State of Application Secret Delivery and Audit Practices

2019

Date: **July 2019**

Author: **Stephen Kuenzli**
Principal Consultant
QualiMente
www.qualimente.com

Status: Published
Version: 1.1.3

Summary

QualiMente investigated DevOps Practitioners' processes for delivering secrets to applications *and* analyzing those processes for threats. This research tests two hypotheses:

1. Most practitioners are dissatisfied with their application secret delivery processes
2. Most practitioners do not have secret usage audit or threat detection mechanisms

The study collected data from practitioners using interviews and a survey offered through DevOps-focused forums, primarily the Phoenix DevOps Meetup and the #NoDrama DevOps Mailing List and Blog.

The study recruited participants from DevOps forums because these people are often responsible for the automated delivery of applications to runtime environments.

This report analyzes responses from the survey's 17 respondents (Appendix - Demographics). This response set is small, but there is a clear signal on several important matters:

- The Majority Are Not Satisfied
- Lack of Satisfaction Looks Risky
- Most Lack Tooling to Detect Unauthorized Secret Use

We learned most traditional secret vaults and privileged access management solutions are difficult to integrate safely into the automated delivery pipelines that are cornerstones of DevOps practices. This puts secrets like user credentials, API keys, and TLS certificates at risk. These #NoDrama DevOps posts describe the problem of bootstrapping trust for automated delivery and operations in more detail: The First Secret Problem, Access Control in CI/CD systems.

The biggest challenges to securing and auditing application secret delivery processes found in the study are:

1. Engineering staff lack understanding of how to solve The First Secret Problem. Unique combinations of secret vaults and deployment platforms complicate the Problem.
2. Applications or delivery tooling do not support safe secret management practices.
3. Teams lack tools to detect suspicious or unauthorized use of application secrets.

The following sections support these conclusions using both quantitative and qualitative data gathered during the research. These findings confirm both hypotheses under investigation, at least for practitioners sampled from the DevOps community.

This report provides resources for improving secret delivery process, including a generalized recommendation for how to Structure a Secure Application Secret Delivery Project for Success in your own environment.

QualiMente can help you assess and improve your secret delivery and management processes. Contact us at devops@qualimente.com

Delivering Secrets Securely

Continuous Delivery and similar automated application delivery methods gained wide adoption over the past 10 years[1]. Teams spent much of that time figuring out how to integrate the various aspects of quality into delivery pipelines. Customer demands drove improvements to functional correctness, performance, and reliability. Security is less visible to customers and many organizations have delayed integrating security practices such as vulnerability scanning into regular delivery processes.

If integration of application security into delivery processes has been late, what about the security of the delivery process itself?

The first goal of this research was to determine if DevOps practitioners were satisfied with their current process for delivering secrets to applications using automated delivery pipelines. Let's establish some context before diving into analysis.

Automated delivery processes are highly-privileged control planes that configure and deploy applications to a runtime environment where they can be used. These processes are generally built on tools that prioritize getting users started quickly over security. Delivery tools often default to open access to execute processes and inspect all related configuration. Keeping data confidential is difficult with the current delivery tools. For example, deployment jobs may lack access control and secrets are often available as environment variables in a shell script where they may be printed accidentally, copied unnecessarily, or stolen intentionally.

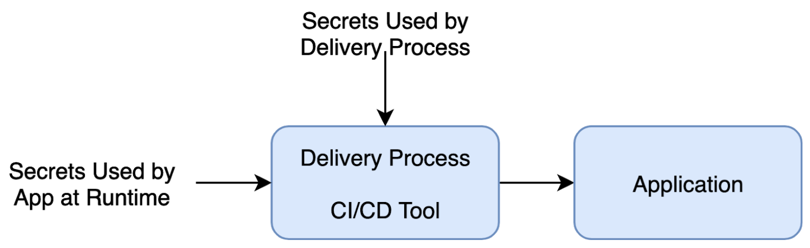
Secrets Used by Delivery Process
Secrets Used by App at Runtime
Delivery Process
CI/CD Tool
Application

Figure 1: Secrets Used in Application Delivery

Application secrets are the most valuable information in an application delivery process. There are two kinds of secrets used in application delivery. First, there are the secrets used by the application to identify itself to datastores and collaborating services at runtime. These secrets are used to read, change, and store application data. Second, there are secrets used by the delivery process itself. The delivery process uses these secrets to manage application-related infrastructure. These secrets, generally Cloud API Keys, are used to create, change, or destroy compute, databases, and network services. Infrastructure API keys are usually *very* powerful.

Now let's examine what DevOps Practitioners said about their satisfaction with their application secret delivery processes, biggest challenges, and what they would improve if they could.

[1] Continuous Delivery (Humble & Farley) was published in 2010

The Majority Are Not Satisfied

We asked DevOps Practitioners:

```
How satisfied are you with your current process for delivering secrets to
applications in their runtime environment via an automated pipeline?
```

Table 1 summarizes respondents' satisfaction:

How satisfied are you with your current process?	**Count**	**% of Total**
Dissatisfied	6	35%
Neither satisfied nor dissatisfied	6	35%
Satisfied	3	18%
Very satisfied	2	12%
Grand Total	**17**	**100%**

Table 1: Satisfaction with Current Application Secret Delivery Process

Only 30% of respondents are 'Satisfied' or 'Very Satisfied' with their delivery process. This means **70% of respondents are not satisfied** with their current automated application secret delivery process. This is concerning because applications operate with privileged access and the power to copy, manipulate, corrupt, or destroy application data.

Participants reported several kinds of challenges with handling secrets safely in application delivery pipelines:

- Not *understanding* how to handle and deliver secrets safely
- Missing or inefficient *tooling* impeding process automation or scalability
- Incomplete support of application delivery platforms by tooling, particularly 'on-prem'

Let's dive further into these challenges and what DevOps practitioners need to improve the safety of application secret delivery.

