GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Building an Audit Engine to Detect, Record, and Validate Internal Employees' Need for Accessing Customer Data

Author: Jack Jekeon Cha, jackcha83@gmail.com
Advisor: Lenny Zeltser
GIAC (GLEG) Gold

Accepted: Nov 18, 2019

Abstract

When using Software-as-a-Service (SaaS) products, customers are asked to store and entrust a large volume of personal data to SaaS companies. Unfortunately, consumers are living in a world of numerous data breaches and significant public privacy violations. As a result, customers are rightfully skeptical of the privacy policies that businesses provide and are looking for service providers who can distinguish their commitment to customer data privacy. This paper examines the viability of building an accurate audit engine to detect, record, and validate internal employees' reasons for accessing a particular customer's data. In doing so, businesses can gain clear visibility into their current processes and access patterns to meet the rising privacy demand of their customers.

© 2019 The SANS Institute, Author Retains Full Rights

© 2019 The SANS Institute Author retains full rights.

1. Introduction

The Software as a Service (SaaS) product architecture requires that customer data be collected and stored under the service provider's domain. SaaS products are now widely used for many reasons. Enterprise customers continue to accelerate the staggering adoption of SaaS products ("Gartner" 2018; Preimesberger, 2019) for the benefit of not having to manage their own IT assets to operate an on-premise version of the same service. The general consumer market has also embraced SaaS services, ranging from early email accounts to social media and music streaming services. Consumers get to enjoy either the freemium or low monthly subscription payments that come with a continuous software update. For this great price and the undeniable overall benefit, some consumers have been willing to waver their privacy concerns. (Gashami, Chang, Rho, & Park, 2016).

However, with the rising of SaaS businesses, along came a series of data breaches and stronger customer data security/privacy concerns (Data, 2018; Privacy, 2018; Soofi, 2014). By the nature of SaaS service, customers have to allow their data to reside within the business's control. SaaS businesses have the responsibility not only to protect customer data but to fulfill their data privacy policy promises (Chang, Wong, Libaque-Saenz, & Lee, 2018; Wu, Huang, Yen, & Popova, 2012) sold to the customers. Privacy laws, such as the General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA), are furthermore asserting legal duties on corporations to meet formally. Especially with CCPA's compliance deadline of Jan 2020, many US-based SaaS companies are now undertaking a significant privacy capability modernization.

Successful software start-up companies are known for their exponential growth (Peechu, 2017). Growth is not only representing in the number of customers but also in the number of employees that handle customer growth, success and care ("KeyBanc", 2019). What began as a small circle of core employees quickly grows into a substantial distributed workforce with specialized roles and functions. In an attempt to limit access to customer data, an organization would typically rely upon a role-based access control (RBAC) system with various permission policy patterns (Rochaeli & Eckert, 2005) to mitigate the organization's perceived security risk.

© 2019 The SANS Institute Author retains full rights.

However, to fully utilize an RBAC system as an effective access control mechanism, one must continuously fine-tune the permission sets assigned to a role; a process known as role mining (Das, Mitra, Atluri, Vaidya, & Sural, 2018). Without the investment to regularly adjust the roles and the included permissions (i.e. continually mining and updating roles), an organization faces the risk of slowly drifting towards an over-provisioned state. The drifted state also leads to unclear ownership of the roles.

In practice, maintaining this fine-tuned state is often tricky. First, a role that is supposed to represent a particular job function often does not receive appropriate permission adjustments as the job function changes over time. Even with an attempt at updating a role, role owners are often content only to add new permissions and very reluctant to remove permissions in fear of potentially breaking one of the role assignees' access. Similarly, RBAC system owners often do not create a new role but instead add new permissions to an existing role for convenience, which results in some role assignees to possess more than the minimal set of permissions. Lastly, a statistical usage analysis of given permissions in a role is often not available. As a result, administrators have a difficult time figuring out whether the permissions are over-provisioned for an individual assigned to a particular role.

Even with the assumption that an RBAC system is finely tuned regularly, there is yet another weakness. Another shortcoming of an RBAC system is that "minimum necessary" access is not evaluated. For example, a customer support agent may be authorized to look up customer information for his or her daily job functions. Therefore, he or she would be assigned to a role that allows the action of accessing customer accounts. However, without a specific customer issue to solve, the agent does not *need* to look up a particular customer's data.

A SaaS company has many legitimate job functions that require access to customer data. For example, even though a company may have the maturity to test its product purely with test data, inevitable situations arise where the customer dataset is required to debug an issue that is not reproducible without the customer data set. Another obvious and more common workflow is customer support services. When a customer calls in to seek help, support agents have to pull up the customer account details to quickly diagnose and aid the customer in resolving their issue. So, how can we systematically validate the need of a customer support representative/engineer

© 2019 The SANS Institute Author retains full rights.

© 2019 The SANS Institute, Author Retains Full Rights

for accessing a particular customer's data on a given day? How can we precisely map each worker's workflow so the businesses can fine-tune the business practices to meet the rising privacy demand from customers?

This paper theorizes that for a modern SaaS company, audit trails in Customer Relationship Management (CRM) and bug tracking systems can be used to validate and justify all access events by finding an active customer's ticket at the time of the access event.

The author has chosen the above two systems under the belief that they are the de-facto central repository of audit records that capture the workflow of a metric-based workforce. For example, key performance indicators (KPIs) such as time to resolve, number of open issues, and age of open issues are indicators that are commonly tracked within a software company and which are only possible with systems like CRM and bug tracking systems. Furthermore, by leveraging the Application Programming Interface (API) functionality that comes with the above two systems, building an automated audit engine is theorized to be highly viable.

2. Research Method

2.1. Research Subject Organization Profile

A real SaaS company with less than one thousand employees was chosen as the subject of this research. This SaaS company offers a mature Human Capital Management (HCM) product to small and medium-sized businesses (SMB). The solution includes the management of health benefits and payroll management, as well. For the nature of the services offered, frequent interactions occur between the customer and company representatives. The following functions require customer data to support the provision of services. This distribution is typical of other SaaS companies:

Function	Example of Customer Data Access Scenarios
1. Customer Support	Validating the identity of inbound support requests. Triaging issues and providing guidance to the customer on how to use the platform.
2. Implementation	Implementation supports customers in accelerating the use of the SaaS platform by assisting with data entry and configuration. This

© 2019 The SANS Institute Author retains full rights.

