Measuring the Security Impacts of Password Policies Using Cognitive Behavioral Agent-Based Modeling

Vijay Kothari
Department of Computer Science
Dartmouth College
vijayk@cs.dartmouth.edu

Jim Blythe
Information Sciences Institute
University of Southern California
blythe@isi.edu

Sean W. Smith
Department of Computer Science
Dartmouth College
sws@cs.dartmouth.edu

Ross Koppel
Department of Sociology
University of Pennsylvania
rkoppel@sas.upenn.edu

ABSTRACT

Agent-based modeling can serve as a valuable asset to security personnel who wish to better understand the security landscape within their organization, especially as it relates to user behavior and circumvention. In this paper, we argue in favor of cognitive behavioral agent-based modeling for usable security, report on our work on developing an agent-based model for a password management scenario, perform a sensitivity analysis, which provides us with valuable insights into improving security (e.g., an organization that wishes to suppress one form of circumvention may want to endorse another), and provide directions for future work.

1. INTRODUCTION

Agent-based models incorporating user behavior, emotion, and cognition can serve as valuable tools that assist computer security personnel design, implement, and maintain security systems, devise security policies, and employ security practices that are congruent with security and other organizational objectives.

Indeed, as the current state of security practice indicates, we need these sorts of tools. Our interviews, surveys, and observations reflect many examples where security fails to accommodate users. Such mismatches between user needs and security policies and mechanisms often induce circumvention, thereby undermining overall objectives. Even if one could design adequate security policies and mechanisms a priori, the dynamic nature of software systems, user needs, and organizational and environmental changes would necessitate frequent readjustments. Consequently, we need tools that allow us to better understand computer security's costs, common perceptions and misperceptions, side effects, and interactions.

HotSoS '15, April 21 - 22, 2015 Urbana, IL USA

http://dx.doi.org/10.1145/2746194.2746207

DASH [1, 2], an agent-based simulation framework that supports the dual-process model of cognition, reactive planning, modeling of human deficiencies (e.g., fatigue, frustration), and multi-agent interactions, enables us to create such tools. In DASH, users are represented as agents with weighted goals, plans to achieve those goals, attributes, knowledge, and abilities. These agents use mental models and have perceptions of the world that often depart from reality. Agents, in accordance with their mental models, take actions, observe and interpret events, and communicate. They dynamically compute and recompute goals and the plans they use to achieve them. DASH models may better enable security personnel to (a) identify weaknesses in security policies and mechanisms, e.g., workflow impediments that prompt user circumvention, (b) estimate the likelihood of user engagement in workarounds, (c) gauge the number of inescapable security infractions from policy-workflow mismatches, (d) estimate the values of security and organizational objective functions, (e) test the accuracy of proxy security measures, and (f) measure how shifts in the environment affect security. A cognitive and behavioral approach to modeling can provide insights into the effectiveness of informing users of practical needs for security, implementing a feedback loop, imposing more stringent policies or harsher penalties for circumvention, and more.

Agent-based modeling is particularly enlightening in scenarios where security in practice radically differs from security in the abstract, where it's extraordinarily challenging to anticipate how emotions, cognitive biases, and other human deficiencies may affect user behavior. Indeed, in order to get security right it is critical that we understand how users interact with our systems. And we must adapt our systems to our users (and not expect our users to adapt to our systems!) so as to induce "good" behavior [3, 4]. In previous work [5] we discussed the potential for agent-based models to be applied to prediction of human circumvention of security, relayed an anecdote regarding timeouts in a medical setting, explained our preliminary work, and discussed our future directions for building such models. In this paper, we follow up on this work by detailing our progress on modeling the password management scenario.

The password management scenario involves establishing password polices for an enterprise. In theory, having a policy that requires users to use strong passwords, to never write

them down, and to never reuse them across sites would improve security. In practice, users commonly circumvent password policies due to perceived cognitive limitations, fatigue, frustration, and work culture; and password choices and password management practices for one service may affect the choices and practices for another, making the services interdependent. By applying agent-based models, security personnel can better understand this complex environment, estimate measures of aggregate security that incorporate circumventions, risks, and costs, and ultimately make better decisions.

This paper is structured as follows. In Section 2, we introduce the DASH modeling framework. in Section 3 we investigate the password modeling scenario, detail our DASH modeling work, perform a sensitivity analysis, and discuss results and takeaways. In Section 4 we discuss future work including the autologout scenario. In Section 5 we conclude.

2. THE DASH AGENT MODELING PLATFORM

The DASH agent modeling platform provides a framework and a set of capabilities for modeling human behavior [1], designed to capture observations from human-centered security experiments, e.g. [6]. In order to model human task-oriented behavior, which is both goal-directed and responsive to changes in the environment, DASH includes a reactive planning framework that re-assesses goal weights and plans after receiving input after an action [7]. In order to model deliberative behavior, DASH includes an implementation of mental models following the approach of Johnson-Laird and others [8] and a simple framework for evaluating costs and benefits of alternative worlds. This approach adopts the view that users follow essentially rational behavior when making decisions about on-line actions including security, but typically have an incomplete or incorrect model of the security landscape.

In order to model bounded attention that affects human decision-making, particularly under stress or cognitive load, DASH adapts psychology's *dual-process framework* [9] in which two modules provide alternative suggestions for the agent's next action. The first is a deliberative system that uses the mechanisms for planning and mental models to arrive at a decision, and the second is a stimulus-driven system that matches surface properties of a situation to find an answer. Once an agent has experience in a domain, the stimulus system provides good answers most of the time while an inexperienced agent may need to fall back on deliberative reasoning more often. Under stress, time pressure, or cognitive load, the deliberative system may not complete, or the stimulus system may gain increased weight, leading to impulsive behavior that may not be correct.