Challenges and Opportunities for Improving Secret Delivery

The study posed open ended questions for participants to describe both the biggest challenges and which single aspect they would improve in their application secret delivery process. Let's start with the question about challenges:

```
What are the biggest challenges in delivering secrets safely to
applications in your runtime environment?
```

The three most representative responses were:

- "Lack of knowledge, lack of time allocated to figure it out"
- "The inability of technologies to be interoperable between each other, such as Kubernetes Secrets and Vault"
- "Application architecture, not lending itself to secret injection. Currently deploying to [redacted Container Orchestrator]. Moving to a containerized future state."

The responses to Challenges distill to two main messages:

1. **Engineering staff lack understanding of how to solve The First Secret Problem, *especially* using the respondent's particular secret vaults and deployment platforms**
2. **Applications or delivery tooling not supporting safe secret management practices**

Respondents confirmed those Challenges when describing what they would improve:

```
If you could improve a single aspect of your application secret
management and delivery process, what would it be?
```

The eight open-ended responses to this question classified to the following categories (a single response may classify to multiple categories):

	Count
Understanding	3
Tooling	5
Advanced Practice	3
Integration Combinations	2

Now let's break down each of these categories further and analyze the responses.

Understanding
The response excerpt that most typefied the desire to improve understanding of how secret delivery could and should work in their environment was:

> "having presentations that explain [how to deliver application secrets securely] with Cloud providers and how to do something approximate deploying On Prem"

This response also classified for 'Integration Combinations' because the respondent calls out that they need to know how to secure secret delivery in the Cloud and in their on-premises datacenter.

Tooling
Tooling is the most frequently suggested aspect to improve. This is a natural finding given the complexity of handling secrets safely and the many possibly deployment platforms and application technologies.

Tooling responses identified the following needs:

- a way for developers to access/use secrets in development environments
- a secret delivery tooling and process that exists through *all* stages of SDLC, e.g. dev through prod to improve consistency of automation
- support for rotating secrets

Take these needs into consideration when designing and evaluating solutions so that you are neither surprised nor disappointed that it is, e.g. difficult to rotate secrets.

Integration Combinations
In survey responses as well as interviews and client engagements, engineers cite addition of application deployment targets as a primary burden to delivery secrets safely. Often, the issue is lack of APIs or automation capability for 'on-premises' systems.

Advanced Practices
Several people with a functioning secret delivery process identified 'Advanced Practices' they would adopt to improve those processes. These three survey responses identified two specific aspects to improve: ***key rotation*** and ***governance***.

Key rotation is the most-identified 'Advanced' practice. Even people who are delivering secrets securely have trouble with rotating secrets.   Overall, this sounds really bad for people in 2019! Key or secret rotation is a core risk mitigation technique and yet even people with tooling to manage secrets find it challenging.

Here’s how one participant responded (*emphasis* added):

> “rotation and/or versioned secrets (*currently very very hard to do so we don't do it*)”

The second kind of response in the ‘Advanced’ category concerns strengthening *governance* of the secret delivery and management process. People would like to ‘remove humans’ and the ‘possibilities of weak practice’ from their delivery processes. Governance includes defining policies and supporting implementations of security practices. Examples include limiting access to credentials and rotating them periodically to limit the impact of a lost credential.

One way these advanced practices come together is the capability of rotating a credential without exposing it to a person using automation. Automation of stronger security policies demanded by governance can faithfully execute generating a strong credential, storing it safely, and triggering application deployments without wearing people out. Desire for this capability surfaced multiple times in research interviews and discussions.

How to use this information: This information provides reasons why DevOps practitioners are dissatisfied with their application secret delivery processes. It also provides insight into challenges you will face when improving your own secret delivery process. The Structuring a Secure Application Secret Delivery Project for Success section outlines how to set an improvement effort up for success.

But before we get to that, let’s examine one of the major reasons you might want to improve your secret delivery process: Risk.

Lack of Satisfaction Looks Risky

Losing control of an application credential can be just as impactful as losing a system administrator credential. For example, an attacker might use those credentials to alter or destroy data. Alternatively, a delivery pipeline might wreak havoc by accidentally using credentials for a production environment instead of development. This can happen through innocent copy-paste of pipeline automation when proper access controls are not enforced.

Let's examine how Satisfaction relates to Risk in application secret delivery processes.

The 'Risk by Satisfaction' figure below plots each respondents' feelings of satisfaction along with a risk assessment for their application secret delivery pipeline. Each person on the plot represents a single respondent's assessment of Satisfaction and Safety for their application secret delivery pipeline. The questionnaire guided respondents in assessing the risk quantitatively in order to improve comparability of responses. This plot suggests several interesting relationships.

