Using a Policy Spaces Auditor to Check for Temporal Inconsistencies in Healthcare Audit Log Files

Tyrone Grandison
Proficiency Laboratory International, Ashland, OR, USA, tgrandison@proficicenylabs.com

Sean Thorpe
University of Technology Jamaica, Kingston, Jamaica, sthorpe@utech.edu.jm

ABSTRACT

The core tenet of the healthcare field is that care delivery comes first and nothing should interfere with it. Consequently, the access control mechanisms, used in healthcare to regulate and restrict the disclosure of data, are often bypassed, especially in emergency cases. This concept is called 'break the glass' (BtG) phenomenon and is common in healthcare organizations. Though useful and necessary in emergency situations, from a security perspective, it is an important system flaw. Malicious users can exploit the system by breaking the glass to gain unauthorized privileges and accesses. Also, as the proportion of system accesses that are BtG increases, it becomes easier for an attacker to hide in the crowd of the audit log. In this paper, we build upon existing work that defined policy spaces to help manage the impact of the break the glass phenomenon in healthcare systems. We present a system that enables the inference and discovery of facts that require further scrutiny. This significantly reduces the burden on the person investigating potentially suspicious activity in the audit logs of healthcare information systems.

Keywords: Healthcare, Security, Exception Handling

1. INTRODUCTION

The nature of healthcare data, and the decisions based on it, makes it vitally important that it be accessible to the medical practitioners that need to deliver the best possible care to their patient(s). This requirement highlights an important imperative for healthcare systems – "nothing interferes with the delivery of care" (Grandison & Davis, 2007). Intuitively, this essential primitive is understandable; as the possible outcomes of having (or not having) access to this data include continued life or death.

Access control (AC) systems are the foundational mechanisms that healthcare systems use to protect medical data. Contemporary access control models and policies assume that the access requests, which have to be complied with, are known in advance and can be stipulated using authorizations. Unfortunately, it has been shown that access control restrictions are often bypassed in emergency situations (Rostad and Edsberg, 2006; Bhatti and Grandison, 2007), especially when the patient's life is at risk. For instance, in an emergency, the on-duty nurse may require (and should be granted) access to data that under "normal" circumstances he or she cannot view. This phenomenon is usually referred to as "break the glass" (BtG). While useful and mandatory in the delivery of care, the break the glass concept and mechanism can represent a weakness for the security of the system, since allowing it in an unconditional or uncontrolled manner can easily open the door to abuses (Bhatti and Grandison, 2007).

To limit (or prevent) such exploits, the AC system should minimize the cases in which no regulation applies and the break the glass principle is enforced (Bhatti and Grandison, 2007). An AC system designed to operate in the healthcare scenario should also be flexible and extensible (i.e. it should not be limited to a particular model or language), should protect the privacy of the patients, and should not allow exchange of identity data, in compliance with legislation.

Prior work (Bhatti and Grandison, 2007; Ardagna et al., 2008) analyzed the audit logs from healthcare information systems, ascertained as many policy rules as possible from the logs that should be included in the access control policy and then transfer them. This increases the number of cases that the access control mechanism covers, i.e. increases the policy coverage (Bhatti and Grandison, 2007).

Ardagna et al. (2008, 2010) introduced an exception-based access control solution whose main goal was to better control the break the glass attempts in healthcare systems, and to reduce possible breaches in the patients' privacy. They defined the concept of policy spaces, which balance the rigorous nature of traditional access control systems with healthcare's "prime directive"[1]. We briefly describe policy spaces in section 2, present the policy evaluation workflow in section 3, introduce our system for utilizing policy spaces and an inference system to help with the discovery of insight (section 4) and conclude in section 5.

2. POLICY SPACES

Ardagna et al. (2008) define a policy space as a policy repository, whose policies regulate access to resources. Space P^+ represents authorized accesses and regulates common practice requests. A request that satisfies a policy in P^+ is permitted, while space E^U represents unplanned exceptions and regulates all those requests for which policies in P^+ are not applicable.

As nothing should interfere with the delivery of care, space P^+ may be bypassed, especially when a patient's life is in danger. In these emergency situations, although the requester does not have the authorization to perform the action requested (i.e. no policy in P^+ applies), the request is always permitted by the policies in E^U, thus breaking the glass. As stated previously, this makes the system vulnerable to malicious users that may leverage the BtG principle to breach the patient's privacy when it is not strictly necessary.