© 2019 The SANS Institute, Author Retains Full Rights

	function may also include assisting customers with the integration of third-party partner services.
3. Customer Advisors	Periodically checks customer satisfaction level and help resolve customer issues before being escalated. Acts as an advocate and aid for the customer to successfully utilize the SaaS service.
4. Engineering	The root cause of a customer reported issue may be the presence of engineering bug. In such case, customer support team would escalate to an engineering team. Testing the fix on the customer data set confirms that the fix can be safely applied in the production environment.
5. Other	Escalation management, legal/compliance inquiries, billing/renewal inquiry, feature usage evaluation

Table 1: SaaS company's functional requirements for customer data access

2.2. Three Implementation Options for the Monitor Function

For a large workforce, internal tooling via a restricted web portal is developed to allow employees to interact with customer data without any direct database-level interaction. This portal typically provides customer search and data read/write functions to enable the workforce to carry out their job functions effectively and in a controlled and monitored fashion. The authentication mechanism is assumed to be built-in, leveraging individual accounts.

The target company has the maturity in controls and processes to provide appropriate internal tools as the only means to access customer data for its workforce. With this advantage, this customer data access web portal became the perfect choke point where a monitor function can be placed to capture access occurrences. The author has considered the following three architectures for placing the monitor function.

The first option, as illustrated below, was to place the monitoring function between the web portal and the database read and write transaction layer.

© 2019 The SANS Institute Author retains full rights.

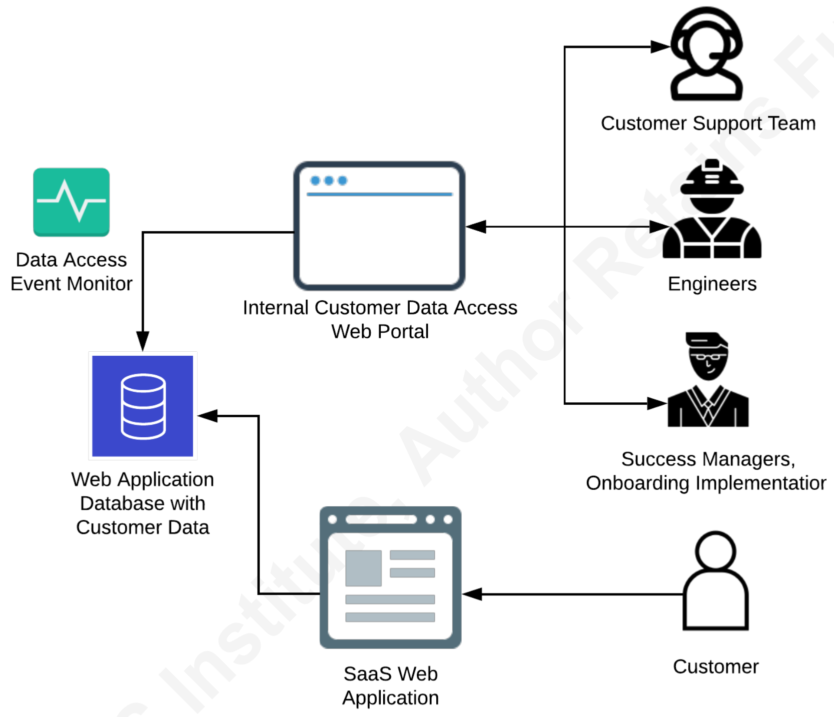

Customer Support Team
Data Access Event Monitor
Internal Customer Data Access Web Portal
Engineers
Success Managers, Onboarding Implementatior
Web Application Database with Customer Data
SaaS Web Application
Customer

Figure 1: Access Monitor to Inspect Database Transaction

The advantage of this approach is the complete capture of all read and write transactions that provides details of not only which customer's data is being accessed, but also, which specific data field is being accessed. For example, with this approach, the monitor function would be able to distinguish whether an internal employee has looked up a particular customer's specific data field such as date of birth. However, practical instrumentation requires building in the knowledge of which database table holds which fields, and also which data row belongs to which customer account. Also, the monitor function situated at this layer would not know which portal user account is invoking this database transaction (since the database access is given to the web application, not to each user). Lastly, inspecting all database transactions with additional table relation lookups to figure out the customer account to which it is associated, would introduce some level of performance penalties to the CPU usage (although not quantified in this study).

An alternative option was to place the monitoring function as a passthrough HTTP reverse proxy module, as illustrated below.

© 2019 The SANS Institute Author retains full rights.