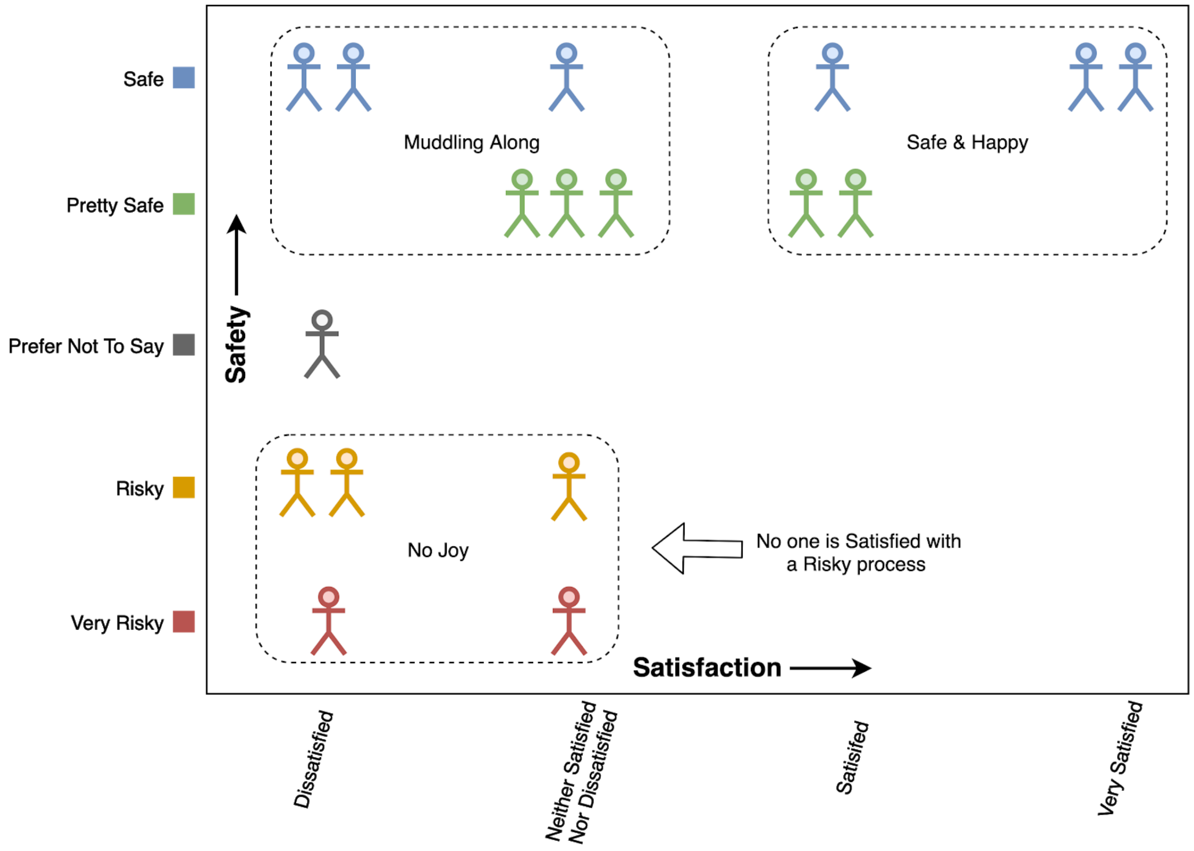

Figure 2: Risk by Satisfaction

First and most importantly, the only respondents reporting that their secret delivery processes were 'Risky', 'Very Risky', or 'Prefer Not to Say' are those that also reported being 'Dissatisfied' or 'Neither Satisfied nor Dissatisfied' with those same processes. These responses are identified in the lower-left *'No Joy'* zone of the plot. This study cannot say whether an explicit or intuitive estimation of risk in the delivery process is causing dissatisfaction, but they appear to be correlated. Given this data, *Satisfaction* seems like a good proxy for *Risk*. More on that later.

By contrast, the *'Safe & Happy'* zone is occupied by respondents who were 'Satisfied' or 'Very Satisfied' with their delivery process and *only* reported feeling their process was 'Safe' or 'Pretty Safe'. No one who is satisfied with their secret delivery process called it 'Risky' or 'Very Risky'. The lower right corner of the Risk By Satisfaction plot is empty. It is easy to imagine this result, but it's good to see such a conjecture supported with data.

The people in the final *'Muddling Through'* zone who report feeling Dissatisfied and Safe might surprise you. The delivery process is safe, after all. These responses reflect that more goes into satisfaction with a secret delivery process than safety. Research interviewees and respondents reported problems with the scalability and efficiency of executing the delivery process, especially rotating secrets.

For example, a team using SOPS[2] to deliver application secrets needs to keep tight control of which people and applications have access to the Cloud or PGP encryption keys used to encrypt and decrypt secrets. Small sets of people on Ops or Security teams often end up with the responsibility to rotate *all* credentials because scaling out the key access policy to application teams can be difficult. Alternately, the organization may have invested in building custom, maybe-not-so-great tooling to integrate with a difficult Enterprise Vault API. These are both signs **integration of common DevOps and Security tools is immature**.

A statistical analysis quantified both the strength and confidence of the correlation between the Satisfaction and Risk factors. There is a positive association of 0.500 for satisfaction with a secret delivery process and the safety of that process. This association is statistically valid at an 85% confidence level. See Appendix - Research Methods for details of the statistical analysis.

Practically, a response of lack of satisfaction is associated with a risky process about 50% of the time, with challenges relating to platform integration, efficiency, advanced practices accounting for the rest.

How to use this information: If you're discussing application secrets with someone and they aren't satisfied with the way applications get their secrets, your Spidey-sense should start to tingle. Probe into the risk around the delivery process or the methods by which applications read their secrets in the runtime environment. There's a good (~50%) chance that some very important risks are lurking just around the corner.

Next, we'll show how to set a secret delivery improvement project on the right track.

[2] SOPS is a tool that helps you encrypt, manage and use secrets in structured file formats like YAML https://github.com/mozilla/sops

Structuring a Secure Application Secret Delivery Project for Success

This study gathered feedback from DevOps practitioners throughout all stages of implementation, satisfaction, and risk levels. Based on this feedback and knowledge of how secret delivery works, we can suggest a set of steps needed to adopt a secret delivery process that fulfills the need of delivering application secrets securely from automated delivery pipelines.

