DIVINA: Discovering Vulnerabilities of Internet Accounts

Ziad Ismail, Danai Symeonidou, Fabian Suchanek
Télécom ParisTech, Paris, France
{ismail.ziad, danai.symeonidou, fabian.suchanek}@telecom-paristech.fr

ABSTRACT

Internet users typically have several online accounts – such as mail accounts, cloud storage accounts, or social media accounts. The security of these accounts is often intricately linked: The password of one account can be reset by sending an email to another account; the data of one account can be backed up on another account; one account can only be accessed by two-factor authentication through a second account; and so forth. This poses three challenges: First, if a user loses one or several of his passwords, can he still access his data? Second, how many passwords does an attacker need in order to access the data? And finally, how many passwords does an attacker need in order to irreversibly delete the user's data? In this paper, we model the dependencies of online accounts in order to help the user discover security weaknesses. We have implemented our system and invite users to try it out on their real accounts.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Security, Safety, Online Accounts, Vulnerabilities, Rules

1. INTRODUCTION

In 2012, Mat Honan found that his Twitter account had been hacked. When trying to figure out what happened, he found that not just his Twitter account was hacked, but also his Amazon account, his Gmail account, and his Apple account. His iPhone was locked, and his Mac was wiped clean. All the data was erased. As it turned out, hackers were able to access his Amazon account. With the credit card numbers stored there, they could access his Apple account. Since his Gmail account sent password recoveries to his Apple account, the hackers could access Gmail – and thus all of Mat's digital accounts [2].

If you want to minimize the chance of getting hacked, you can enable *two-factor authentication* for an account, if it is available. With two-factor authentication, you have to provide not only your password if you want to log in, but also a second security token. This can be a code that you receive by SMS, a number generated by an app on your smartphone, or a special code that you possess in a printed version. This would have made the attack on Mat impossible.

The problem is, though, that if you forget your password and you do not have access to your second security token, you lose access to your account. Owen Williams, for example, woke up one day to find that his Mac was locked. Apparently, hackers had tried to access his Apple account. They failed, but for security reasons Apple locked the account. To unlock it, Owen had to provide the printed recovery key. It's just that he never printed it! Thus, he was not able to access his Apple account any more, losing access to all the data stored on Apple devices [5]. Hence, if you lose your recovery keys, you may lose access to your data. Therefore, we find ourselves in a trade-off: The more protection we add to an account, the less vulnerable it is to hackers. However, by increasing the protection, we also increase the risk of losing the access ourselves.

To make matters worse, security measures are often intertwined. Assume, e.g., that your Gmail password can be recovered from your Apple account, and that you have two-factor authentication enabled on your Gmail account. This gives you some degree of security. However, if you send your Gmail recovery codes to your Apple email address and a hacker manages to gain access to your Apple account, then this security implodes. In general, the more accounts you have, the more difficult it is to keep track of the security dependencies between them. In Britain, for example, the average adult has 26 different online accounts [1]. However, they use an average of just five different passwords. As a result, a hacker could compromise one account and leverage it to get into the others.

There are a number of tools to manage account passwords, such as 1Password, LastPass, and Personal. However, none of these analyzes the dependencies between online accounts, or the potential impact when a set of access keys is compromised. In fact, there seems to be no tool that allows users to check the dependencies of their accounts automatically [3].

In this paper, we use Datalog rules to model the security dependencies between online accounts. We present DIVINA, a system that allows users to assess the transitive impact of compromising a set of access keys. Using game-theory, DIVINA also suggests the security guarantee that the user has to give for each of his access keys, in order to bound the probability that one of his accounts will be hacked. Our

This research was partially supported by Labex DigiCosme (project ANR- 11-LABEX-0045-DIGICOSME) operated by ANR as part of the program "Investissement d'Avenir" Idex Paris-Saclay (ANR-11-IDEX-0003-02).

WWW 2015 Companion, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3473-0/15/05.
http://dx.doi.org/10.1145/2740908.2742836.

system can be tried out with the dependencies of real online service providers. It allows users to assess the security risks of their current configuration, and to simulate alternatives.

2. MODELING ACCESS

Our data is protected by *keys*. A key is any item that is necessary to access the data – such as a physical key, a password, a smartphone, or the physical possession of a digital storage device or a physical document. For every online account, we want to ensure 3 things:

Data Safety: We can still access the data as long as we lose less than n keys.

Data Security: An attacker needs at least n keys to access the data.