Other cognitive architectures such as SOAR [10] or ACT-R [11] provide many of the same behaviors. One distinguishing factor of DASH is that its stimulus system is not related to the deliberative system by a compilation learning process and can often produce results that differ qualitatively rather than in terms of speed. DASH also provides support for mental models and tradeoff analysis as more fundamental components.

3. THE PASSWORD MANAGEMENT SCENARIO: SECURITY DEPENDENCIES INTRODUCED BY WORKAROUNDS

With an understanding of DASH we now discuss our password simulation. In Section 3.1 we cover preliminaries including a discussion of password problems and related work. In Section 3.2 we explain how our simulation works. In Section 3.3 we provide and discuss our results. In Section 3.4 we provide takeaways.

3.1 Preliminaries

In terms of usability and security, many consider passwords a failure. Users are notorious for choosing weak passwords. In an effort to mitigate the security risks linked to weak passwords, many services now require users to choose passwords that satisfy complex password composition rules. Unfortunately, this brings with it a slew of other security challenges (e.g., [3, 4, 12, 13]). Users who are unable to cope with the increased cognitive demands of having to remember dozens of passwords resort to circumventing password policies and employing poor password management strategies; they write passwords down on Post-it notes, reuse passwords across multiple services with little or no variation, and leave passwords in plaintext files on their computers. However, perceived cognitive limitations are not the only impetus for user circumvention of password policies. In some domains, users need to share information with others who have different access rights than themselves, but the "proper" channel for information sharing is slow and inefficient. So, they share passwords instead [3].

Services are culpable too. Some services effectively discourage strong passwords by setting low ceilings on password length, disallowing special characters, using easily guessable security questions, and assigning default passwords that are often left unchanged. Others impose excessive password complexity requirements and require frequent password resets, which further incentivizes users to circumvent. In recent years, many services have also been the target of massive password breaches; in some cases, they have even exposed passwords to malicious actors in cleartext. Moreover, due to password reuse, risks associated with poor password practices are not confined to those services that are lax about password security. That is, the security of even those services that make good efforts to secure user passwords can easily be compromised by vulnerabilities on other sites [14].

While tremendous effort has been spent on trying to replace passwords, it has been met with questionable success. Bonneau et al [15] compared passwords to other authentication schemes in three domains: usability, deployability, and security. They showed that no alternative authentication scheme dominates passwords.

In short, passwords pose numerous memorability and usability challenges that frequently manifest in user circumvention. They pose confidentiality, risk mitigation, and public perception challenges for services. And they do not appear to be going away any time soon. This motivates the need for better techniques to both assess and mitigate costs and risks associated with password policies.

Numerous recent studies have looked into password modeling. Shay et al [16] developed a simulation for understanding the effectiveness of password policies. Choong [17] proposed a user-centric cognitive behavioral framework for

the password management lifecycle, from password creation to password reset. SimPass is a highly configurable agent-based model for measuring the efficacy of password policies [18]. Our work is similar to SimPass in that we've developed a password simulation with knobs that can be adjusted to measure aggregate security associated with password policies under different circumstances. Whereas SimPass employs numerous parameters to better understand password management scenarios with minimal assumptions, we adopt the view that many of these parameters cannot be known, nor do they need to be known, *a priori* to have a useful predictive model. Our simulation instead relies on a smaller number of parameters with more underlying models, especially those related to cognition and behavior. For example, there are underlying models for a password belief system and cognitive burden. While this approach provides valuable insights into the cognitive and behavioral factors that affect security, it necessitates different kinds of modeling assumptions.

3.2 Simulation Details

Our simulation models human users interacting with computer systems that employ username and password authentication. Specifically, agents construct plans to achieve high-level subgoals for creating accounts, signing in to accounts, and signing out of accounts. Each of these subgoals are broken down by the agent into a series of steps during action invocation as determined by the agent's beliefs, the agent's cognitive burden, and other factors. To better understand how these processes work, we first explore the underlying models for the agent's belief system and the agent's cognitive burden.

Let us first briefly discuss the agent's password belief system. For this discussion, we limit ourselves to passwords; a similar model exists for usernames. For service S and password P, let $V_{S,P}$ denote the strength of the agent's belief that password P is the correct password for her account on service S. During the sign-in process, these password strength values are used to determine whether the agent recalls a password for a given service and, if so, which password the agent recalls. Agents slowly forget passwords during periods of non-use as reflected by reductions in password strengths. In fact, following every user action, all password strengths over all services are decremented by service-specific password forget rates.

We now discuss the underlying model for cognitive burden. As before, we limit our discussion to passwords. The model uses a generalization of the Levenshtein distance to sets and makes use of an openly available Prolog implementation of Levenshtein distance [19]. The Levenshtein distance between a string S_1 and S_2, denoted as $Lev(S_1, S_2)$, is the minimum number of character insertions, deletions, and substitutions required to convert S_1 into S_2. For set S, define the Levenshtein measure $L(S)$ as the weight of a minimum spanning tree T over the vertex set $S \cup \{\epsilon\}$ for which edge weights are specified as $w(v_1, v_2) = Lev(v_1, v_2)$. Here, ϵ denotes the empty string. The cognitive burden of a set S_P of passwords in our simulation is approximately $L(S_P)$. There is also a small cost associated with mapping passwords to services in memory. We very roughly approximate this by including an additive factor of 1 for each service that has a corresponding password that is in the agent's memory.