© 2019 The SANS Institute, Author Retains Full Rights

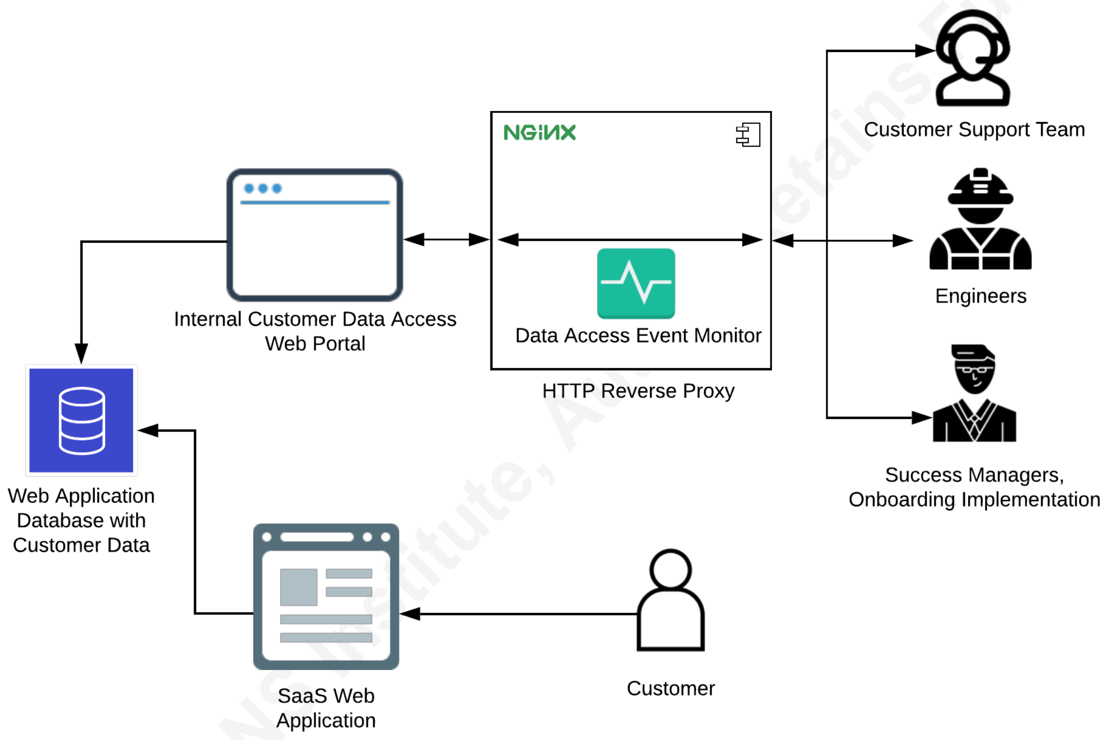

NGINX
Data Access Event Monitor
HTTP Reverse Proxy
Internal Customer Data Access Web Portal
Web Application Database with Customer Data
SaaS Web Application
Customer
Customer Support Team
Engineers
Success Managers, Onboarding Implementation

Figure 2: Alternative design for a monitor placement – HTTP Reverse Proxy

NGINX is a popular open-source web server that can be configured to be an HTTP reverse proxy. A reverse proxy, in simple terms, takes requests between a client and a server. The advantage of this approach would be relatively simple instrumentation cost as a plug-and-play module, especially if an organization already has a reverse proxy component deployed for its web portal (Sommerlad, 2003). This monitor can certainly be built as a plug-and-play module for an existing reverse proxy server.

For detecting customer data access events, the monitor function can inspect HTTP request paths and extract customers' unique identifiers directly. As an example, https://portal.saascompany.com/customer_info/12345 can indicate a customer data read event to a customer account whose ID is 12345.

Implementing path inspection would be a relatively low effort task and parsing the path string as a regular expression pattern match would require minimal compute power. By appointing a few common URL paths that include a particular customer's unique identifier, recording a complete list of accessed customer accounts is possible. However, unlike the first

© 2019 The SANS Institute Author retains full rights.

approach discussed, this approach does not grant the monitor function to gain visibility into which individual customer data fields are being accessed. The monitor function will simply record which customer's data is being accessed.

One additional drawback of using simple HTTP reverse proxy deployment is that user authentication/list synchronization with the portal would not be part of the out-of-the-box configuration of a typical HTTP reverse proxy server. In such a case, the monitor can only record which customer's data was accessed, but not who accessed the data. Configuring a shared source of user list/authentication may require a more significant amount of effort than initially thought, making this architecture less attractive.

The third and the last option places the monitor function within the portal web application itself, as shown below.

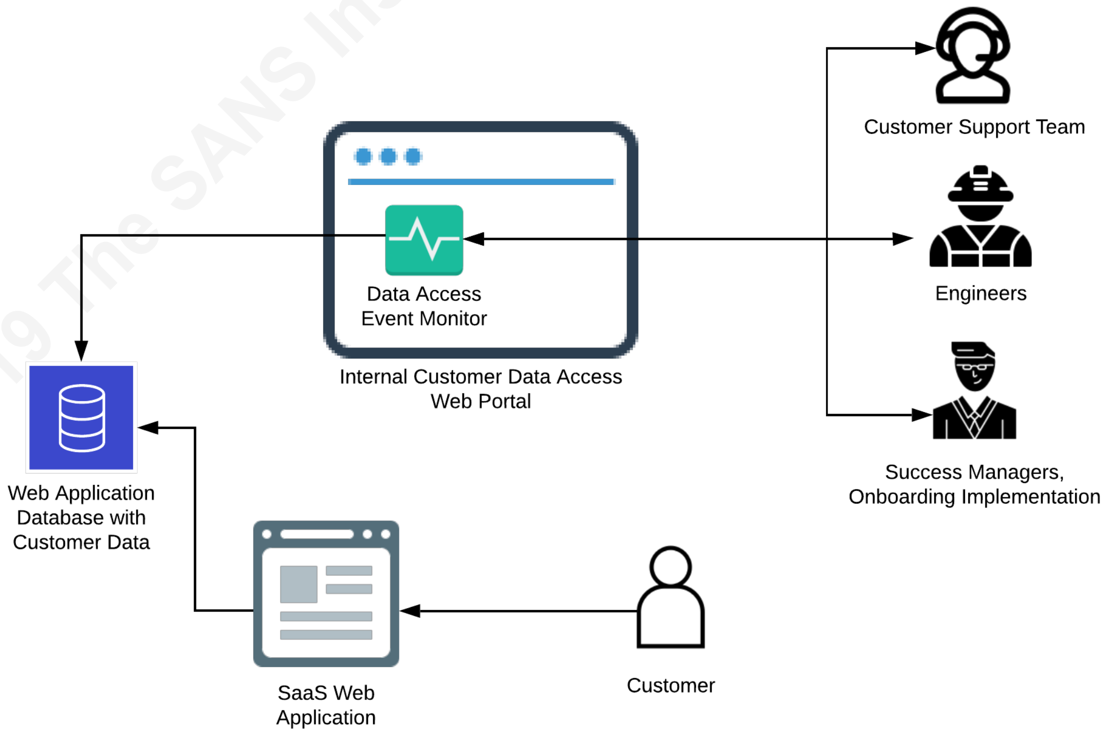


Figure 3: Access monitor embedded within the customer data access web portal

This approach requires making an application code change. However, as an integrated component to the web portal, such architecture provides many benefits. First, the monitor function can take the same approach as the second option in parsing the request path URLs to

© 2019 The SANS Institute Author retains full rights.

© 2019 The SANS Institute, Author Retains Full Rights

extract the accessed customers' unique identifiers. Additionally, since the monitor function and the rest of the web portal logic is in a monolith environment, the monitor can also find the relationship between tertiary objects and the owning customer to increase the scope of access audit coverage. For example, https://portal.saascompany.com/address_info/67890 has an address_info data object with a unique identifier of 67890. The monitor function can query to figure out to which particular customer this address_info object belongs to. By building in the known tertiary object URL paths and relationship-finding logic, the monitor function can be more thorough in compiling a list of customer data accessed.

