currentResetType = chooseResetType
  var newResetRequest : helpdeskTicket = none
  var password : password = none

  passwordChangeHistogram[getPasswordAge(userID, serviceID)] :=
    passwordChangeHistogram[getPasswordAge(userID, serviceID)] + 1

  if [currentResetType == LIGHT] {
  call resetPassword (userID, serviceID)
  var portalBreachEvent = portalBreach
  if[portalBreachEvent == 1] {
    breaches += 1
  }

  var elapsedTime = resetTypeTimes[LIGHT]
    var user = users[userID]
    var timeLost = user#timeLost + elapsedTime
    users[userID] := user with {| timeLost = timeLost |}

  }
  or [currentResetType == HEAVY] {
    newTicketID += 1
    newResetRequest :=
        helpdeskTicket {| ticketID = newTicketID, submitted = -1, serviceID = serviceID, userID = userID |}
        password := passwords[userID, serviceID]
        passwords[userID, serviceID] := password with {| resetRequested = true |}

    call submitPasswordResetRequest (newResetRequest)               // This lets user get on with something else ...
  }
}



 //  The point of this is to handle admin of password locking out - its not really done by the user!!
routine  submitPasswordResetRequest  (resetRequest : helpdeskTicket) = {
  var userID     =  resetRequest#userID
  var serviceID  =  resetRequest#serviceID
  var pwd = passwords[userID,serviceID]

  passwords[userID, serviceID] := pwd with {| lockedOut = true |}   // user is locked out for given service
  resetRequest := resetRequest with {| submitted = NOW |}
  put [resetRequest] into newResetRequests                          // request queue
}


routine resetPassword (userID : int, serviceID : int) {
  var pwd = passwords[userID,serviceID]
    passwords[userID,serviceID] :=
      pwd with {|
        lastRenewal = NOW,
        nextRenewal = getNewRenewal(serviceID),
        lockedOut = false,
        resetRequested = false |}
    capturedPasswordData[userID, serviceID] := 0
}


process helpDeskOps {
  var currentTicket : helpdeskTicket = none
  var completedResetRequest : helpdeskTicket = none

  var userID : int = -1
  var serviceID : int = -1
  var submitted = -1

  var elapsedTime = -1

  var user : user = none
  var timeLost = -1

  var pwd : password = none

                                                                    // Help Desk Task-- process the password
  repeat {
    get [currentTicket] from newResetRequests

    userID      :=  currentTicket#userID
    serviceID   :=  currentTicket#serviceID

    elapsedTime := currentTicket#submitted

    pwd         :=  passwords[userID, serviceID]


    hold(resetTypeTimes[HEAVY])                                     // do the work now ...

    call resetPassword(userID,serviceID)                            // reset password state + captured

    elapsedTime := (NOW - elapsedTime)                              // calc the elapsed Time ....

    user := users[userID]
    timeLost := user#timeLost + elapsedTime
    users[userID] := user with {| timeLost = timeLost |}
  }
}


process passwordAging {
  launch passwordAging after day
  var password : password = none
  for userID = 1 to newUserID {
    for serviceID = 1 to numberOfServices {
      password := passwords[userID,serviceID]
      if[password # nextRenewal < (NOW + day) ] {
        passwords[userID, serviceID] := password with {| lockedOut = true |}
        call startPasswordReset (userID,serviceID)
      }
      or [capturedPasswordData[userID,serviceID] >= policies[serviceID] # minPasswordLength] {
        breaches += 1
        if[policies[serviceID] # countermeasures==true] {
          call startPasswordReset (userID,serviceID)
        }
      }

      hold(minute / 60 / 1000)
    }
  }
```

```
}


do  totalHelpDeskStaff          { launch helpDeskOps       }

do  userTypeCount[OFFICE_STAFF]  { launch user(OFFICE_STAFF) }
do  userTypeCount[ROAD_WARRIOR]  { launch user(ROAD_WARRIOR) }
do  userTypeCount[EXECUTIVE]     { launch user(EXECUTIVE)    }


launch passwordAging after day

hold(runTime)

trace ("=====================  USER INFO  ==============================")
for userID = 1 to newUserID {
  var user  : user       =  users[userID]
  var utype : userTypes  =  user#userType

  userTypeTimeLost[utype] := (userTypeTimeLost[utype] + user # timeLost)
}

userTypeTimeLost_OFFICESTAFF := userTypeTimeLost[OFFICE_STAFF]
userTypeTimeLost_ROADWARRIOR := userTypeTimeLost[ROAD_WARRIOR]
userTypeTimeLost_EXECUTIVE := userTypeTimeLost[EXECUTIVE]


trace ("===============  PASSWORD CHANGE DEBUG  ========================")
for i = 0 to extent passwordChangeHistogram {
  trace("day %v - %v", i, passwordChangeHistogram[i])
}

dump()

close
```

HP LABS, BRISTOL[2]
E-mail address: simon.arnell@hp.com

UCL AND HP LABS, BRISTOL
E-mail address: a.beautement@cs.ucl.ac.uk

UCL
E-mail address: p.inglesant@cs.ucl.ac.uk

HP LABS, BRISTOL
E-mail address: brian.monahan@hp.com

UNIVERSITY OF ABERDEEN AND HP LABS, BRISTOL
E-mail address: d.j.pym@abdn.ac.uk

UCL
E-mail address: a.sasse@cs.ucl.ac.uk

[2]Now with HP Information Security, Bracknell