Equipped with an understanding of these two underlying models, we can now look more deeply at the subgoals associated with creating an account for a service and signing in to a service.

During account creation, the agent must first construct a username and password combination. If the agent's cognitive burden is under a specified threshold, the password reuse threshold, she chooses the weakest password she can think of that satisfies the password composition requirements. If, however, her cognitive burden exceeds this password reuse threshold, she attempts to recycle an existing password before considering a new, unique password. The particular password chosen for reuse is determined by the password reuse priority parameter which specifies whether the agent should reuse the longest or shortest viable password. Once an account has been created, the agent may opt to either memorize her password or write it down. This process is again determined by comparing the agent's cognitive burden to a specified threshold, the password write threshold. If the agent's cognitive burden is under the threshold, she will try to memorize the password; else, she will write it down. If the agent opts to memorize password P for service S then the password strength $V_{S,P}$ will be initialized to 1, while if she instead opts to write down the password, $V_{S,P}$ will be initialized to 0.5. And, in both cases, all S-specific password strengths associated with passwords different than the chosen password are set to 0; that is, $V_{S,P'}$ will be set to 0 for $P' \neq P$. Additionally, during account creation the service-specific password forget rate is initialized to a model parameter entitled initial password forget rate. While we've discussed the process of account creation, the same processes largely apply to the password reset process, the primary difference being that the agent will not create a new username.

When an agent wishes to sign in to service S, she first attempts to recall her password for the service. This is done by choosing the password P with greatest S-specific password strength. If $V_{S,P}$ exceeds a parameter called the recall threshold, she attempts to sign in using P. If she cannot recall a password, that is, if there is no password with S-specific password strength that exceeds the recall threshold, she checks to see if she wrote down a password. If she did write down a password she uses that password; else, she resets her password.

As discussed earlier, after each action is performed password strengths are decremented by service-specific password forget rates. These forget rates are initialized to an initial password forget rate during account creation and password resets, and they are changed during sign-in attempts. Whenever the agent enters a password P for a service S and it is accepted, the password forget rate for that service is halved, the password strength $V_{S,P}$ is set to 1, and, for all $S' \neq S$, $V_{S',P}$ is strengthened by the product of the password forget rate for S' and the strengthen scalar, a model parameter. When the agent enters a password P for a service S and it is rejected, $V_{S,P}$ is set to 0. While this model is not faithful to reality (e.g., it does not incorporate the time duration between successive recalls) we, again, believe it serves as a good, simple first approximation.

To assess the risk of password compromise, we consider three attack vectors. The first is a direct attack in which the attacker either exploits a service vulnerability or brute forces the password. This is a function of a direct attack risk scalar and a raw password strength function that maps passwords to strength values. The second is an attack wherein the at-

tacker sees the agent's password written down and uses it to access the agent's account. If the password has been written down the risk for this attack is equal to a model parameter that specifies the stolen password attack risk; else, it is 0. The third attack is an indirect attack in which the attacker, using one of the previously mentioned attacks, discovers the agent's password for another site, and then reuses that password to sign in to the agent's account for the target service. The risk of this attack is equal to one minus the probability of being safe from indirect attacks, where the probability of being safe from indirect attacks is the product of probabilities of being safe from indirect attacks from each individual service. The probability that a service S is safe from an indirect attack stemming from S' is the product of a a model parameter, the reuse attack risk, and the probability that S' is not compromised by one of the two aforementioned attacks. For future discussion, we define the security measure M to be the probability that a service is safe from attacks, averaged over all services.

Services are loosely grouped into four classes based on the complexity of their password composition policies: weak, average, good, and strong. Complexity requirements affect the minimum length, minimum number of lower-case alphabetic characters, minimum number of upper-case alphabetic characters, minimum number of digits, and minimum number of special characters required for a password to be accepted. All member services of a single class use the same process to generate their password composition policies.

Here, we briefly explain the code and primary processes in the simulation. The simulation involves agent-side code that is responsible for choosing and performing agent actions and a world hub that is responsible for carrying out all service processes, keeping world state, and printing statistics. A target service is also passed to the world hub. Printed statistics include the number of accounts that have been created, the number of usernames and passwords each agent has written down, the number of usernames and passwords each agent has memorized, the number of passwords resets each agent has performed, and aggregate security measures M associated with each agent's set of accounts. Additionally, for each agent, the hub prints similar statistics for the target service.

3.3 Results & Sensitivity Analysis

For the purposes of better understanding our model and gleaning valuable insights into the security implications of different password policy settings, we performed a variation of one-factor-at-a-time sensitivity analysis, the results of which we study in this section. Although many parameters are highly interactive, we believe this approach still provides valuable insights.

In Section 3.3.1 we review the parameters and state the fixed values used in the analysis. In Section 3.3.2 we discuss the methodology. In Section 3.3.3 we briefly discuss sources of error. In each section thereafter we study a single parameter.

3.3.1 The Parameters

Below, we specify the fixed value we use for each parameter considered in our sensitivity analysis. We also provide a short description of the parameters.

Initial Password Forget Rate: 0.0025
This parameter specifies the initial password forget rate that is set for a service during account creation and password reset.