Another advantage is the ability to record the user identity that is already authenticated to the portal web application. Hence, an authenticated user would not be able to repudiate the recorded customer data access events.

Below is a summary of the three options discussed, with the last option chosen to be the best fit for the test subject company.

Monitor Location Options	Accessed Data Field Detection Granularity	Implementation Complexity	Performance Penalty	Confidence of Capturing All Customer Data Access	Can capture user identity context for non-repudiation
1. Between Database and Web Portal	Very High	Moderate	Some	Very high	No
2. As an HTTP Reverse Proxy	Low	Easy	Negligible	High	No
3. Embedded Within Web Portal Application	Low	Moderate	Negligible	Very High	Yes

Table 2: Considered Options for Monitor Function Architecture

© 2019 The SANS Institute Author retains full rights.

2.3. Access Reason Validation – As a Batched Run

For this study, the author has chosen 24 hours, from midnight to midnight, for the monitor function to collect a list of customers accessed by internal employees. At the end of the monitoring period, the validation function picks the recorded customer list. The validator function makes API calls to the two systems (CRM and bug tracking) to locate any tickets associated with the recorded customer IDs.

Instead of conducting individual validation runs on each user separately, the validator function processes the combined list of all accessed customers as a batch process. Batch processing shortens the overall time taken to validate all access events because multiple users may have a common subset of customer account accessed. Such a scenario occurs as team members help each other by swarming to solve customer issues as a team.

If the overall validation run did not occur as a batch process, the run could repeat the same APIs calls multiple times. For example, User 1 accessed customer data belonging to customers [A, B, C] and User 2 accessed customers [B, C, D]. If the validation ran for each user separately, a total of six API calls for [A, B, C, B, C, D] would be made. But a batch run would only have to make four APIs calls for [A, B, C, D] as a combined single unique list, which would shorten the total daily validation time significantly.

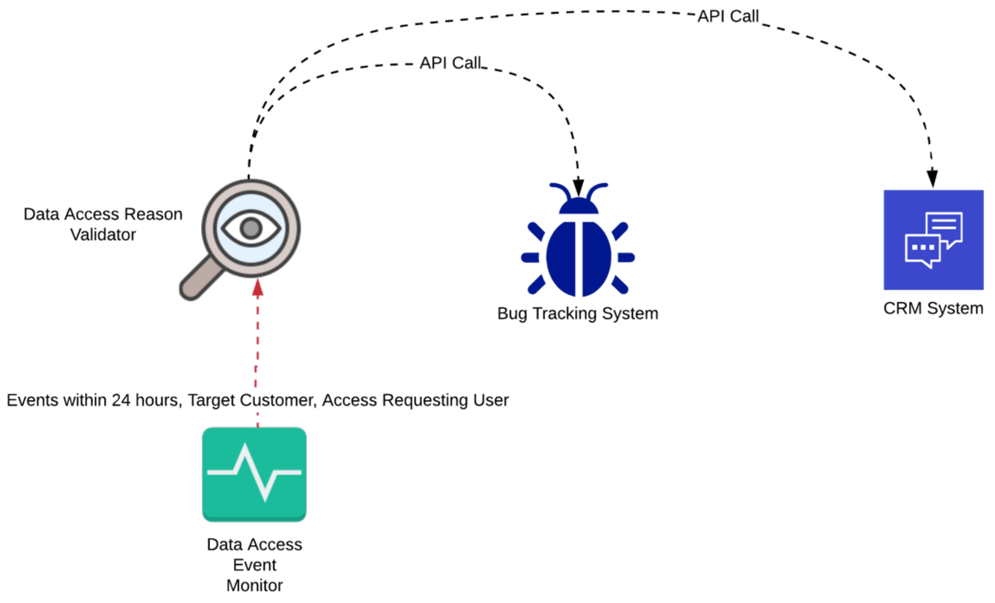

API Call
API Call
Data Access Reason Validator
Bug Tracking System
CRM System
Events within 24 hours, Target Customer, Access Requesting User
Data Access Event Monitor

Figure 4: Validator Function

Jack Jekeon Cha, jackcha83@gmail.com

© 2019 The SANS Institute Author retains full rights.

© 2019 The SANS Institute, Author Retains Full Rights

© 2019 The SANS Institute, Author Retains Full Rights

2.4. Access Justification Definition

Fast-moving SaaS businesses require their employees' workflows to stay agile and nimble to serve the customer faster and with the most efficiency. Customer support representatives and engineers often work as a team to solve customer issues quickly. Even though one particular representative may be the primary point person in handling a customer ticket, there may be multiple employees helping to solve a complex issue. For this reason, whether a customer recently has had an active issue ticket open is taken as the primary indicator for a customer data access to likely occur by multiple representatives. In the context of this study, a reasonably justified customer data access event is:

- When an unresolved/open CRM or a bug tracking ticket(s) exists for a particular customer or
- When a recently (with a short grace period) resolved/commented/updated CRM or bug tracking ticket exists for a particular customer

The second condition mentioned above gives portal users a few days grace period for situations such as following up on a customer whose issue may have been resolved (hence a closed ticket) but who may require further follow up.

Below is the flowchart depicting the validation priority. Each validation condition in blue requires an API call made to the two systems (CRM and bug tracker). Each API call comes with added delays which make an authenticated connection, a query processing time, and then a network delay between the request and response. The validation run is sped up by stopping if a record in found in the CRM system and not checking the bug tracking system. The author has assumed that the CRM system would have the highest probability of holding customer associated case records since the CRM system is supposed to capture interactions with customers as the first line contact.

© 2019 The SANS Institute Author retains full rights.

