Autopayment Security Hardening

Midtrans

Michael Cahyadi | COO Analyst
akademis.id | 2021

This presentation will dive into the security hardening practices of payment service providers

Source : Adventure Time (Comic) Issue 13, Page 17

Opinion : Compared to the Competition, Midtrans Sucks

Reasons(1)

+ Non-existent hardening guidelines for PCI DSS or SOC 2
+ Core API Guidelines only tells how integration works, does not explain how the core system interacts with ours
+ No option to generate multiple API keys based on permissions
+ No SSL/TLS Enforcement
+ No support for content-security-policy

1. Compared to other payment providers such as Dokupay and Xendit

Content Security Policy

The HTTP **`Content-Security-Policy`** response header allows web site administrators to control resources the user agent is allowed to load for a given page. With a few exceptions, policies mostly involve specifying server origins and script endpoints. This helps guard against cross-site scripting attacks (XSS).

Usage Example

Administrator wants to allow content from a midtrans's domain and all its subdomains

```
Content-Security-Policy: default-src 'self' midtrans.com *.midtrans.com
```

HTTPS Enforcement on Midtrans API

```
Content-Security-Policy: default-src https://api.midtrans.com
```

(Mock) PCI DSS Compliance Through SAQ D

The **Payment Card Industry Data Security Standard** (PCI DSS) is a set of requirements intended to ensure that all companies that process, store, or transmit credit card information maintain a secure environment. Even if we cant obtain a PCI DSS certification, it's always good to use the Self Assessment Questionnaire (SAQ) to evaluate our security practices against industry standards.

(Mock) PCI DSS Compliance Through SAQ D (cont'd)

Examples of merchant environments that would use SAQ D may include but are not limited to:

+ E-commerce merchants who accept cardholder data on their website.
+ Merchants with electronic storage of cardholder data
+ All payment acceptance and processing are entirely outsourced to PCI DSS validated third-party service providers

Encryption at Transit (aka SSL/TLS Enforcement)

TLS refers to the process of securely transmitting data between the client—the app or browser that your customer is using—and your server. This was originally performed using the SSL (Secure Sockets Layer) protocol. However, this is outdated and no longer secure, and has been replaced by TLS. The term SSL continues to be used colloquially when referring to TLS and its function to protect transmitted data.

Payment pages must make use of a modern version of TLS (e.g., TLS 1.2) as it significantly reduces the risk of you or your customers being exposed to a man-in-the-middle attack. TLS attempts to accomplish the following:

- Encrypt and verify the integrity of traffic between the client and your server
- Verify that the client is communicating with the correct server. In practice, this usually means verifying that the owner of the domain and the owner of the server are the same entity. This helps prevent man-in-the-middle attacks. Without it, there's no guarantee that you're encrypting traffic to the right recipient.

Encryption at Transit (cont’d)

Even though `midtrans.min.js` and `snap.js` is served only through HTTPS, Midtrans does not enforce its customers to implement SSL/TLS on their own website.

Encryption at Transit (cont’d)

A *digital certificate*—a file issued by a *certification authority* (CA)—is needed in order to use TLS. When installed, this certificate assures the client that it’s really communicating with the server it expects to be talking to, not an impostor. You should get a digital certificate from a reputable certificate provider, such as:

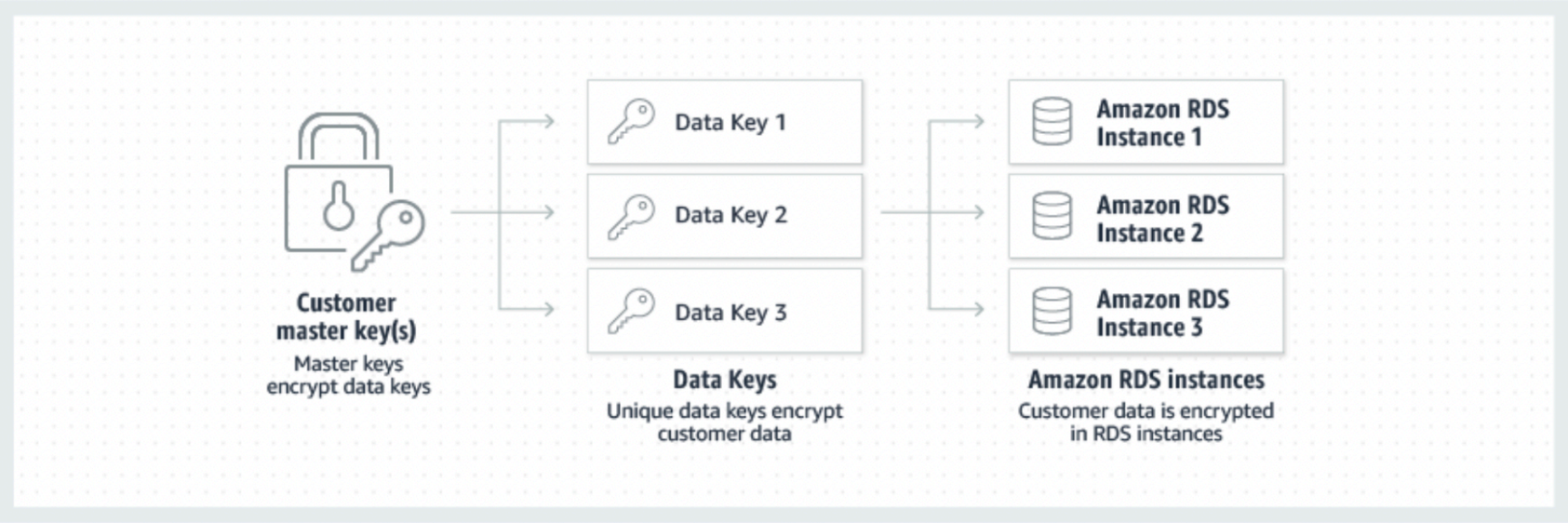
- Let’s Encrypt
- DigiCert
- NameCheap

Certificates can vary in cost, depending on the type of certificate and provider, however Let’s Encrypt is a certificate authority that provides certificates for free.

Encryption of Data at Rest (ft. AWS)

Amazon RDS encrypts your databases using keys you manage with the AWS Key Management Service (KMS). On a database instance running with Amazon RDS encryption, data stored at rest in the underlying storage is encrypted, as are its automated backups, read replicas, and snapshots. RDS encryption uses the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your RDS instance.

Amazon RDS also supports Transparent Data Encryption (TDE) for SQL Server (SQL Server Enterprise Edition) and Oracle (Oracle Advanced Security option in Oracle Enterprise Edition). With TDE, the database server automatically encrypts data before it is written to storage and automatically decrypts data when it is read from storage.Transparent Data Encryption in Oracle is integrated with AWS CloudHSM, which allows you to securely generate, store, and manage your cryptographic keys in single-tenant Hardware Security Module (HSM) appliances within the AWS cloud.