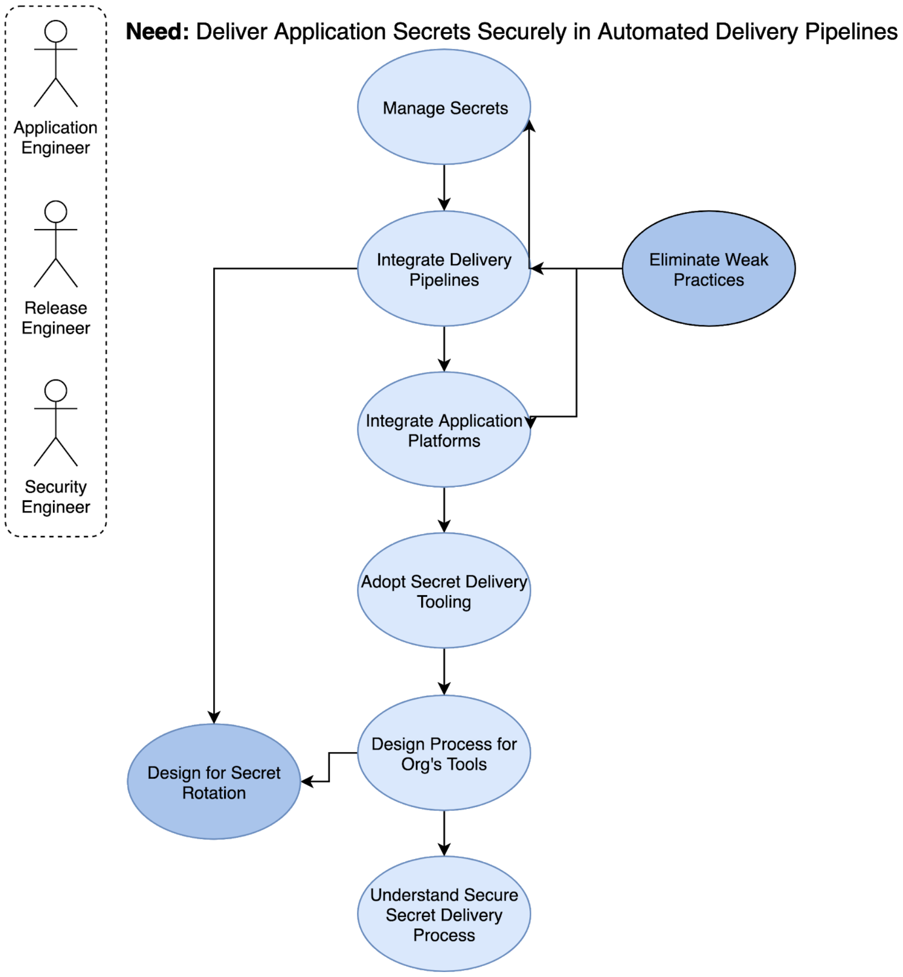
Application Engineer
Release Engineer
Security Engineer
Need: Deliver Application Secrets Securely in Automated Delivery Pipelines
Manage Secrets
Integrate Delivery Pipelines
Eliminate Weak Practices
Integrate Application Platforms
Adopt Secret Delivery Tooling
Design Process for Org's Tools
Design for Secret Rotation
Understand Secure Secret Delivery Process

Figure 3: Secret Delivery Value Chain

Figure 3 depicts secret delivery process dependencies in the form of a Value Chain. This Value Chain can be mapped onto your *own* technology landscape using a Wardley Map to help you explore what you need to implement a secret delivery process in your environment. It also outlines a Work Breakdown Structure for project execution.

The next section will examine the audit practices and capabilities DevOps Practitioners use for application secrets.

Auditing Secret Usage

The second goal of this research was to examine DevOps practitioners' audit practices to determine if people would know if and when they lost control of a secret.

Most Lack Tooling to Detect Unauthorized Use

The study asked participants:

```
Do you have any tools that help you audit or detect unauthorized use of
an application secret?
```

At least half of respondents aren't using tools to audit or detect unauthorized use of an application secret. Questions about audit capability were added in version 2 of the survey and there are a total of thirteen responses on this topic.

- Half of the respondents (7/13) said they do not have the tools to audit secret use or weren't sure if they had tools; two respondents preferred not to say
- Only a quarter of respondents (3/13) reported having a tool to audit secret usage

When asked what single aspect they would like to see improved in their secret usage auditing process, respondents answers boiled down to:

- Start auditing secret usage or manage audit logs better
- Adopt a simple tool to monitor secret usage and detect anomalies
- Provide real time, reliable alerts and hooks into the software delivery lifecycle

This data shows that most DevOps practitioners do not generally have an effective secret use audit and analysis toolset. In addition to the audit features described above an audit toolset needs to deal with the structural and operational forces that are currently transforming application deployments.

Structural Change - Applications are Decomposing

Organizations are transforming their application architectures to provide teams with independence and autonomy. This decouples teams from each other and speeds each team's individual deliveries. The architecture is transformed by decomposing large applications into collections of smaller, specialized components with independent delivery processes. Those smaller components might be services, microservices, or functions.

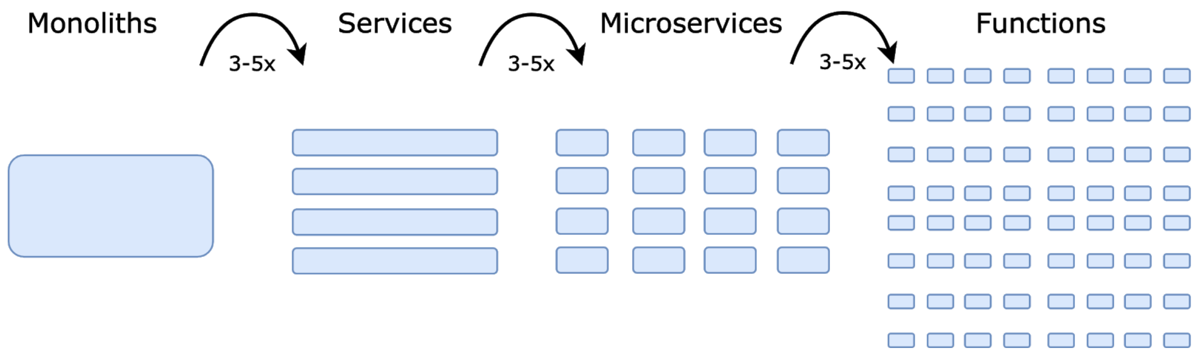

Figure 4: The Number of Independent Application Components is Growing

A challenge for audit practices is that each stage of decomposition will likely bring 3-5x more components that each need an access policy defined, enforced, and monitored for anomalies. Figure 4 depicts the geometric progression of an architecture that splits each component into four smaller components at each stage of decomposition from a *single* monolithic component to *64* functions.