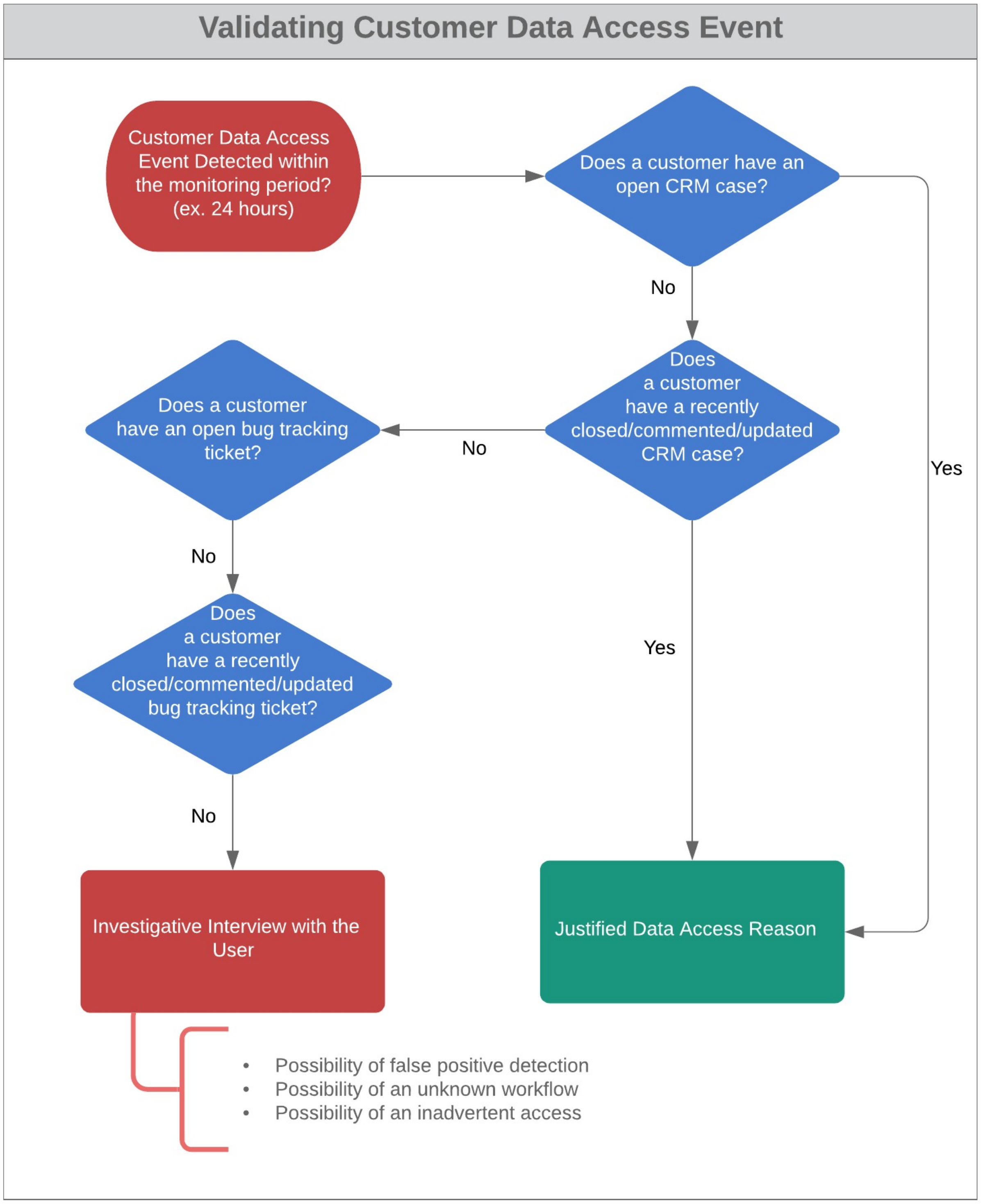
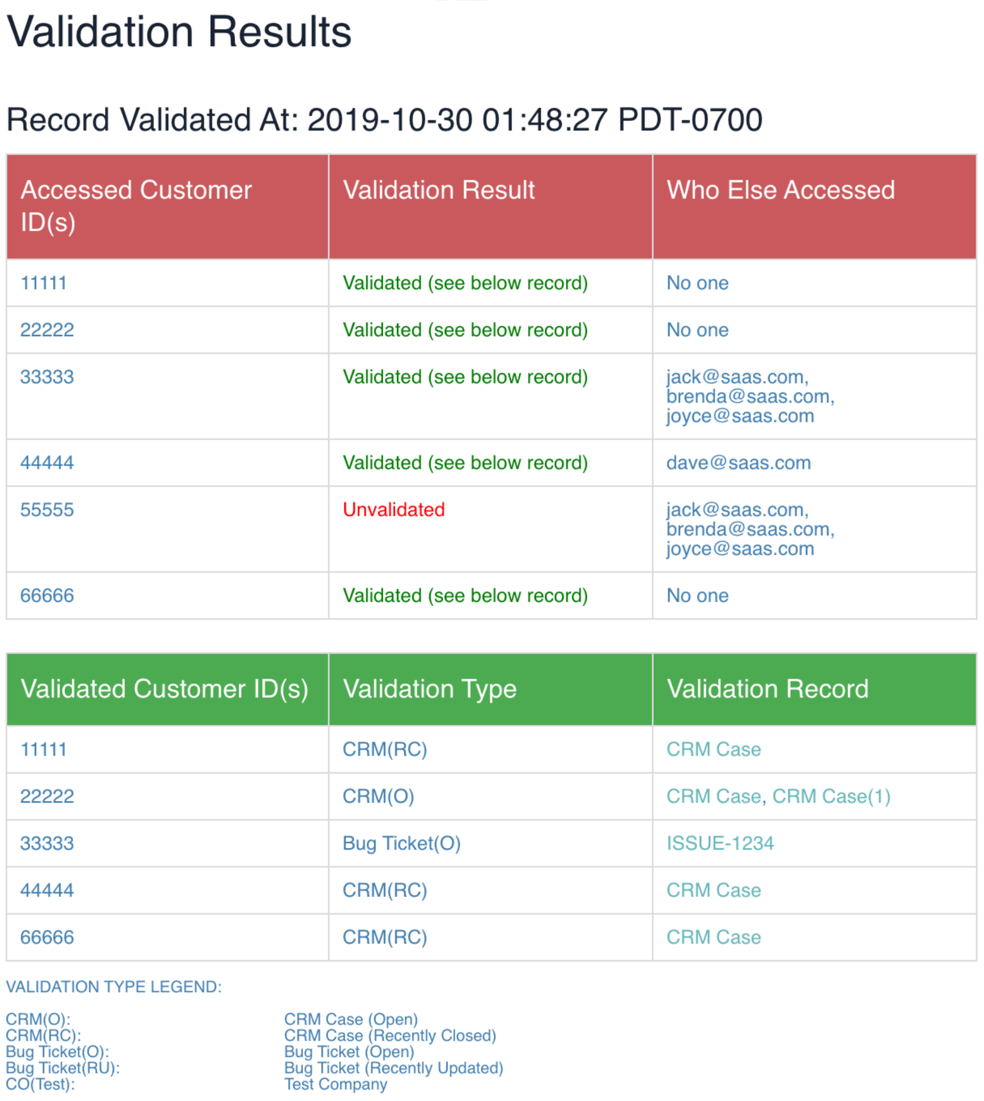