Data Protection: An attacker cannot erase the data as long as he has less than n keys.

Here, n is an integer number that defines the level of safety, security and protection, respectively. For example, if an account has a security level of 3, then an attacker needs at least 3 keys in order to gain access to the data.

To see that these desiderata are orthogonal, consider a user who can access her Gmail account using only her password. If she loses her password, she can reset it through 3 alternative email accounts (with different passwords). In this scenario, the level of data security is 1, since an attacker needs to know only one password to access the account. The level of the data safety is 4, because the user has to forget 4 passwords to lose access to her data. The level of protection is 1, because an attacker can delete all emails by knowing only one of the passwords. If the user backs up her mails on her computer, then the level of protection increases to 2: In order to destroy the data, the attacker has to be able to log in to Gmail *and* to steal the user's computer.

In general, data can be secured without being protected. For example, an encrypted hard drive is secure at level 2 (an attacker has to steal the hard drive and to decrypt it to access the data), but protected only at level 1 (the attacker can just steal and destroy the hard drive). Vice versa, an account can be better protected that secured, if, e.g., the service requires only one password to read the data, but two passwords to modify it. Dropbox, e.g., distinguishes between sharing a folder (which requires a full login and allows writing), and sharing the link of a folder (which requires only a URL, and allows only reading).

3. APPROACH

3.1 Datalog

We propose to model the dependencies between accounts as *datalog rules*. A datalog rule r over a set of atoms $\mathcal{A}$ is a rule of the form $a_1, a_2, ..., a_n \rightarrow a_0$. Here, $a_1, ..., a_n \in \mathcal{A}$ are the *antecedent* of r, and $a_0 \in \mathcal{A}$ is the *succedent* of r. Intuitively, the rule says that the succedent is true if all atoms in the antecedent are true. We consider only null-ary atoms, i.e., the elements of $\mathcal{A}$ are simple propositions such as *hasGmailAccess* without arguments. Then, we say that r *derives* a_0 from a set of atoms $A \subseteq \mathcal{A}$, written $A \vdash_r a_0$, if $A \supseteq \{a_1, ..., a_n\}$. Given a set of rules $\mathcal{R}$, and a set of atoms $A \subseteq \mathcal{A}$, we say that an atom $a \in \mathcal{A}$ can be derived, written $A \vdash_R^* a$, if (1) $a \in A$ or (2) there exists an atom $a' \in \mathcal{A}$ and a rule $r \in \mathcal{R}$, such that $A \vdash_r a'$ and $A \cup \{a'\} \vdash_R^* a$. If the set of rules is fixed, we will just write $A \vdash a$ for $A \vdash_R^* a$.

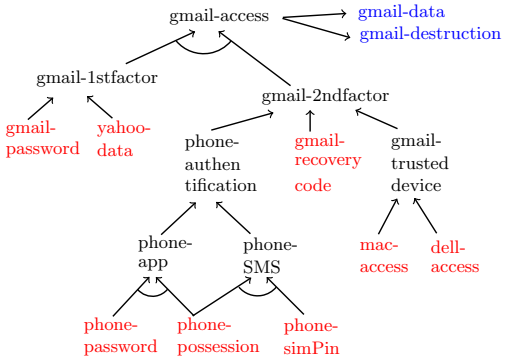

Figure 1: Dependencies of a Gmail account

3.2 Atoms

In our scenario, the atoms represent stages of a hacking process that the attacker has reached. We consider the accounts of only one user. Therefore, our atoms are all of the form *account-stage*. Here, *account* is an online account (such as Gmail, Amazon, or Dropbox). For uniformity, we also consider physical devices (such as a phone, a computer, or a hard drive) accounts. *stage* is a stage that the hacker has reached for this particular account. Important stages for an account x are:

***x-password*:** knowing the password of the account.

***x-2ndfactor*:** being able to overcome the second factor of the two-factor authentication of the account.

***x-possession*:** having physical possession of the item (e.g., of a phone, computer, or a hard drive).

***x-access*:** having access to an account (e.g., possessing a phone, and knowing the code to unlock it; possessing a hard drive, and knowing the code to decrypt it).

***x-data*:** having access to the data stored in x.

***x-destruction*:** being able to destroy the data stored in x.

Other atoms can be created ad libitum.