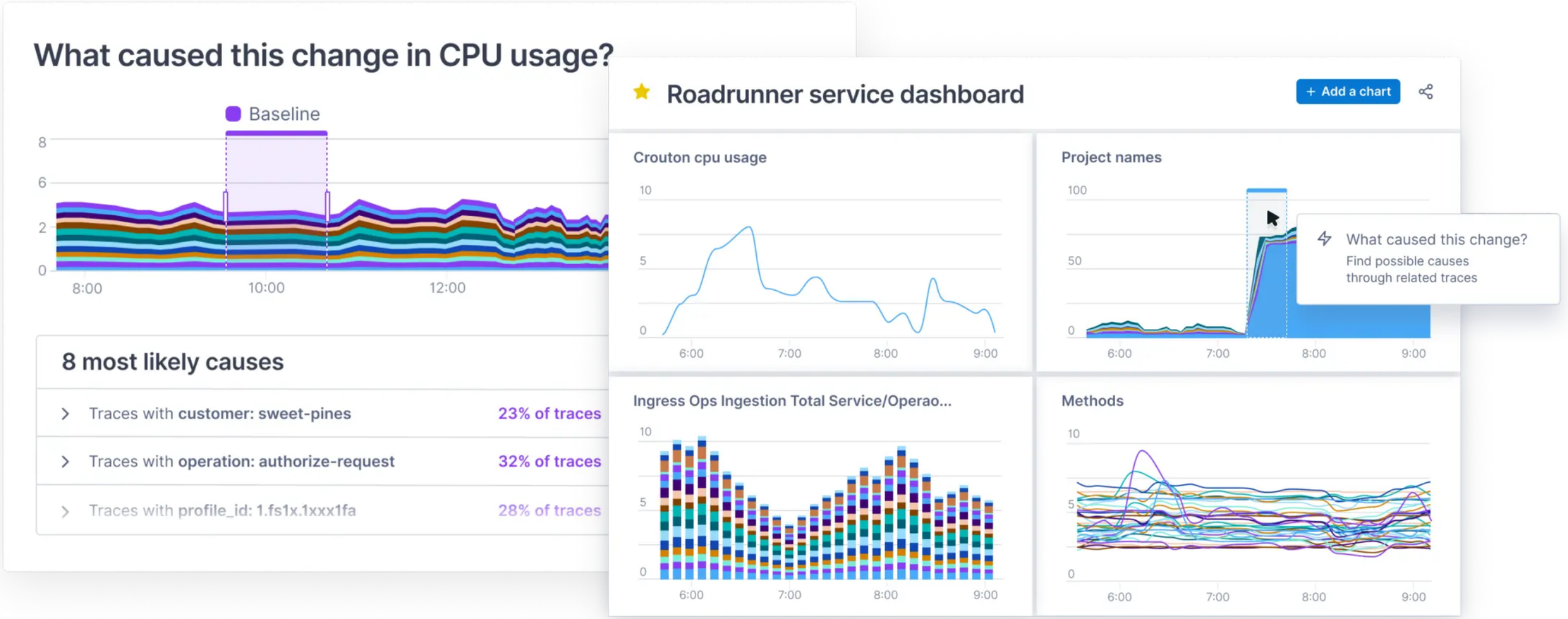
Perform a System-wide Audit with SIEM tools

Security Information and Event Management (SIEM) is a **set of tools and services** offering a holistic view of an organization's information security. SIEM tools provide: Real-time visibility across an organization's information security systems. Event log management that consolidates data from numerous sources.

Common SIEM tools

+ Auditd
+ **Cmdwatch** <- Recommended
+ Elastic Security (ELK Stack)

Instrumentation



Instrumentation will allow us to gain more visibility towards our infrastructure and can help us pinpoint data leaks and performance issues, a common (and recommended) tool for this is **Lightstep**

Additional : A ToS and Privacy Policy

A **privacy policy** and **Terms of Service** are one of the most important documents on any website. It details your company's views and procedures on the information collected from visitors. This is also a requirement for PCI DSS Compliance. Customer should have information about how their data is being treated, especially when we are saving sensitive information such as **payment information of minors**.

https://policies.google.com/privacy and https://policies.google.com/terms are good examples of transparent and accessible privacy policies and terms of service, you can **definitely copy them.**