Validating Customer Data Access Event
Customer Data Access Event Detected within the monitoring period? (ex. 24 hours)
Does a customer have an open CRM case?
No
Yes
Does a customer have a recently closed/commented/updated CRM case?
No
Yes
Does a customer have an open bug tracking ticket?
No
Does a customer have a recently closed/commented/updated bug tracking ticket?
No
Investigative Interview with the User
Justified Data Access Reason
Possibility of false positive detection
Possibility of an unknown workflow
Possibility of an inadvertent access

Figure 5: Data Access Reason Validation Flow

© 2019 The SANS Institute Author retains full rights.

Below is a sample of a validation run. In the example, an employee has accessed six customer accounts. All but one customer account ID of 55555 has found records of customer tickets, either active or recently closed within the grace period. The "Who Else Accessed" column provides a hint of whether multiple representatives worked on the ticket or not. In this sample validation run, the representative appears to have worked with Jack, Brenda, and Joyce on the customer account of 33333 and 55555 together. Perhaps, the team has noticed that customer 55555 might be experiencing a similar issue to what customer 33333 has reported, although not yet noticed and reported by customer 55555.



Validation Results

Record Validated At: 2019-10-30 01:48:27 PDT-0700

Accessed Customer ID(s)	Validation Result	Who Else Accessed
11111	Validated (see below record)	No one
22222	Validated (see below record)	No one
33333	Validated (see below record)	jack@saas.com, brenda@saas.com, joyce@saas.com
44444	Validated (see below record)	dave@saas.com
55555	Unvalidated	jack@saas.com, brenda@saas.com, joyce@saas.com
66666	Validated (see below record)	No one

Validated Customer ID(s)	Validation Type	Validation Record
11111	CRM(RC)	CRM Case
22222	CRM(O)	CRM Case, CRM Case(1)
33333	Bug Ticket(O)	ISSUE-1234
44444	CRM(RC)	CRM Case
66666	CRM(RC)	CRM Case

VALIDATION TYPE LEGEND:

CRM(O):	CRM Case (Open)
CRM(RC):	CRM Case (Recently Closed)
Bug Ticket(O):	Bug Ticket (Open)
Bug Ticket(RU):	Bug Ticket (Recently Updated)
CO(Test):	Test Company

Figure 6: Sample Validation Run

© 2019 The SANS Institute Author retains full rights.

Through an interview process with employees, and by confirming with more verbose application request log, false positives were identified and trimmed. These false positives came from taking unique identifiers from request URL paths that were thought to be the customer ID but were not. For example, the path of */payroll_customer_id/12345 was generating customer data access event for customer 12345. However, 12345 referred to a unique identifier of a payroll account instance, not the customer account.

3. Findings and Discussion

The chart below depicts the validation result as percentage composition. The validation results are from a 72-hour period belonging to power users of the portal (i.e., those who accessed the highest number of customer accounts) in each department. The composition is rounded up to the nearest percentage.

Department Group	Access Volume (relative number of customer account accessed)	Open or recently updated case found in the CRM System	Open or recently updated ticket found in the bug Tracking System (Skipped if CRM record is found)	Accessed Customer Account is a Test Account	Combined Captured Workflow	Uncaptured Workflow
Customer Support	High	81%	0%	14%	95%	5%
Customer Advisors	Medium	90%	5%	0%	95%	5%
Engineering	Low	75%	8%	0%	83%	17%
Other	Low	38%	19%	8%	57%	35%
Implementation	Highest	5%	40%	11%	56%	44%

Table 3: Customer Data Access Reason Validation Composition Per Department

Customer support, customer advisor, and engineering group members show a high percentage of their customer data access workflow captured either in the CRM system or in the

© 2019 The SANS Institute Author retains full rights.

bug tracking system. In contrast, both other, and implementation group members seem to have workflows involving customer data outside of CRM and the bug tracking system. The implementation team had the highest relative volume of customer data access, showing that the implementation managers work as embedded customer representatives during the implementation. Engineering and other team members had a low volume of access events, as expected for their job function.

The author approached each group member to discuss the accessed customer accounts not found in the CRM or the bug tracking system. The following section discusses the insights from the interview sessions about each team's edge workflows cases that were not capturable in the CRM or the bug tracking system.

3.1. Customer Support Team's Uncaptured Workflow

As expected, the Customer Support team had a very high percentage of their customer data access accounted for in the CRM record. The uncaptured workflow of 5% included situations like triaging a customer issue where another customer had experienced a similar issue in the same time period. In such a case, support team members look for similarities between the two accounts. The other customer who had a similar issue before might have had a CRM ticket in the past. However, if the past case got closed beyond the grace period, the validation run would not count the closed ticket and the customer data access would be flagged as unvalidated.

3.2. Customer Advisor Team's Uncaptured Workflow

The Customer Advisor team had the highest percentage of its customer data access event captured in the CRM system, which was a pleasant surprise. The initial interview with one of the customer advisor team members indicated that each advisor manager has a book of business (i.e., a list of customers they are responsible for). A customer advisor manager would have regularly scheduled calls with his or her designated set of clients. The author was warned that for those regular calls, the customer's data may be accessed without any open CRM ticket associated. It seems that such case is only 5% of their workflow in real life, and the remaining 95% is associated with customers who have an active customer ticket.

© 2019 The SANS Institute Author retains full rights.

3.3. Engineering Team's Uncaptured Workflow

Although the CRM and the bug tracking system captured the majority of the engineering team's workflow (83%) that involved customer data access, the other 17% uncaptured workflows had several scenarios. First, when an engineer debugs an issue that one customer encounters, there could be other customer accounts that may be experiencing the same bug, although they may not have reported it yet. To confirm the manifestation of the issue, sometimes another customer account with matching bug dataset is accessed.