3.3 Rules

Based on the atoms, we can define rules. Typical rules are:

gmail-password, gmail-2ndfactor $\rightarrow$ *gmail-access*
phone-possession, phone-password $\rightarrow$ *gmail-2ndfactor*
gmail-recoverycode $\rightarrow$ *gmail-2ndfactor*

The first rule says that if I have the Gmail password and I am able to overcome the second factor of the two-factor authentication, then I have access to Gmail. The second rule says that the second factor can be overcome by possessing the phone and being able to unlock it. The third rule says that the second factor can also be overcome by using a recovery code. We see that the first rule defines a conjunction: Two stages are needed to access the Gmail account. The other two rules define a disjunction: Either of the methods can overcome the second factor. Figure 1 shows the real dependencies of a Gmail account, assuming that the user has a password backup address at Yahoo, that there are 2 trusted devices on which the second factor is not required, that the phone is locked by a code, and that the SIM card is also locked by a code. Arcs between arrows state that both previous stages are needed. As we can see, the dependencies are already quite intricate for only one account.

3.4 Safety, Security, Protection

In all of the following, we assume a given set of rules $\mathcal{R}$ over a fixed set of atoms $\mathcal{A}$. The atoms that we particularly care about for an account x are *x-data* and *x-destruction* (in blue in Figure 1). Access to these atoms defines the level of safety and security (for the first), and protection (for the second). Any atom that is not the succedent of a rule becomes a *key*: $\mathcal{K} = \mathcal{A} \setminus \{a_0 : (a_1, ..., a_n \rightarrow a_0) \in \mathcal{R}\}$. Keys are usually possessions of physical items (such as *phone-possession*), or knowledge of codes or passwords (such as *gmail-recoverycode*). They are in red in Figure 1. When we add more security mechanisms, a key may cease to be a key. For example, if the recovery code in the figure is not just in a safe place, but also encrypted, then we could add the rule:

gmail-recoverycodeAccess, gmail-recoverycodeDecryption $\rightarrow$ *gmail-recoverycode*

Then, *gmail-recoverycode* ceases to be a key, and the stages in the antecedent become a key. Now, safety, security, and protection can be defined in terms of rules:

Data Safety: An account x is safe at level n, if:

$$\forall K \subseteq \mathcal{K}, |K| < n : \exists K' \subseteq \mathcal{K}, K' \cap K = \emptyset : K' \vdash x\text{-}data$$

Data Security: An account x is secure at level n, if:

$$\forall K \subseteq \mathcal{K}, |K| < n : K \nvdash x\text{-}data$$

Data Protection: An account x is protected at level n, if:

$$\forall K \subseteq \mathcal{K}, |K| < n : K \nvdash x\text{-}destruction$$

3.5 Optimal Security Strategy

Some keys are more important than others. Given the dependencies between the user's accounts, we would like to identify the set of keys that can cause the most impact once compromised and propose a strategy that guarantees the security of these accounts. We use game theory as a mathematical framework for this purpose.

In most cases, the attacker chooses his attack strategy based on the deployed security mechanisms. Therefore, we model the interactions between the user and the attacker as a Stackelberg game [4]. In this type of games, the leader chooses his strategy first. Then, the follower, who knows the leader's choice, chooses his strategy. In our model, the leader is the user, who tries to find the optimal protection level for his keys. The follower is the attacker, who tries to compromise the user's keys. We define the utilities U_a and U_u of the attacker and the user, respectively, as follows:

$$U_a(p,q) = \sum_i \sum_{j \in a(i)} \{p_i^j \prod_{k \in j} q^k W_i - p_i^j C_a^j\}$$

$$U_u(p,q) = -\sum_i \sum_{j \in a(i)} \{p_i^j \prod_{k \in j} q^k W_i + (1-q^k) C_u^k\}$$

Here, i refers to an account, and $a(i)$ is the set of different key combinations that can access i. For example, if Gmail can be accessed either by the Gmail password and the phone, or by the phone and the Yahoo account (which can reset the Gmail password), then $a(gmail) = \{\{$*gmail-password, phone-possession*$\}, \{$*phone-possession, yahoo-password*$\}\}$. j is one combination of keys in $a(i)$, and k refers to one key in j. p_i^j is the probability that the attacker attempts to compromise account i by the set of keys j. q^k is the probability that key k can be compromised. W_i refers to the value of the data in account i. C_a^j and C_u^k refer to the cost of attacking the set of access keys j, and the cost of protecting an access key k, respectively.