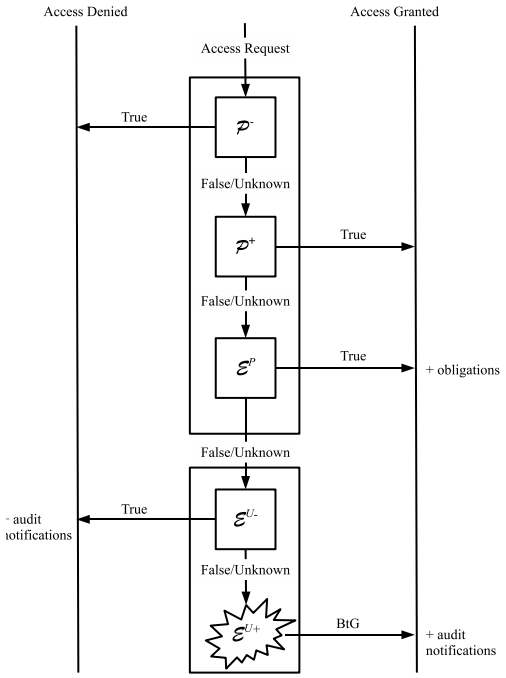
To limit the possible abuses exploiting the BtG option, Ardagna et al. (2008) proposed the idea of defining a solution based on the following set of policy spaces:

- Authorized Accesses (P^+). Space P^+ corresponds to traditional access control policies. Intuitively, P+ includes positive authorizations regulating 'common practice'.
- Denied Accesses (P^-). Space P^- corresponds to access control policies that are used to prevent abuses. Policies in this space are meant to limit exceptions that can result in unauthorized accesses exploiting the BtG option. As a consequence, they must be strictly enforced and do not allow any exception. These policies reflect actions that cannot help even in emergency situations, but can only cause privacy breaches and must be avoided. They can be specified a priori to eliminate accesses that should never be authorized (i.e. accesses that should not be bypassed by BtG) and/or inserted a posteriori because of observed abuses.
- Planned Exceptions (E^P). Space E^P corresponds to policies regulating access requests that do not fall into the normal routine, as well as activities that should not be normally allowed. Policies in E^P are associated with, and indexed by, conditions on the context information represented by attributes in exception space E and on dynamic information in the profiles (e.g., status of the patient), which are used to restrict their applicability. Policies in E^P cannot override policies in P^-. Policies in E^P regulate exceptions that can be foreseen, for example, according to past observations.
- Unplanned Exceptions (E^U). Space E^U corresponds to policies regulating all access requests not covered by the previous policy spaces (P^+, P^-, and E^P). Space E^U is composed of two sub-spaces, denoted E^{U+} and E^{U-}, respectively. The applicability of the policies in these two subspaces strictly depends on the state of the system (i.e., attributes in E) and on dynamic information in the profiles. Specifically, E^{U-} enforces the deny-all default policy and is applicable to all requests that happen in non-emergency cases, when the enforcement of the BtG principle would be an abuse. Space E^{U+} enforces the permit-all default policy and is applicable to all requests that happen in emergency situations, thus allowing all accesses not explicitly allowed or denied by policies in other spaces. All the accesses falling in E^U are inserted into an auditing log for a posteriori analysis.

[1] Each industry or sector has at least one axiom that must be adhered to by any system or subsystem, computerized or not, that is involved in the production of its main deliverable. This axiom is referred to as the *Prime Directive* for that industry.

An important characteristic of these spaces is that they are not limited to a particular access control model, language, or implementation. The auditing process can show access requests that: i) correspond to common practice and should be explicitly permitted by appropriate policies in P^{+}; ii) should never be admitted and should be explicitly denied by defining appropriate policies in P^{-}; iii) are frequent but not common and should be captured by appropriate exceptions in E^{P}. The following section presents the basic flow of policy evaluation in healthcare.

3. POLICY EVALUATION



4.