Another insight gained from the interviews is a scenario involves new feature development. Although the standard practice of preparing a test data set exists, sometimes the test data set itself has to be modeled after a real dataset from customer accounts to account for a realistic distribution of data points.

Also, another uncaught workflow scenario is the Engineering team directly interacting with platform developers who are also customers (e.g. paid development accounts). SaaS companies typically have open APIs for its users and partners to securely interact with the platform. To provide quick technical support, a developer support forum is often set up for engineers to answer technical questions quickly. The inquiries made through the forum space may sometimes lead to an engineer to check the inquirer's dataset to root cause the issue that the customer/partner developer faces.

3.4. Other Users' Uncaptured Workflow

Escalation managers and product team members often analyze customer issues to identify product gaps. This workflow represents 38% of their access event captured in the CRM system. It is interesting to note that this group accessed test accounts (8%), presumably to get a view of the current product. The uncaptured workflow scenarios included analyzing feature usage and feature requests coming from customers who did not have an active CRM ticket. Feature requests would typically come from customer community forums with any CRM interaction tied.

This group also contained uncaptured workflow for customers who are in the contract renewal period, where billing/renewal representatives have to familiarize themselves with the customer details and their use of the product for renewal or upsell purposes.

© 2019 The SANS Institute Author retains full rights.

© 2019 The SANS Institute, Author Retains Full Rights

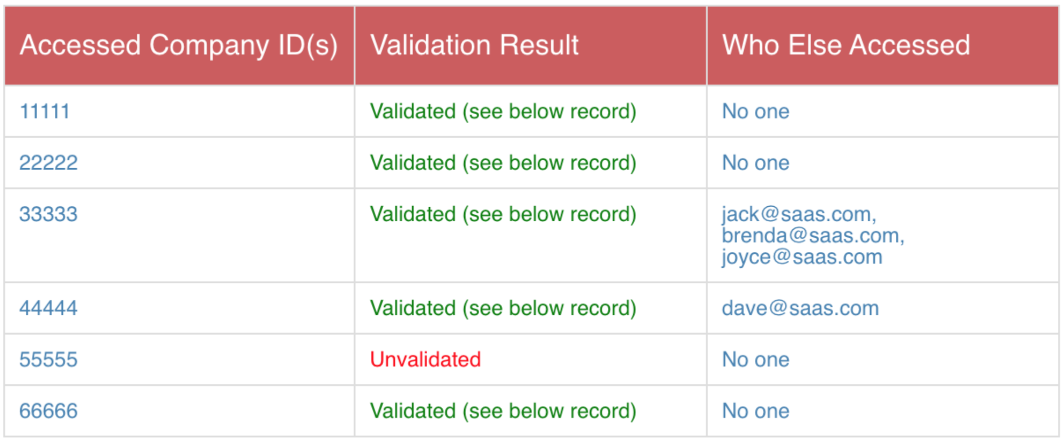
3.5. Implementation Team's Uncaptured Workflow

Naturally, the implementation team had a high percentage of uncaptured workflow because, during an onboarding phase, customers would not have an open CRM ticket. The implementation team had the highest volume of customer data access events.

3.6. Usage of Internal Instant Messaging Application

During the interviews with the users to understand their workflows, communication done between team members through an instant messaging application to seek help from others was mentioned significantly. This communication pattern that results in customer data access action by different actors was more prevalent than anticipated. For example, customer support team members often reach out to senior members through messaging to consult on customer issues that they could triage themselves. The senior members who decided to help end up looking up the customer's data.

Interestingly, almost all posted messages had a CRM case involved and the channel members were following an unspoken rule of message convention to include a web link to the CRM record containing a customer's unique identifier. Perhaps as future research, instant messages can be analyzed to find strong audit record for tracking why a certain employee ended up accessing which customer's data. Interestingly, when the customer data access events across multiple users were correlated, it became apparent which customer issue required team collaboration. For example, the validation run below displays many other users who looked up the same customer 33333's data.



Accessed Company ID(s)	Validation Result	Who Else Accessed
11111	Validated (see below record)	No one
22222	Validated (see below record)	No one
33333	Validated (see below record)	jack@saas.com, brenda@saas.com, joyce@saas.com
44444	Validated (see below record)	dave@saas.com
55555	Unvalidated	No one
66666	Validated (see below record)	No one

Figure 7: Example customer issue that involved team collaboration

© 2019 The SANS Institute Author retains full rights.

3.7. Self-Reporting Through Instant Messaging Application Bot

If the validator function can guarantee capturing greater than 95% of each team's workflow, the validator function could also be used to trigger an access challenge. Perhaps such higher coverage percentile can be achieved by expanding its validation scope beyond the CRM and bug tracking systems.

With prevalent usage of instant messaging applications like Slack, an automated bot interaction mechanism can be used to build a speedier challenge and response communication path, and/or as an exception filing pathway. Contrary to a traditional ticket-based exception filing process, which involves waiting for supervisory approval, workflow designed for a messaging application can be an effective and culturally accepted process for today's modern workforce.

The workflow through bot interactions within the messaging application has the potential to speed up the approval process significantly. At the same time, digital audit trails would be discoverable at scale with API instrumentation. Furthermore, the messaging app bot can be configured to auto generate an organization's traditional exception workflow ticket if desired. Such an easy, speedy, and psychologically well-accepted process could gain the cooperation of the workforce and be an essential key to validating all customer access events.

4. Limitation and Direction for Future Research

The overall findings indicate that even for job functions with well-defined customer interactions, CRM, and the bug tracking systems alone did not contain all audit records to validate 100% of customer data access events. Through post-validation interviews with the users, this study has discovered many different workflows that were not capturable in those two systems. It seems that customer status states (like whether the customer is onboarding, offboard, or in between renewal period) can and must supplement the validation process to increase the audit record coverage. As an example, a sales representative may be expected to look up information about a particular customer if he or she is developing a renewal contract. Hence, even without an active CRM case, the data access event may be marked as justified due to customer status.

© 2019 The SANS Institute Author retains full rights.