Recall Strengthen Scalar: 4
This parameter affects the amount that a password belief is strengthened for one service when the password under consideration is successfully used for another service. Specifically, when an agent successfully signs in to service S with password P, for each $S' \neq S$, the password strength value $V_{S',P}$ is incremented by the product of the recall strengthen scalar and the password forget rate for S'.

Recall Threshold: 0.5
This parameter specifies the threshold over which the agent can recall passwords. When the agent is trying to sign in to a service S, she will consider the password P with highest strength value, $V_{S,P}$ associated with that service. If $V_{S,P}$ exceeds the recall threshold, she will attempt to sign in with password P. Else, she will be unable to recall a password and will instead resort to another action.

Password Reuse Priority: long
This parameter can take on one of two values: short or long. When an agent attempts to reuse an existing password during account creation or password reset for a service, this parameter specifies whether she reuses the shortest or the longest password that satisfies the password composition requirements for the service should there exist a recallable password satisfying the password composition requirements.

Password Reuse Threshold: 40
If an agent creates a new account for a service or resets her password for a service and her cognitive burden exceeds the value of this parameter, she will opt to reuse an existing password.

Password Write Threshold: 60
If an agent's cognitive burden exceeds the value of this parameter after creating an account for a given service or resetting her password for a service she will opt to write down the password instead of attempting to memorize it.

Direct Attack Risk: 0.25
This parameter affects the probability that an account may be compromised directly via a service vulnerability or brute force attack, not a stolen password or reuse attack. It effectively acts as a scalar for the password strength associated with a given service to determine the direct attack risk.

Stolen Password Risk: 0.25
This parameter specifies the probability that the attacker may find the agent's password written down and successfully use it in an attack.

Reuse Attack Risk: 0.25
This parameter specifies the probability that an attacker successfully launches a reuse attack on a service S by exploiting a given direct attack or stolen password attack on another service S'.

Distribution of Services: (6,6,6,6)

This parameter is a vector of four integers that specifies the distribution of services according to the strengths of their password composition policies. The shorthand (W, A, G, S) means that W, A, G, and S services employ weak, average, good, and strong password composition policies respectively.

3.3.2 Methodology

We performed a one-factor-at-a-time sensitivity analysis wherein we decided *a priori* on fixed values for each of the ten parameters specified in Section 3.3.1. We varied each parameter within a constrained, feasible parameter space and recorded the aggregate security M (refer to Section 3.2 for more details on M) for six independent trials for each parameter configuration we considered. Our sensitivity analysis is actually a slight variation of the traditional one-factor-at-a-time approach in that, for the distribution of password composition policies parameter, we performed a series of trials for three different configurations of the cognitive thresholds (i.e., password write threshold and password reuse threshold) to better understand the interplay between the three parameters.

For all but one parameter, we stopped simulations when the agent's minimum per-service password forget rate dropped below 0.0005. The exception occurred during testing of the initial password forget rate parameter, which was performed first. While testing the password forget rate parameter, we stopped simulations when the minimum per-service password forget rate dropped below 0.00025.

After gathering data as described above, we generated plots with error bars corresponding to the standard deviation.

3.3.3 Sources of Error

Computer-based arithmetic accounts for one source of error. While not a true source of error, we do see some peculiarities in our graphs due to the use of a finite set of approximately thirty passwords and the use of a step function for evaluating password strength. Though the password list and password strength evaluation function are in some sense parameters, specifying a feasible solution parameter space for them and varying them accordingly is beyond the scope of this initial paper. Last, we recognize that performing only six trials for each configuration of parameters is a limitation of this paper. We will perform more rigorous analyses in future work.

3.3.4 Initial Password Forget Rate

In Figure 1 we see that increasing the initial password forget rate reduces security. Our belief is that as we increase the initial password forget rate users are more inclined to reset their passwords and write down the newly reset passwords during the process.

3.3.5 Recall Strengthen Scalar

In Figure 2 there seems to be a slight increase in security as we increase the recall strengthen scalar. While this may just be error, this may also be in part due to a reduction in passwords being written down as the value of this parameter increases.

3.3.6 Recall Threshold

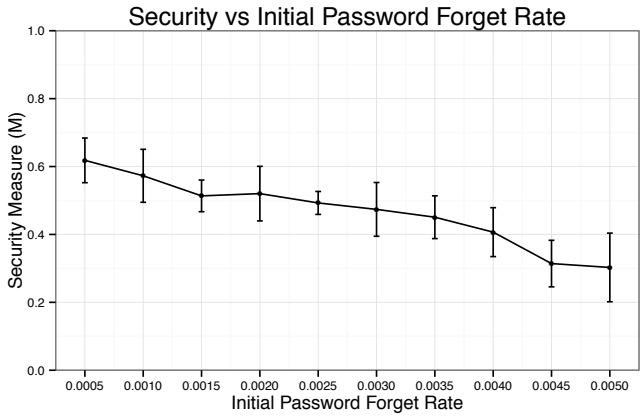

Figure 1

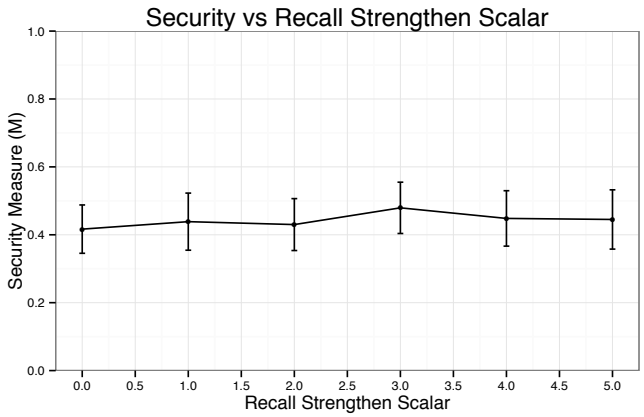

Figure 2

In Figure 3 we see that increasing the recall threshold decreases security. This indeed makes sense. A higher recall threshold means it is more difficult for the user to recall passwords. So, the user will frequently reset her password instead of remembering her passwords. After the user has accrued a large number of accounts, these frequent resets will lead the user to circumvent as a coping mechanism.