Figure 1: Policy Evaluation Flow. Source: (Ardagna et al., 2008; Ardagna et al., 2010)

Access requests are of the form ⟨*user-id, action, object, purposes, timestamp*⟩, where *user-id* is the identifier characterizing the requester, *action* is the action that is being requested, *object* is the object on which the requester wishes to perform the action, *purposes* is the purpose (or set thereof) for which the access is requested, and *timestamp* is the time the request is made.

It is assumed that the personal information of patients is collected for a given purpose (e.g., providing patient care). In normal scenarios, data cannot be used for any other purpose without the specific informed consent of the patient it concerns, while in exception scenarios, restrictions to the purpose can be expressed in environmental condition parameters, hereafter called *envcond*, and used to evaluate the applicability of the policies. The purpose of a request is also stored in log files, to possibly identify fraudulent use of data and take adequate countermeasures.

When an access request is received, the sets of applicable policies in P^{+}, P^{-}, E^{P}, and E^{U-} are selected by evaluating environmental conditions *envcond* using context information and the information stored in the subject and object profiles. Authorization in E^{U+} is instead always applicable as a default policy (i.e., permit all).

Figure 1 shows the policy evaluation flow, where each policy space is represented with a box that receives as input an access request and returns as output an evaluation response. It is assumed that, for each of the spaces introduced, the policy evaluation can result in three outcomes: i) true, positive evaluation; ii) false, negative evaluation; iii) unknown, no applicable policy has been found. Based on the response, the access request is granted, denied, or forwarded to the next policy space.

The evaluation process works as follows. First, policies in P^- are evaluated against the access request. If the evaluation result is 'true', the access is denied. Otherwise, the request is redirected and evaluated against the set of applicable policies in P^+. If the evaluation result of policies in P^+ is 'true', the access is granted. Otherwise, the request is redirected and evaluated in space E^P of planned exceptions. Like for policies in P^+, if the evaluation is 'true', the access is granted, otherwise, the request falls in E^{U-}. Note that the evaluation of applicable policies must take into consideration complex policies and their composition operators. When a request is redirected to E^{U-}, if the environment state of the request is not critical, the access is denied. Otherwise, the access is granted in E^{U+} by BtG, and the request is inserted into a log file. In both cases, the supervisor receives a notification of the request and the result of the evaluation. The supervisor is then able to perform a subsequent analysis to possibly individuate abuses or access requests that should be regulated by the defining a proper set of policies in spaces P^+, P^-, or E^P.

5. PROPOSED SYSTEM

We propose a BtG policy space auditor that generically incorporates a small set of rules to check for temporal inconsistency within the audit logs, which we adopt from Thorpe et al. (2013). The intended functionality enables the health care system administrator, an auditor or a forensic user to specify a timeline and an unplanned exceptions (E^U) set to be checked for temporal inconsistencies. The rules intended for the BtG policy space auditor uses the following algorithm.

Table 1: BtG Host System Inconsistency Algorithm

evtA= (null, null, **s**, "logon", "success")
evtB= (null, null, null, "modified", "success")
rule=**evtA** *happened-before* **evtB**
where field 2 of **evtA**== **x** and where field 2 of **evtB**== **x**
for each **evt** in **H(x)**
    if **evt**= (*, **x**, **s**, "logon", "success")
        a= index of **evt**
    if **evt**= (*, **x**, *, "modified", "success")
        b= index of **evt**
next **evt**
if **a** >**b** then
    rule has been broken

A function of our BtG policy space auditor's rule-base is that there are some events that need to occur before some other event can happen. This sort of relation between events is described as the *happened-before* relation (Gladyshev and Patel, 2005), and can be easily transcribed to this context. An example of such a relation between two events would be that a user x must "login" successfully to the computer host system before the user x can "execute" the application y. So the *happened-before* ($\rightarrow$) relation implies that the activity timeline, the time of the login access event must be before the time of the execution event. We express this as follows. Let $x \in \boldsymbol{P}$, $y \in \boldsymbol{A}$, and t_m, $t_n \in \boldsymbol{T_y}$. Then $((t_m, x, y, \text{login, success}) \rightarrow (t_n, x, \text{host system, execution, success})) \Rightarrow (t_m > t_n)$, where $\Rightarrow$ is the logical implication operator. Note that the *happened-before* relation is transitive.