This game encodes a balance, where the user chooses to protect certain keys k (at a cost C_u^k) in order to keep the value W_i of account i, and the attacker chooses to obtain certain keys k (at a cost C_a^k) in order to obtain the value W_i. We are interested in finding a Nash Equilibrium (NE) of that game, i.e., a state in which none of the players has an incentive to deviate unilaterally. We solve the Stackelberg game by backward induction. Since the user's utility is a decreasing function w.r.t. p_i^j, the user has an incentive to choose a strategy that reduces this value. From the attacker's utility function, we notice that he will not attempt to compromise a set of keys $j \in a(i)$, if $\prod_{k \in j} q^k W_i - C_a^j = \epsilon$, where ϵ is a small negative number. Therefore, in this game, the NE has the following property: For each account i, and for each set of keys $j \in a(i)$, the probability that the attacker will compromise this set is bounded by C_a^j / W_i. This can be interpreted as follows: Let us assume that the value of the data W_i is constant for all i. Fix, for every account i, the *desired* probability p_i that i will be hacked during the next year. This probability will usually be low, say 1%. Choose $C_a^j = p_i W_i$. Then, an equilibrium state q of the game can be interpreted as follows: The user has to take care that the probability of compromising key k within one year does not exceed q^k. We say that the user has to give a *security guarantee* of $1 - q^k$ to key k. In this case, the attacker will not have an incentive to conduct an attack to compromise the user's keys. Technically, the probability that an account i will be hacked in the coming year will be bounded by the desired value p_i. Thus, the Nash Equilibrium of the game will tell the user which security guarantees $1 - q^k$ he has to give for his keys k, in order to get a security assurance p_i for his account i.

Interpreting the probabilistic security guarantees is not always straightforward. However, there are a number of tools and statistics that can help. For example, burglary statistics can help estimate the probability of theft, and there are tools that can estimate how long it takes to guess a password by brute force. Otherwise, common sense is a good indicator: if the user has to give a security guarantee of 95% to his password, then his password should be rather long.

4. DIVINA

4.1 Rules

In order to analyze the safety, security, and protection of online accounts, we have to identify the dependencies that exist between the different accounts, and the security mechanisms that they offer. To this end, we have studied real-world services and modeled their dependencies as rules. We have modeled Gmail, Dropbox, Amazon, Facebook, and Apple. In addition, we have modeled real-world dependencies, such as the code needed to unlock the phone, the impact of backups, and the effect of using the same password on several services. To tailor these rules to a particular user, we have developed an online questionnaire that collects information about the user's accounts, links between his accounts, and the security measures that he has deployed on each of them. Based on this input, we compute rules that are specific to the user scenario.

4.2 Algorithm

Given a set of rules $\mathcal{R}$, we can deduce the keys $\mathcal{K}$. Then we use a simple algorithm to compute the levels of safety, security, and protection: For security, we try out all possible combinations of keys $K \subset \mathcal{K}$, and see if we can derive *x-data*

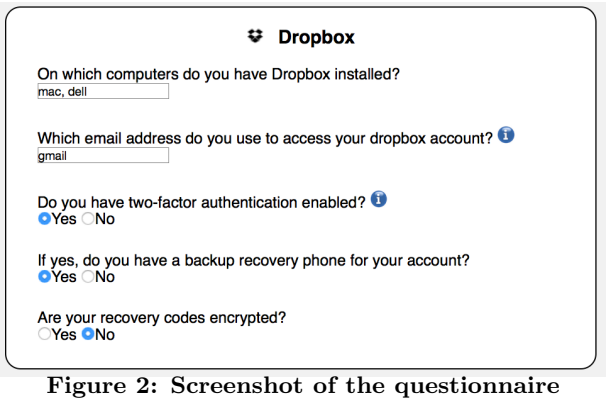

Figure 2: Screenshot of the questionnaire

for an account x. If we cannot, then x has a level of security larger than $|K|$. Similarly, for protection, we try out all possible combinations of keys $K \subset \mathcal{K}$, and see if we can derive x-*destruction*. If we cannot, then x has a level of protection that is larger than $|K|$. For safety, we try out all combinations $K \subset \mathcal{K}$, and see if we can derive x-*data* from $\mathcal{K} \setminus K$. If we can, then x is safe at level $|K|$.