It would be wise for growing businesses transforming their architecture to plan for a 5x, 10x, or even greater increase in application components to account for decomposition and growth over the next five years.

Operational Change - Application Instances are Becoming More Ephemeral
A second challenge is that the deployment and operation of application instances is becoming more dynamic. Organizations are adopting practices such as Continuous Delivery, containerized deployment on generic platforms, and autoscaling to respond load and manage costs. As a result, application processes may live for only a few minutes or seconds and be identified only by a generated id. Monitoring and audit practices requiring any sort of manual activity will not be able to keep pace. Older tools may have trouble organizing an ever-growing list of application instances. Increasingly rapid application change and instance turnover means applications' access to secrets and auditing needs to be automated.

Ensure audit tools you adopt scale along with the major technical trends changing application structure and operations so that you are not buried under a mountain of static, out of date reports.

What about Compliance?

Not knowing whether application secrets are being used without authorization sounds bad for security. You may be wondering whether it creates a regulatory or security framework compliance issue. As with many compliance questions, it depends on what security procedures you have defined and what systems are in scope for the assessment.

Let's try to answer whether detecting unauthorized use of a secret is required by HIPAA.

HIPAA

The Health Insurance Portability and Accountability Act (HIPAA) law governs the security of electronic patient health information (EPHI) and systems in the United States.

One of the core rules of HIPAA is the “Information System Activity Review” rule, which says:

> §164.308(a)(1)(ii) (D) Implement procedures to regularly review records of information system activity, such as audit logs, access reports, and security incident tracking reports.

and the “Audit” rule says:

> §164.312(b) Implement hardware, software, and/or procedural mechanisms that record and examine activity in information systems that contain or use electronic protected health information.

Compliance with both of these rules is ‘required’ by HIPAA, as opposed to ‘addressable’ at the provider’s discretion. However, HIPAA provides no specific requirements or guidance on *how* to audit a covered system. Health & Human Services’ Technical Safeguards[3] says:

```
It is important to point out that the Security Rule does not identify
data that must be gathered by the audit controls or how often the audit
reports should be reviewed. A covered entity must consider its risk
analysis and organizational factors, such as current technical
infrastructure, hardware and software security capabilities, to determine
reasonable and appropriate audit controls for information systems that
contain or use EPHI.
```

Let’s make these system audit recommendations more concrete with an example wherein two health care companies exchange data using industry-standard integration techniques.

[3] HIPAA Security Standards: Technical Safeguards
https://www.hhs.gov/sites/default/files/ocr/privacy/hipaa/administrative/securityrule/techsafeguards.pdf?language=es

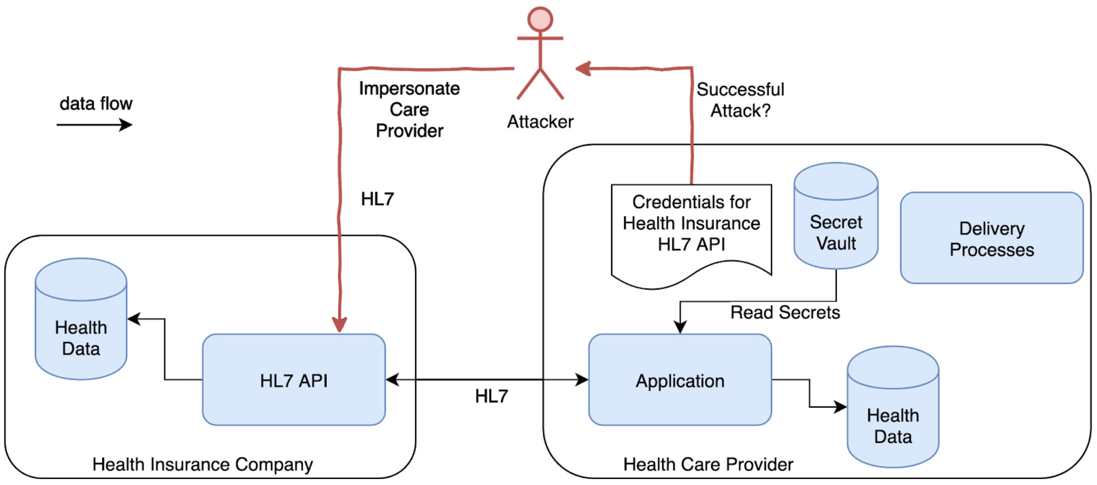

Figure 5: Health Care Integration Scenario Protected by a Secret

Suppose a Health Care Provider has an application that records notes about a visit and submits that patient health care data to a Health Insurance Company. The Provider's application submits the data to the Health Insurance Company's API via the HL7 FHIR R4 protocol[4] after authenticating using the application's user+password credentials (or OAuth api key/JWT token). The application and API are covered by HIPAA because they manage patient health information.

In order to comply with HIPAA, the Provider must assess their risks, analyze their information flows, audit and review security critical actions, and respond to incidents.

A risk in this system is leakage or theft of the Provider's HL7 credentials by an attacker. The Provider's application needs to manage and use the Insurer's API credentials safely, *as do the application's delivery and operational processes*. If an attacker is able to obtain the Provider's credentials for the Insurer's API, the attacker might be able to impersonate the care provider application and steal or modify patient health data with the full privileges of the application.