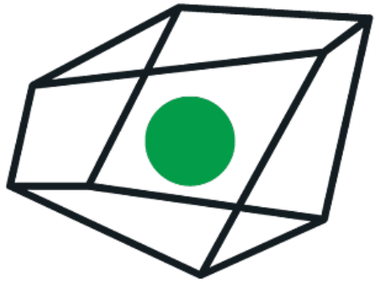
In Conclusion

Its recommended that we

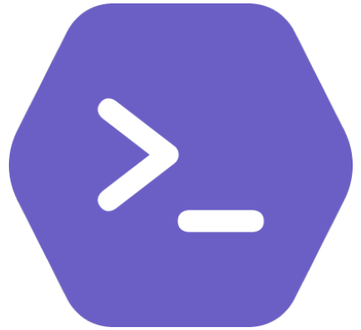
+ Secure all data transit using SSL/TLS
+ Perform a PCI DSS Attestation using PCI SAQ D
+ Perform a security audit on all of our systems to ensure security compliance to industry standards
+ Begin instrumenting our applications to gain better visibility towards security threats not just in the autopayment module, but on the whole system
+ We should start outlining the legal rights of customers and tell them how we process their data

We should also begin to use these applications (both of these have free versions that has all the features we need)

Instrumentation and Performance Monitoring



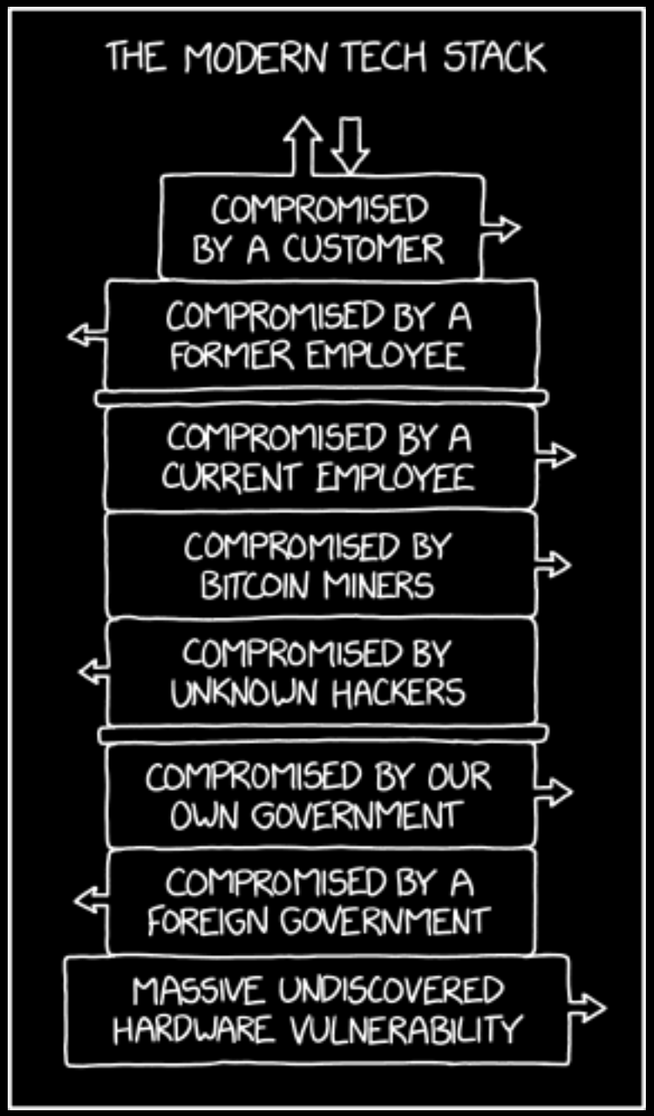
Security and Access Control



SYN ACK? Fin.

Further Reading

1. https://aws.amazon.com/rds/features/security/
2. https://www.pcisecuritystandards.org/documents/SAQ_D_v3.pdf
3. https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP
4. https://stripe.com/docs/security/guide
5. https://www.xendit.co/en/
6. https://cmd.com/wp-content/uploads/2019/12/Cmd-for-Cloud-Compliance-and-Auditing.pdf
7. https://docs.midtrans.com/en/core-api **<- this sucks**



Source : https://xkcd.com/2166/