After the construction of an audit log timeline (which is a sequence over the set of archetype events, ***Evt***), the BtG space log auditor is launched to evaluate all the events ordered by their timestamp. If an event evt_a has a *happened-before* relation to evt_b, but the audit kernel log timestamp (t_b) of evt_b suggests that evt_b occurred before evt_a then we can say that t_a and t_b are inconsistent. In order to detect this inconsistency, a rule base must be created that describes the *happened-before* relations for several classes or types of events. When the host machine's timeline is evaluated against the rule base, the inconsistent events can be identified and policy assertions about their timestamps can be made. In the healthcare context, observing an event that states that potentially addictive medication was ordered for a patient before the patient was checked into the hospital would be an example of an inconsistency that merited further scrutiny.

For the purposes of the rule base algorithm, let the time-lining function *H(x)* produce a timeline (where a timeline is an ordered set of discrete time instances) corresponding to a single episode of care for patient *p* by healthcare official *x*. The first rule states that a patient *p* must be admitted into the hospital before any other actions are possible on his behalf. The second rule states that healthcare practitioner *x* cannot prescribe medication for patient *p* before they have been checked in. If a prescription event evt_b occurs, the check-in event evt_a must happen before it, and evt_b must happen before the check-out event evt_c. Therefore, the physical time t_c at which the event evt_c must have occurred must be after the physical time t_b at which the event evt_b must have occurred, which must in turn be after the physical time t_a at which the event evt_a must have occurred.

If, given the two rules $evt_a \rightarrow evt_b$ and $evt_b \rightarrow evt_c$, and it is not the case that $t_c > t_b > t_a$, then the timestamps (t_a, t_b, t_c) do not reflect the physical times at which the system events must have occurred. The timestamps are therefore deemed to be inaccurate, as they suggest an internally inconsistent chronology within the evaluated BtG policy space. Such inconsistencies are flagged and brought to the attention of someone. From this example, the utility of the *happened-before* relation as a basis for proposing rules for the detection of inconsistent E^U events is evident.

6. CONCLUSION

In healthcare emergency situations, it is sometimes necessary to circumvent the access control system of the healthcare information system. Though "breaking the glass" may be critical to delivering care, it increases the set of possible security risks. The concept of policy spaces streamlines and optimizes the different types of security requests in a typical healthcare setting. We use this framework to examine the unplanned exceptions in healthcare audit logs for temporary inconsistencies. In future, we hope to 1) create a more robust set of rules that handle a wider range of temporally anomalous situations, and 2) extend this tool to provide insight within other policy spaces.

REFERENCES

Ardagna, C. A., De Capitani di Vimercati, S., Foresti, S., Grandison, T. W., Jajodia, S., and Samarati, P. (2010)."Access control for smarter healthcare using policy spaces". Computers & Security, 29(8), 848-858.

Ardagna, C. A., di Vimercati, S. D. C., Grandison, T., Jajodia, S., and Samarati, P. (2008)."Regulating exceptions in healthcare using policy spaces". In Data and Applications Security XXII (pp. 254-267). Springer Berlin Heidelberg.

Bhatti, R., and Grandison, T. (2007). "Towards improved privacy policy coverage in healthcare using policy refinement". In Secure Data Management (pp. 158-173).Springer Berlin Heidelberg.

Grandison, T., and Davis, J. (2007). "The impact of industry constraints on model-driven data disclosure controls", In Proc. of the 1st International Workshop on Model-Based Trustworthy Health Information Systems, Nashville, Tennessee, USA.

Rostad, L., and Edsberg, O. (2006). "A study of access control requirements for healthcare systems based on audit trails from access logs", in: Proc. of the 22nd Annual Computer Security Applications Conference, Miami Beach, Florida, USA.

Thorpe, S., Ray, I., Grandison, T., Barbir, A., France, R. (2013). "Hypervisor Event Logs as a Source of Consistent Virtual Machine Evidence for Forensic Cloud Investigations", in: Proc. Of the 27th Annual IFIP WG11.3 Working Conference on Data Security and Privacy(DBSEC), Newark, New Jersey, USA.

Gladyshev, P., and Patel, A. (2005). "Formalizing event time bounding in digital investigations," International Journal of Digital Evidence. Vol. 4.

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.