Is it "reasonable and appropriate" for the Health Care Provider to (try and) detect when its privileged application secrets may be used by unauthorized parties?

These are the "keys to the kingdom" of patient and business data, so detecting unauthorized secret use ought to be reasonable and appropriate. It would also be very helpful if the authenticating party (Insurer) had a mechanism to detect unauthorized use of a credential or send audit events back to the Provider for analysis. Knowing a secret is being used by an unauthorized party in a timely manner could reduce the impact of a breach greatly. The affected provider(s) could rotate the credential or even shut down access entirely to limit the damage and impact to patients.

[4] The HL7 FHIR R4 Protocol described at https://www.hl7.org/implement/standards/product_brief.cfm?product_id=491

However, it is up to the covered Provider to assess their risks, analyze their information flows, audit security critical actions, and respond to these kind of events.

In this scenario, the Provider might establish two detective controls to manage their risk:

1. an Enterprise-wide detective control that audits and analyzes accesses to the internal Secret Vault to detect attackers attempting to steal secrets
2. a specific detective control for this application's integration with the Insurer built on a feed of the HL7 audit events recorded by the Insurer's API for this application's access

Similar threats can be modeled for other organizations that may be subject to HIPAA, PCI, or NIST standards. See Appendix Appendix - Regulated Audit Requirements for a summary of relevant controls.

Many Enterprise secret vaults and Privileged Access Management solutions have audit and threat detection capabilities. However, integrating those vaults with modern application deployment platforms is challenging. The main issues with Enterprise secret vaults are that they:

1. do not integrate with the identity services of Cloud or container platforms such as AWS IAM
2. often model application/service accounts poorly, focusing on people interacting through a session instead

DevOps practitioners and operators need a privileged threat analysis system for Cloud Native secret stores like AWS Parameter Store, Azure Vault, and Hashicorp Vault to help assure the confidentiality of application secrets. QualiMente is looking for customers to guide development of such a system, contact devops@qualimente.com if you are interested.

Conclusions

This research was conducted and presented to help you understand and address risks that have always been present in application delivery processes. These risks grow and become more challenging as organizations automate their application delivery pipelines, decompose large applications into many smaller ones, and adopt additional compute platforms.

The key hypotheses of this research have been confirmed or practically so:

1. Most practitioners are dissatisfied with their application secret delivery processes

Only 30% of DevOps practitioners are Satisfied or 'Very Satisfied' with their application secret delivery processes. Study participants reported a lack of understanding and tools to manage secrets safely with their mix of applications, application platforms, and delivery tools.

2. Most practitioners do not have useful secret usage audit or threat detection mechanisms

Only a quarter of DevOps practitioners reported having tools to audit secret usage or detect unauthorized use. Further, practitioners need new or enhanced tools to help them assure the confidentiality of secrets used by Cloud Native applications.

QualiMente can help you improve your application secret delivery and management processes by:

1. understanding and then assessing where you are at
2. making specific recommendations on how to deliver secrets safely with your mix of tools
3. helping you construct an audit and usage analysis program

Contact devops@qualimente.com for assistance or questions about this research.

Acknowledgements

Every research project of this size depends on the generosity and insights of many people. The principal researcher, Stephen Kuenzli, would like to thank the people who supported this work the most.

Thank you to:

Each of the participants in this study; this research would not have been possible without sharing your perspective and experiences. Every response you provided improved this research and it is appreciated.

The review team; QualiMente team members for providing feedback on initial drafts, Rachel Ruebbelke at VenturPlex who helped me transform the draft report into publishable quality, and the Topple team for improving the effectiveness and impact of this report with their document review and feedback service.

Philip Morgan and my colleagues in The Expertise Incubator; you challenged us to perform original, valuable research and guided us through the process to success.

Others who assisted with the project, but prefer to remain anonymous.

Appendix - Demographics

Demographic data was collected from 13 respondents. Demographic questions were not included in the first version of the questionnaire.

Respondents were asked to identify up to 3 areas of focus:

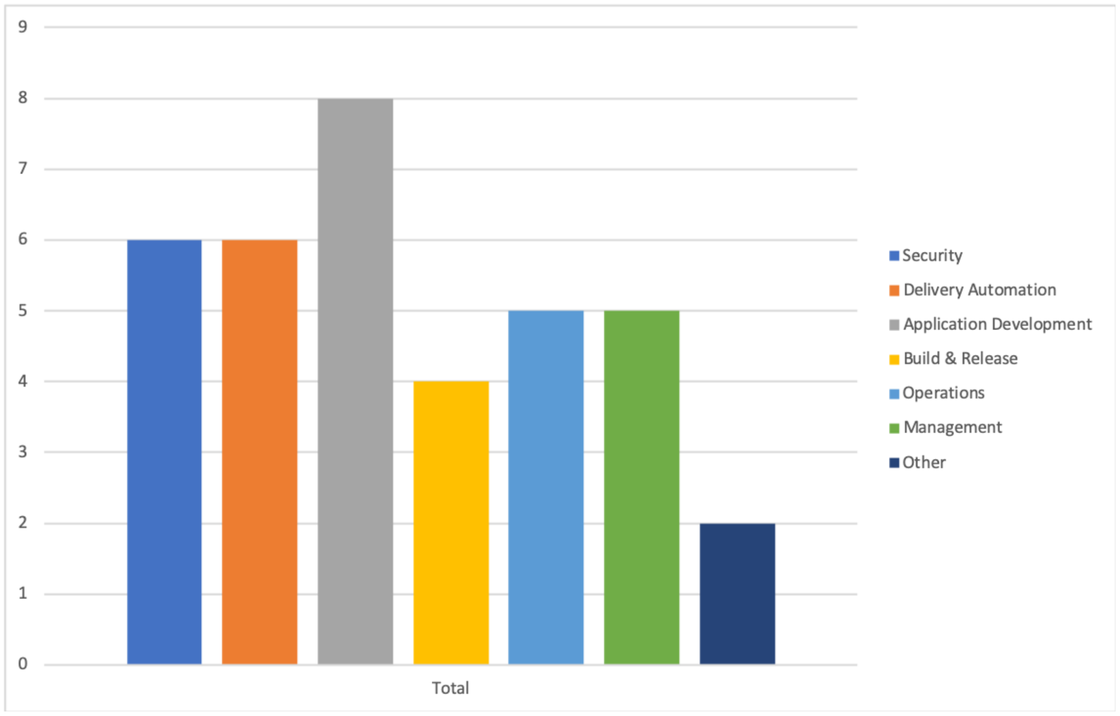

Figure 6: Respondents' Areas of Focus

These areas of focus correspond well with the intended population of DevOps Practitioners for this research.

The same 13 respondents also reported the industry they work in:

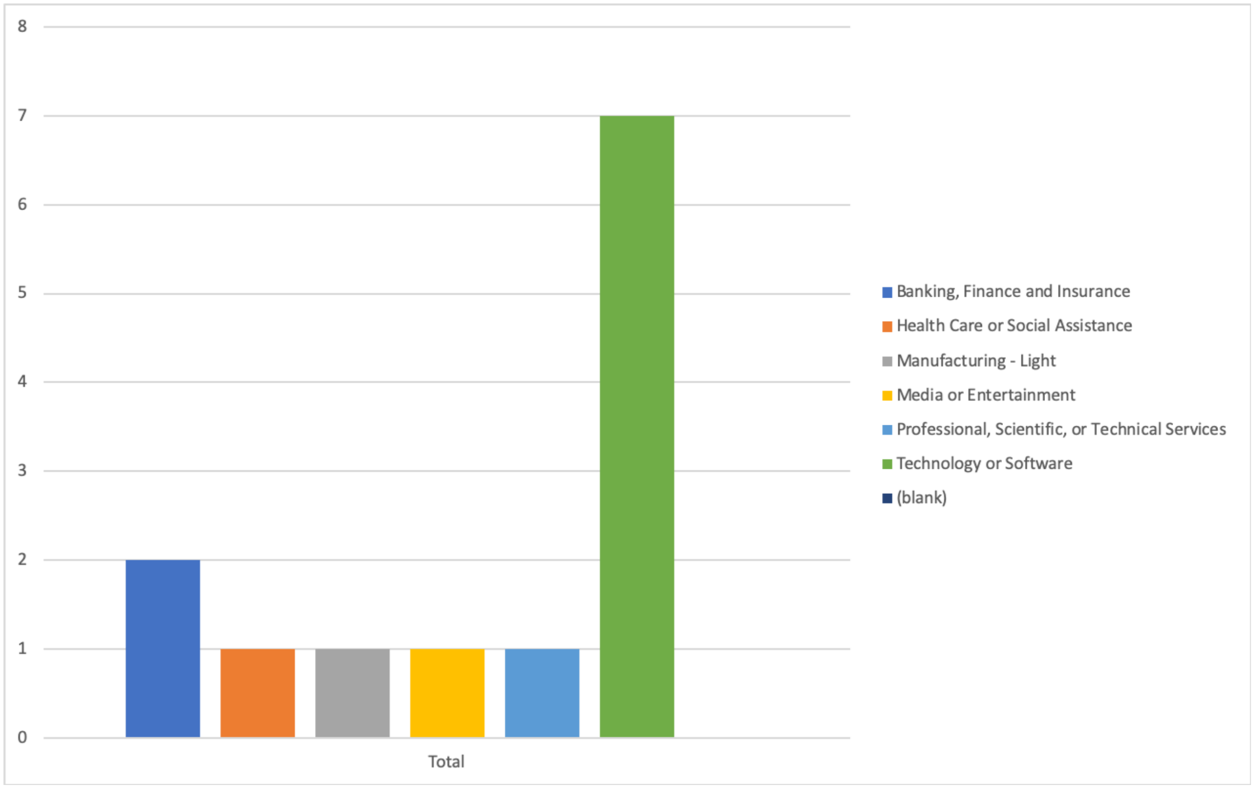

Figure 7: Respondents' Industry

More than half (7/13) of the respondents identified as working in the 'Technology or Software' industry. The reported feelings of satisfaction and risk appear similar to those of the whole population. This is interesting in that:

1. The high representation of 'Technology or Software' does not appear to bias the overall sample.
2. People working in 'Technology or Software' feel the same way as those outside of the industry, even though one might think people in 'Technology or Software' have better practices. The grass is not greener or safer on the other side.

Appendix - Research Methods

The survey was conducted as a convenience, opt-in sample using questionnaires offered by SurveyMonkey. Three versions of the questionnaire were used in total with questions being added in each version. Responses were collected for two months from April 24, 2019 through June 22, 2019. The v3 questionnaire improved the phrasing of a question added in v2. Most questions did not require a response, which is the main reason there inconsistent numbers of responses between questions.

Statistical Analysis of Association Between Satisfaction and Safety
There is likely an association between satisfaction with your secret delivery process and the safety of that process. A Goodman & Kruskal's Gamma Test was conducted to test for independence of satisfaction and safety responses.

The p-value for this test of *independence* between satisfaction and safety is 0.108, meaning we *can* reject the null hypothesis of independence and say that there is an association between the responses at a significance level of 15%, but not at a more conservative level of e.g. 5%. If you're willing to act on an 85% probability that this data is indicative of actual trends, you would say there's a medium-strength association between satisfaction and risk. The Goodman & Kruskal Gamma test statistic is 0.500. The Gamma statistic is a measure of rank correlation and estimates the strength of association of two ordinal factors on a scale of -1 to 1. A rank correlation measures the similarity of the orderings of the data when ranked by each of the quantities.