© 2019 The SANS Institute, Author Retains Full Rights

Furthermore, as an organization's workforce grows, more unexpected workflows that involve customer data access would arise. For example, during the research, it was discovered an individual whose job was to educate and train customer support agents on how to use the customer data access web portal effectively. The educator's daily activity involved building customer issue scenarios with test accounts – seeding dataset and feature configurations that would cause a customer issue. Although the educator built most of the scenarios without using any real customer account data, some scenarios required researching prior customer issues to build test datasets that would represent complex customer issues. Hence, for a growing workforce, a flexible exception filing pathway for ever changing workflow requirements is much needed, especially for scenarios that are not capturable in traditional issue tracker systems like CRM and bug tracking.

4.1. Future Research: Data Access Behavior Analytics Engine

With the daily history of each employee's customer data access activity captured, it may be possible to build a baseline profile of each employee's daily workload expected with the following input:

- The average number of customer data access events per day
- Time of the day in which customer data access events occur at
- Peak business season where the user's workload is much higher than average
- Book of business / customer segments assigned (if applicable)

Future research into building an accurate baseline profile for each job function would be a valuable topic for detecting an abnormality in access event patterns.

4.2. Future Research: Active Gating Mode

For most of, if not for all of the security controls, fine-tuning the detection algorithm to the healthy balance of false-positive and true-negative is the most crucial part. With such a healthy balance, and perhaps in combination with the access behavior analytics engine assisting, the validator function may be able to detect unauthorized data access attempts in real-time with low false-positive rates. An active block mode may interrupt workflows, but it would prevent the

© 2019 The SANS Institute Author retains full rights.

© 2019 The SANS Institute, Author Retains Full Rights

inappropriate data access events from ever occurring in the first place. Future research into developing policy rules on when to trigger the hard-gating challenge dialog to block proceeding to customer data access without significantly affecting the productivity of teams would be another valuable endeavor. Such effort will propel the gating algorithm to become an Attribute-Based Access Control (ABAC) system that can dynamically regulate an employee's access to customer data (Das, Mitra, Atluri, Vaidya, & Sural, 2018; Hu, Kuhn, Ferraiolo, & Voas, 2015; "Axiomatics" 2019).

5. Conclusion

Customers face a unique dilemma in the age of SaaS proliferation. SaaS companies attempt to delight their customers with exceptional product updates and efficient customer services. The low monthly subscription fee schedule also helps decide with ease in signing up for the service. However, customers now have to entrust a significant amount of their digital data and identity to SaaS companies. More and more SaaS enterprises are entrusted with an ever-growing amount of customer data. After being affected by mega breaches, general consumers realize the real risk associated with their digital assets being in the hands of SaaS companies. Understandably, the consumers now are calling for more transparency into how internal SaaS employees are accessing their data.

This paper has attempted to build an audit record framework that can act as a baseline to answer not only who accessed which customer's data, but also why the data access had to occur. Although the thesis statement was proven to be wrong, this study allowed the author to gain insights into different workflows of a modern SaaS workforce that were not capturable in traditional ticketing systems. As the next step, the author hopes to continue researching into the future topics of building a flexible challenge-response pathway via instant messaging platform, behavior analytics, and real-time gating capability. Ultimately, the future research effort is to lead to the full maturity of tying all customer data access to an authorized and documented use cases. With the augmentation of forcibly gating access control, future research effort hopes to prevent any inadvertent access events that do not align with a justified and fully documented use cases.

© 2019 The SANS Institute Author retains full rights.

References

Axiomatics. 2019. *Attribute Based Access Control (ABAC).* Retrieved from https://www.axiomatics.com/attribute-based-access-control/

Chang, Y., Wong, S. F., Libaque-Saenz, C. F., & Lee, H. (2018). *The role of privacy policy on consumers' perceived privacy*. Government Information Quarterly, 35(3), 445–459. https://doi.org/10.1016/j.giq.2018.04.002

Das, S., Mitra, B., Atluri, V., Vaidya, J., & Sural, S. (2018). *Policy Engineering in RBAC and ABAC*. In From Database to Cyber Security (pp. 24-54). Springer, Cham.

Data Breach Investigations Report 2018: the year of ransomware. (2018). *Computer Fraud & Security*, 2018(5), 4. https://doi.org/10.1016/S1361-3723(18)30040-X

Gartner Foresees the End of On-Premises ECMs in 2019. (2018). *Information Management Journal, 52(6), 10*. Retrieved from https://search.ebscohost.com/login.aspx?direct=true&db=iih&AN=133397571&site=ehost-live

Gashami, J. P. G., Chang, Y., Rho, J. J., & Park, M.-C. (2016). *Privacy concerns and benefits in SaaS adoption by individual users.* Information Development, 32(4), 837–852. https://doi.org/10.1177/0266666915571428

Hu, V. C., Kuhn, D. R., Ferraiolo, D. F., & Voas, J. (2015). *Attribute-based access control*. Computer, *48*(2), 85-88.

Peechu, S. (2017). *Re-Imagining the Rule of 40 for Early Stage Startups: The 70 Percent Growth Efficiency Heuristic*. Siliconindia, 69. Retrieved from https://search.ebscohost.com/login.aspx?direct=true&db=iih&AN=127479749&site=ehost-live

© 2019 The SANS Institute Author retains full rights.

© 2019 The SANS Institute, Author Retains Full Rights

Preimesberger, C. (2019). *Key SaaS Trends in the Enterprise. EWeek, N.PAG*. Retrieved from https://search.ebscohost.com/login.aspx?direct=true&db=iih&AN=137109930&site=ehost-live

Privacy Rights Clearinghouse. 2019. *Data Breaches Dataset*. Retrieved from https://www.privacyrights.org/data-breaches

Rochaeli, T., & Eckert, C. (2005). RBAC policy engineering with patterns. In *W9: The Semantic Web and Policy Workshop (SWPW)* (p. 148).

Soofi, A. A., Khan, M. I., Talib, R., & Sarwar, U. (2014). *Security issues in SaaS delivery model of cloud computing*. International journal of computer science and mobile computing, *3*(3), 15-21.

Sommerlad, P. (2003, June). *Reverse Proxy Patterns*. In EuroPLoP (pp. 431-458).

Wu, K.-W., Huang, S. Y., Yen, D. C., & Popova, I. (2012). *The effect of online privacy policy on consumer privacy concern and trust*. Computers in Human Behavior, 28(3), 889–897. https://doi.org/10.1016/j.chb.2011.12.008

© 2019 The SANS Institute Author retains full rights.

© 2019 The SANS Institute, Author Retains Full Rights