3.3.7 Password Reuse Priority

We found that having agents reuse the shortest acceptable password leads to a higher security measure than reusing the longest password. With a short password reuse priority we saw a mean security measure of $M = 0.5222$ with a standard deviation of 0.0616. With a long password reuse priority we saw a mean security measure of $M = 0.4528$ with a standard deviation of 0.0770. One possible explanation for this discrepancy is that a tendency toward reusing shorter passwords reduces the likelihood that a single password is reused across most accounts. That is, since password composition policies vary across services, longer passwords will be more likely to satisfy a greater fraction of password composition policies than shorter ones. Ergo, longer passwords will be more susceptible to reuse attacks.

3.3.8 Password Reuse Threshold

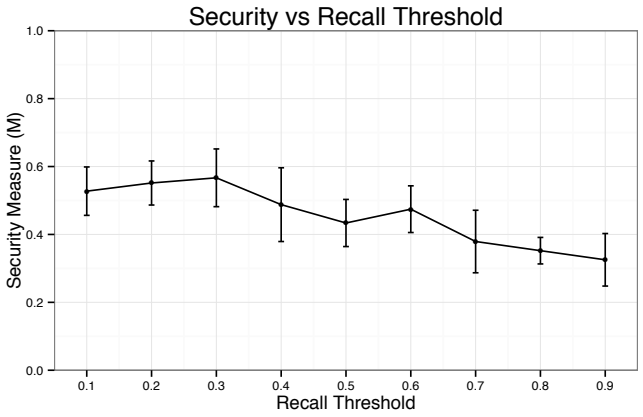

Figure 3

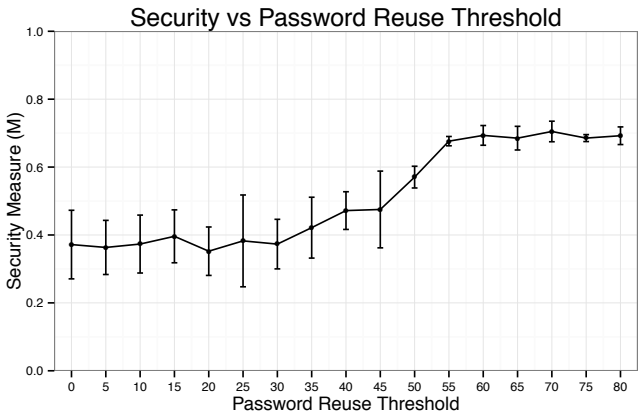

Figure 4

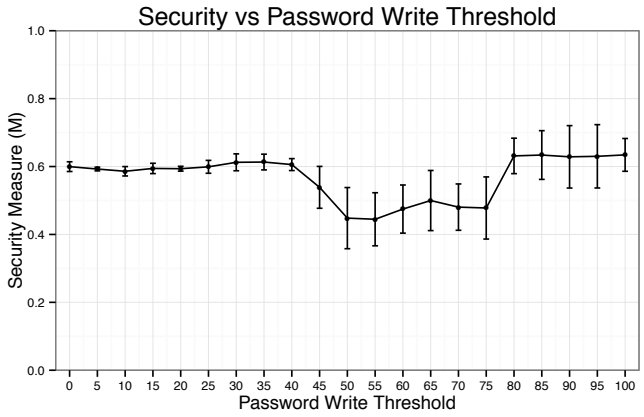

Figure 5

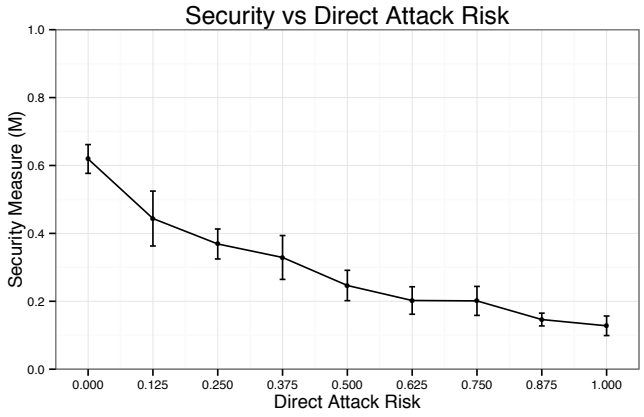

Figure 6

As expected, in Figure 4, increasing the password reuse threshold improves security.

3.3.9 Password Write Threshold

At first glance, Figure 5 may seem a bit surprising. When the password write threshold is very low, M is reasonably high. As we increase the password write threshold, we see a dip in M. And, as we further increase it we see M rise to a value slightly above its value when the password write threshold was very low.