How to use this information: If someone says they are dissatisfied with the way applications get their secrets, your Spidey-sense should start to tingle. Try to probe into the risk around the delivery process or the mechanisms by which applications read their secrets. There's a decent chance that some very important risks are just around the corner.

Goodman & Kruskal's Gamma
Here is Wikipedia's introduction to Goodman & Kruskal's Gamma:

> In statistics, Goodman and Kruskal's gamma is a measure of rank correlation, i.e., the similarity of the orderings of the data when ranked by each of the quantities. It measures the strength of association of the cross tabulated data when both variables are measured at the ordinal level. It makes no adjustment for either table size or ties. Values range from −1 (100% negative association, or perfect inversion) to +1 (100% positive association, or perfect agreement). A value of zero indicates the absence of association.

Appendix - Regulated Secret Delivery and Audit Requirements

HIPAA
The HIPAA Security Rule establishes national standards to protect individuals' electronic personal health information that is created, received, used, or maintained by a covered entity. The Security Rule requires appropriate administrative, physical and technical safeguards to ensure the confidentiality, integrity, and security of electronic protected health information.

HIPAA's Security Rule contains several rules that apply to application secret delivery practices:

- §164.308(a)(1)(ii) (D) - Implement procedures to regularly review records of information system activity, such as audit logs, access reports, and security incident tracking reports.
- §164.308(a)(5)(ii)(C) - Procedures for monitoring log-in attempts and reporting discrepancies.
- §164.308(a)(5)(ii)(D) - Procedures for creating, changing, and safeguarding passwords.
- §164.308(a)(6)(i) - Implement policies and procedures to address security incidents.
- §164.308(a)(6)(ii) - Identify and respond to suspected or known security incidents; mitigate, to the extent practicable, harmful effects of security incidents that are known to the covered entity or business associate; and document security incidents and their outcomes.
- §164.312(b) - Implement hardware, software, and/or procedural mechanisms that record and examine activity in information systems that contain or use electronic protected health information. Note: Depends on scope defined in §164.308(a)(1)(ii) (D)

PCI DSS
The Payment Card Industry Data Security Standard (PCI DSS) was developed to encourage and enhance cardholder data security and facilitate the broad adoption of consistent data security measures globally. PCI DSS provides a baseline of technical and operational requirements designed to protect account data.

PCI DSS contains several rules that apply to application secret delivery practices:

- 10.1 - Implement audit trails to link all access to system components to each individual user.
- 10.2 - Implement automated audit trails for all system components for reconstructing these events:
    - all individual user accesses to cardholder data;
    - all actions taken by any individual with root or administrative privileges;

    - access to all audit trails;
    - invalid logical access attempts;
    - use of and changes to identification and authentication mechanisms (including creation of new accounts, elevation of privileges),
    - all changes, additions, deletions to accounts with root or administrative privileges;
    - initialization, stopping or pausing of the audit logs;
    - creation and deletion of system-level objects.
- 10.3 - Record audit trail entries for all system components for each event, including at a minimum: user identification, type of event, date and time, success or failure indication, origination of event, and identity or name of affected data, system component or resource.
- 10.6 - Review logs and security events for all system components to identify anomalies or suspicious activity. Perform critical log reviews at least daily.
- 11.5 - Deploy a change-detection mechanism (for example, file-integrity monitoring tools) to alert personnel to unauthorized modification (including changes, additions, and deletions) of critical system files, configuration files, or content files; and configure the software to perform critical file comparisons at least weekly

NIST 800-53
Many organizations use the National Institute of Standards & Technology's (NIST) Special Publication 800-53, Security Controls and Assessment Procedures for Federal Information Systems and Organizations, as guidance for their own security controls. This standard describes two families of controls that apply to application secret delivery practices:

- AC - Access Control
- AU - Audit and Accountability

These families of controls contain P1 ("Implement First") controls that recommend:

- AC-3 - Access Enforcement: "Access control policies (e.g., identity-based policies, role-based policies, control matrices, cryptography) control access between active entities or subjects (i.e., users or processes acting on behalf of users) and passive entities or objects (e.g., devices, files, records, domains) in information systems. In addition to enforcing authorized access at the information system level and recognizing that information systems can host many applications and services in support of organizational missions and business operations, access enforcement mechanisms can also be employed at the application and service level to provide increased information security."
- AC-6 - Principal of Least Privilege: "The principle of least privilege is also applied to information system processes, ensuring that the processes operate at privilege levels no higher than necessary to accomplish required organizational missions/business functions."
- AU-2 - Audit Events: "Organizations identify audit events as those events which are significant and relevant to the security of information systems and the environments in which those systems operate in order to meet specific and ongoing audit needs. Audit

events can include, for example, password changes, failed logons, or failed accesses related to information systems, administrative privilege usage, PIV credential usage, or third-party credential usage."

- AU-6 - Audit Review, Analysis, and Reporting: "Audit review, analysis, and reporting covers information security-related auditing performed by organizations including, for example, auditing that results from monitoring of account usage, remote access, [...], physical access, [...], communications at the information system boundaries, [...]. Findings can be reported to organizational entities that include, for example, incident response team, help desk, information security group/department."

As you might expect, the access control family stipulates that both people and application & system processes operate with access controls configured to permit actions in accordance with the Principle of Least Privilege (AC-6) and deny the rest (AC-3).

The audit family requires that access control enforcement mechanism generate audit events (AU-2) and that those events are reviewed and the results reported to the security function within an organization.

In particular AU-6 (1) requires: "The organization employs automated mechanisms to integrate audit review, analysis, and reporting processes to support organizational processes for investigation and response to suspicious activities."