The complexity of this algorithm may seem prohibitively high. Yet, it is not. We first observe that if x is not safe at level n, then it cannot be safe at any level $n' > n$. The same goes for security and protection. Thus, we can gradually increase n and stop investigating a desideratum as soon as it is violated. By definition, n also has a natural upper bound $|\mathcal{K}|$. If k is the maximal level that we are interested in, then the complexity of trying out all possible sets of keys K until $|K| \leq k$ (with $k < \frac{1}{2}|\mathcal{K}|$) is:

$$\sum_{i=1}^{k} \binom{|\mathcal{K}|}{i} < k \times \binom{|\mathcal{K}|}{k} \leq \frac{|\mathcal{K}|^k}{(k-1)!}$$

For a constant k, this expression is polynomial in the total number of keys. At each step, we have to derive all derivable atoms. These are at most $|\mathcal{A}|$. Thus, the overall complexity of the algorithm is $O(|\mathcal{K}|^k \times |\mathcal{A}|)$. In practice, k is very small. In our analysis of real-world accounts, we found no account that is secure beyond level 3. Also, people rarely have more than 4 distinct backups of a piece of data, implying that protection does not go beyond level 4. This means that, while further optimizations can clearly be envisaged, the complexity of our algorithm is acceptable.

Our system, DIVINA, is implemented in JavaScript and included in the online questionnaire. This way, the entire system runs in a Web page in a Web browser on the client side. This has an impact on performance, but overall response times are, at a few seconds, still very reasonable.

5. DEMO

The goal of our demo is to help users assess the vulnerabilities of their online accounts. This works as follows. The user first fills out our online questionnaire (Figure 2). Our questions mirror the security policies deployed by the respective services. While the questions of the form are predefined, the user can use any account names in the fields. The user can also add any user-specific rules. These can be rules for accounts that we have not yet foreseen, or rules for physical dependencies (such as physical keys, safes, and offices). Overall, the quality of the result depends on the correctness of rules provided by the user.

After this stage, the system shows the computed rules. This allows the user to understand the dependencies between his accounts. Finally, a click on the *compute button* will make the system compute the level of safety, security, and protection for each account, together with the key sets K. For example, the user could find that one of his encrypted computers is not protected beyond level 1, because it is linked to a Dropbox account that is also linked to an unencrypted laptop. Thus, access to that unencrypted laptop will allow an attacker to erase the Dropbox and thus the data on the first computer.

The system also computes the required security guarantees for each key. This allows the user to know which keys are particularly precious. The values may prompt the user to think differently about where he keeps his phone, or which passwords he chooses.

If the user is not satisfied with the current levels of safety, security, and protection, he can simulate different improvement strategies by modifying his answers to the questions. For example, if the safety of an account is too low, the user could add a backup email address.

The system runs completely locally and stateless. At no point does the form ask for passwords or personal information. No attempt is made to contact the service providers. No data leaves the browser, and no data is stored in the system once the browser is closed. Thus, users can simulate the vulnerabilities of their accounts without any security risk.

6. CONCLUSION

In this paper, we have defined the concepts of data safety, data security, and data protection for online accounts. We have proposed to model the dependencies between accounts by Datalog rules. We have also shown how to compute the importance of keys by modeling the dependencies as a game-theoretical problem. Our system, DIVINA, allows users to spot the weak points in their current account configuration, and to simulate improvements. Our demo can be tried out online[1]. With DIVINA, we hope to raise awareness for the vulnerabilities of online accounts, and to help users improve the safety, security, and protection of their data.

7. REFERENCES

[1] E. CreditExpert. Online ID OD: illegal web trade in personal information soars. 2012. `http://press.experian.com/United-Kingdom/Press-Release/illegal%20web%20trade%20in%20personal%20information%20soars.aspx?&p=1`.

[2] M. Honan. How Apple and Amazon Security Flaws Led to My Epic Hacking. Wired, 2012-08-06. `http://www.wired.com/2012/08/apple-amazon-mat-honan-hacking/all/`.

[3] M. Kelly. Your weakest link: All those online accounts you've forgotten about. VentureBeat, 2013-03-05. `http://venturebeat.com/2013/03/05/online-accounts-security/`.

[4] M. J. Osborne and A. Rubinstein. *A course in game theory*. MIT Press, 1994.

[5] O. Williams. The dark side of Apple's two-factor authentication. The Next Web, 2014-12-08. http://thenextweb.com/apple/2014/12/08/lost-apple-id-learnt-hard-way-careful-two-factor-authentication/.

[1] `http://suchanek.name/programs/divina`