Our rationale for this behavior is as follows. When the password write threshold is low, under 40, users do indeed write down passwords, but writing these passwords down means that the passwords contribute less to the cognitive load of password remembrance; this leads to a larger set of unique passwords at the cost of more passwords being written down, which is a net win as determined by the parameter settings of direct attack risk, reuse attack risk, and stolen password risk that determines M. We see a dip when setting the threshold between 40 and 80 because while users are less inclined to write passwords down during this range, they will be more inclined to reuse passwords as passwords that are not written down contribute a larger cognitive burden. For thresholds over 80, users may still reuse more passwords, but the gains from not writing down passwords finally begins to outweigh gains from not reusing passwords.

3.3.10 Direct Attack Risk

In Figure 6 we see that increasing the direct attack risk value reduces security as expected.

3.3.11 Stolen Password Risk

In Figure 7 we see that increasing the stolen password risk value reduces security roughly as expected. We do see an unusual local maximum for a stolen password risk value of 0.5. We attribute this solely to error because we performed too few trials.

3.3.12 Reuse Attack Risk

In Figure 8 we see that increasing the reuse attack risk value reduces security as expected. We see a peak at 0.625, but we again attribute this to error due to an insufficient number of trials.

3.3.13 Service Distribution

In Figure 9 we look at how changing the number of services while maintaining a fixed percentage of weak, average, good, and strong password composition rules for three pairs

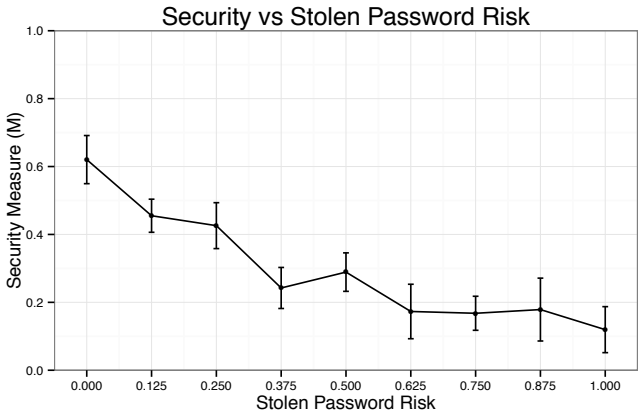

Figure 7

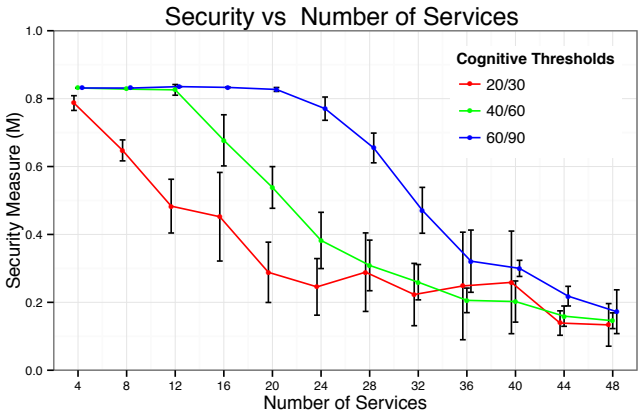

Figure 9

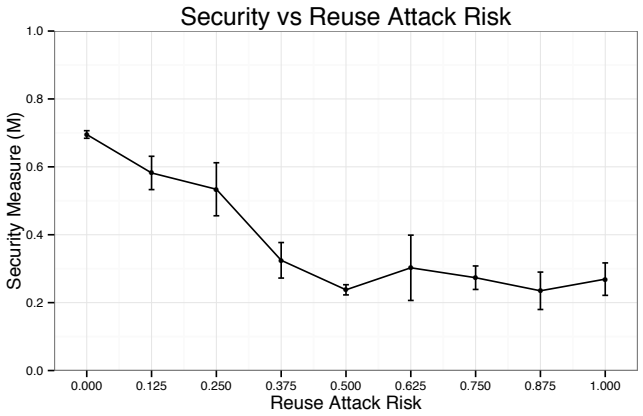

Figure 8

of cognitive threshold settings affects security. Each curve appears to reflect a sigmoid function flipped along the y-axis and shifted accordingly. This is what one might expect. For a small number of services users are able to simply remember their passwords without resorting to circumvention. As the number of services grow users circumvent.

In Figure 10 we use a fixed number, 24, of services and vary the distribution of password composition policies for the same three pairs of cognitive threshold settings. These cognitive threshold pairs correspond to the password reuse and password write thresholds respectively. For the low cognitive threshold pairs, (20/30) and (40/60), circumvention is rampant for most distributions; hence, simply having the most stringent password composition policies tends to make sense because the primary factor in our security measure becomes the raw password strength. For the highest cognitive threshold pair considered, (60/90), there's less circumvention; users may be able to choose a larger set of unique passwords for less stringent distributions, thereby reducing the likelihood of reuse attacks.

We believe further experimentation would demonstrate that even for low cognitive threshold pairs we achieve better security by using weaker distributions under different, but still viable, parameter settings (e.g, changing the password reuse attack risk from 0.25 to 0.5). We leave this for future work.

3.4 Takeaways

While we cannot make specific password policy recommendations based on our model, which requires further validation, we do believe our results provide some valuable insights that serve as indicators of how to improve password policies:

- Always choosing the most stringent password composition policy may be disastrous, endangering both usability *and* security with no gains.
- All circumvention is not the same. To improve security at a given organization, one must pinpoint the threat model and design policies accordingly.
- Endorsing relatively benign circumventions at an organization may reduce the prevalence of particularly malignant circumventions. As an example, It may very well make sense for an organization to give their employees a small card to write their passwords on if security personnel are more worried about a password reuse attack than an adversary stealing passwords from cards.

4. FUTURE WORK

While we feel there's a lot to be done in this space, primary foci for future work include adding to the password management model and building an agent-based model for an autologout scenario.

4.1 Password Management Scenario

We are interested in incorporating more faithful and/or better reasoned models and processes (e.g., [12])) for password recall, cognitive burden, and forgetfulness into our simulation. Once we've done this, we'd also like to revisit the work mentioned in this paper and explore other password management challenges. For a few examples, we'd like to (a) develop a more elaborate password simulation that incorporates communication and password sharing between users, exploring how group dynamics affect circumvention, (b) model how users cope with enterprise requirements requiring them to frequently reset their passwords, or (c) test

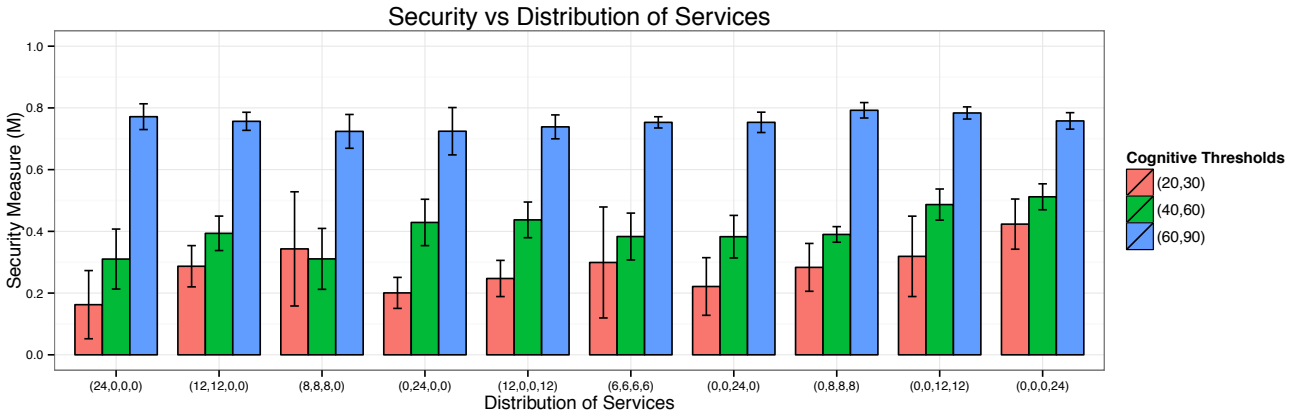

Figure 10

alternative password policies (e.g., what would happen if we allowed users to write passwords on Post-It notes for a limited duration of time, but told them to rip up the Post-It notes afterward?). The idea of recognizing and even incorporating existing circumvention into the security model also seems like an interesting pursuit for modeling work. Last, while we have tried to validate our work with previous studies, this is an ongoing challenge and we would like to pursue new avenues and perhaps devise new experiments to aid on this front.

4.2 The Effect of Security Policies on Group Behavior and The Auto-logout Scenario

Tackling even an ostensibly simple problem, such as setting a "good" timeout threshold, can be a nightmare in practice. On paper, the general shape of a timeout vs security curve seems obvious: surely, it's a monotonically decreasing curve! In practice, where humans act according to flawed belief systems, humans interact with other humans and other systems, work toward achieving many competing goals, and humans are plagued by deficiencies that lead to sub-optimal decision-making, behaviors, and other phenomena, we find the resulting curve can often be counter-intuitive. Indeed, in a recently compiled corpus of circumvention scenarios we've collected, we've observed many examples of such *uncanny descents* where dialing up a security knob worsens aggregate security. [20].

Regarding the challenges of the timeout problem, consider the following anecdote. In a large hospital, clinicians frequently left shared computers logged-in but unattended [5]. Security officers, concerned about inappropriate access and inadvertent modification of patient data, opted to attach proximity sensors to the machines in an effort to mitigate these risks. These sensors detected when users had left terminals logged-in but unattended for some fixed timeout threshold. When such an event was detected, the logged-in user was automatically logged out of the machine. Clinician reception of these proximity sensors startled security officers. Clinicians, annoyed with the system, which was an impediment to doing actual work, placed styrofoam cups over the proximity sensors, which effectively tricked the proximity sensors into believing clinicians were nearby when they were not. The proximity sensors were an absolute failure. Resources had been spent with the goal of improving security, but doing so yielded no security gains; instead, it was utterly defeated and it probably created a greater rift between clinicians and security personnel, making future security challenges even more difficult to address.

This anecdote highlights that it is essential to find solutions that make sense in the context of enterprise workflow– solutions that can be successfully adopted by users while also realizing security objectives without sabotaging other objectives.

So, how do we arrive at these solutions? It is usually impractical for security personnel to test out different security approaches within existing enterprises. Even if it is feasible, doing so often involves, at the minimum, substantial time, implementation costs, maintenance costs, and depletion of a finite user compliance budget [21]. We contend that multi-agent simulations may help distinguish good solutions from bad ones by predicting stress points of candidate implementations, thereby suggesting foci to improve upon. However, we are not suggesting that agent-based modeling is some magical panacea that can be used to address all security problems. It has its limits; it is nigh impossible to predict the unprecedented. Instead of trying to predict inventive workarounds such as the placement of styrofoam cups over proximity sensors, we aim to gauge user inclination to circumvent.

We can estimate the risk from user circumvention in terms of motive, opportunity and potential harm. First, motive stems from the frequency of workers leaving and returning to shared workstations, where the time taken to log in becomes a significant drain when summed over many instances. Second, opportunity also arises from the shared environment, where workers might remain logged in to avoid these costs, or use another's credentials, inadvertently or not. Third, the potential harm comes from the nature of the task, since medication prescriptions or notes of delivery may then be ascribed to the wrong patient.

Using simulation, we can explore the relevant factors that affect security risks associated with a clinician using a terminal to which another clinician is logged in. The likelihood of risk is affected by the number of agents, the number of workstations, group attitudes towards security and circumvention, and the dynamic nature of tasks; the actual risk

is affected by the kinds of tasks performed. Simulations allow us to compare how burdensome different kinds of solutions are on users. For example, we might compare an auto-logout solution to a solution involving authentication challenges after a period of inactivity, which may slightly reduce the burden of having to log back in to a service; or, we could detect tasks that are disparate from the current task and warn the user that they may be using a terminal to which someone else is logged in. For some tasks, it may be possible to predict whether the worker must return to complete her session, and to apply different policies based on this prediction.

Last, while we mentioned the timeout problem in the medical setting, there are numerous other scenarios where auto-logouts may be relevant. And, we believe modeling approaches could be developed for them as well.

5. CONCLUSION

We have discussed our work toward building an agent-based model for a password management scenario. While validation is a challenge, we have made first steps toward building a useful cognitive behavioral agent-based model for password circumventions; we've also performed trials that have generated what we believe to be interesting and perhaps even counter-intuitive results. For example, under certain assumptions, making password composition requirements more stringent may actually lead to a decrease in aggregate security. As another example, allowing users to write down passwords may actually improve security by reducing the likelihood of password reuse and reuse attacks. Password management is just one of many areas where we believe cognitive behavioral agent-based models can serve as a useful tool. In Smith et al [20], we observed that a pattern of policy choices at one site counter-intuitively affects security at other sites. Applying agent-based modeling to these sorts of scenarios and others, such as those mentioned in 4 may provide useful insights that are otherwise difficult to attain.

Acknowledgment

This material is based upon work supported by the Maryland Procurement Office under Contract No. H98230-14-C-0141.

6. REFERENCES

[1] Jim Blythe and L Jean Camp. Implementing mental models. In *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on*, pages 86–90. IEEE, 2012.

[2] Jim Blythe. A dual-process cognitive model for testing resilient control systems. In *Resilient Control Systems (ISRCS), 2012 5th International Symposium on*, pages 8–12. IEEE, 2012.

[3] Jim Blythe, Ross Koppel, and Sean W Smith. Circumvention of security: Good users do bad things. *Security & Privacy, IEEE*, 11(5):80–83, 2013.

[4] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.

[5] Vijay Kothari, Jim Blythe, Sean Smith, and Ross Koppel. Agent-based modeling of user circumvention of security. In *Proceedings of the 1st International Workshop on Agents and CyberSecurity*, page 5. ACM, 2014.

[6] Rachna Dhamija, J Doug Tygar, and Marti Hearst. Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 581–590. ACM, 2006.

[7] Michael Bratman. Intention, plans, and practical reason. 1987.

[8] PN Johnson-Laird. Mental models: towards a cognitive science of language, inference, and consciousness. 1986.

[9] Keith E Stanovich and Richard F West. Advancing the rationality debate. *Behavioral and brain sciences*, 23(05):701–717, 2000.

[10] John E Laird, Allen Newell, and Paul S Rosenbloom. Soar: An architecture for general intelligence. *Artificial intelligence*, 33(1):1–64, 1987.

[11] John R Anderson. Problem solving and learning. *American Psychologist*, 48(1):35, 1993.

[12] Dinei Florêncio, Cormac Herley, and Paul C Van Oorschot. Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts. In *Proc. USENIX Security*, 2014.

[13] Shirley Gaw and Edward W Felten. Password management strategies for online accounts. In *Proceedings of the second symposium on Usable privacy and security*, pages 44–55. ACM, 2006.

[14] Blake Ives, Kenneth R Walsh, and Helmut Schneider. The domino effect of password reuse. *Communications of the ACM*, 47(4):75–78, 2004.

[15] Joseph Bonneau, Cormac Herley, Paul C Van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 553–567. IEEE, 2012.

[16] Richard Shay, Abhilasha Bhargav-Spantzel, and Elisa Bertino. Password policy simulation and analysis. In *Proceedings of the 2007 ACM workshop on Digital identity management*, pages 1–10. ACM, 2007.

[17] Yee-Yin Choong. A cognitive-behavioral framework of user password management lifecycle. In *Human Aspects of Information Security, Privacy, and Trust*, pages 127–137. Springer, 2014.

[18] Karen Renaud and Lewis Mackenzie. Simpass: Quantifying the impact of password behaviours and policy directives on an organisation's systems. *Journal of Artificial Societies and Social Simulation*, 16(3):3, 2013.

[19] Levenshtein distance - Rosetta Code. `http://rosettacode.org/wiki/Levenshtein_distance`.

[20] Sean W Smith, Ross Koppel, Jim Blythe, and Vijay Kothari. Mismorphism: a Semiotic Model of Computer Security Circumvention. Technical report, Dartmouth College, Department of Computer Science, 03 2015.

[21] Adam Beautement, M Angela Sasse, and Mike Wonham. The compliance budget: managing security behaviour in organisations. In *Proceedings of the 2008 workshop on New security paradigms*, pages 47–58. ACM, 2009.