International Journal of Management, IT & Engineering
Vol. 16 Issue 01, January 2026,
ISSN: 2249-0558 Impact Factor: 7.119
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gate as well as in Cabell's Directories of Publishing Opportunities, U.S.A

Process for Identifying and Securing Data Compoents of Application

Veerbhadra Magdum

Abstract

This article examines Data at Rest (DAR) and Data in Transit (DIT) security components, and controls around those components in a web application environment. Traditional policies around these controls look at all the applications and systems with a one solution fits all, kind of an approach while devising policies around DAR and DIT. Modern applications are getting increasingly complex and use distinctive security components and security methods to protect DAR and DIT. This complexity poses risk that agnostic and abstract AppDevSecOps process falls short to identify true attack surface, and risk posed by such distinctive security components and methods. The article has identified the distinctive security components encompassing the attack surface in detail for Data under consideration. The paper also explains actual experiences on how CICD feedback process between application teams, vulnerability monitoring tools, and policy makers in a AppDevSecOps scenario helped improve the security posture of involved applications. The paper has further built a set of security guidelines to apply to the DAR/DIT security components and to the tools involved in the AppDevSecOps process.

Keywords:

Application Security;
SecOps; DevSecOps;
AppDevSecOps;
Data at Rest;
Data in Transit;
DAR;
DIT.

Author correspondence:

Veerbhadra Magdum, CCSP, FRM,
Email: v_magdum@yahoo.com

1. Introduction

1.1 Importance of Data Security

As per the Verizon data breach investigation report, on page 17, published in 2025, credentials are not just the old fashioned user id and password but they can be various types of secrets that can expose organizations data for mass leakage and such incidents could require a considerably large time to remediate[1].

Given the magnitude of such events, many professionals across various industries dedicate considerable efforts and time to comply with information security policies set forward by the enterprise security teams. One of the policies consistently monitored in recent times is around securing the data that is at rest or in transit or in use. This policy requires the data processed in the application and all the channels used to access this data to be threat proof. This entails configuring the recommended data encryption methods, usage of vaults and using secure protocols of communication and secrets across the endpoints. Any lapses in securing these components or endpoints will be risk prone and would require security checks and practices in place.

1.2 Objective and Research Methodology

This article will detail out the various types of secrets and components used to protect data that is at rest or in transit or in use. Also, the article describes few real experiences of AppDev team wherein policy makers have built policies to protect only critical secrets or endpoints of data but are not aware of other secrets or endpoints which create a hidden risk. Such real experience references are used in this article to suggest a AppDevSecOps process in which feedback loop is recommended between app dev teams, policy makers, policy implementers so that all the vulnerable secrets, endpoints or components are identified and protected. The article also lists the recommendations and guidelines to identify and protect all such secrets, endpoints or components reducing the chances of any breach.

2. Components, challenges with DevSecOps and recommended improvements

The application under review is a three-tiered web application - comprising application, web and database components, with each tier serving the application business logic, application UI and the store of structured or unstructured data, respectively.

2.1 Components and gaps in Data at Rest (DAR) DevSecOps process

Following is the discussion of all the places that can have data at Rest (DAR), in the application under consideration. Also discussion includes real experiences for some of the pitfalls observed in the AppDevSecOps process and the practices that should be followed to strengthen AppDevSecOps process

2.1.1 Secrets or Passwords of critical accounts

Sertup and components - Every tier of the application has an entry point through the operating system service account or a database account which again can open doors within to multiple internal accounts. These service accounts or database accounts are considered critical as any breach to these accounts access can lead to a risk of DAR breach and can open the entire IT infrastructure for a more severe form of malware. Given the importance of these accounts, policy makers want to ensure that access to these accounts is secure. Following is a brief introduction and listing of critical components involved in securing access to such service accounts.

Plaintext - The input text to a cipher suite that is converted into a secure cipher text. For e.g. Applications service account password in the current case of discussion.

Cipher suite - An algorithm or a set of rules for performing encryption or decryption of the plaintext passed as input.

Key – Additional randomized salt that a cipher suite employs in combination with the plaintext while performing encryption or decryption.

Ciphertext - The text encrypted with the use of a cipher suite and a key, that cannot be transformed back to its original plaintext form without the usage of decryption.

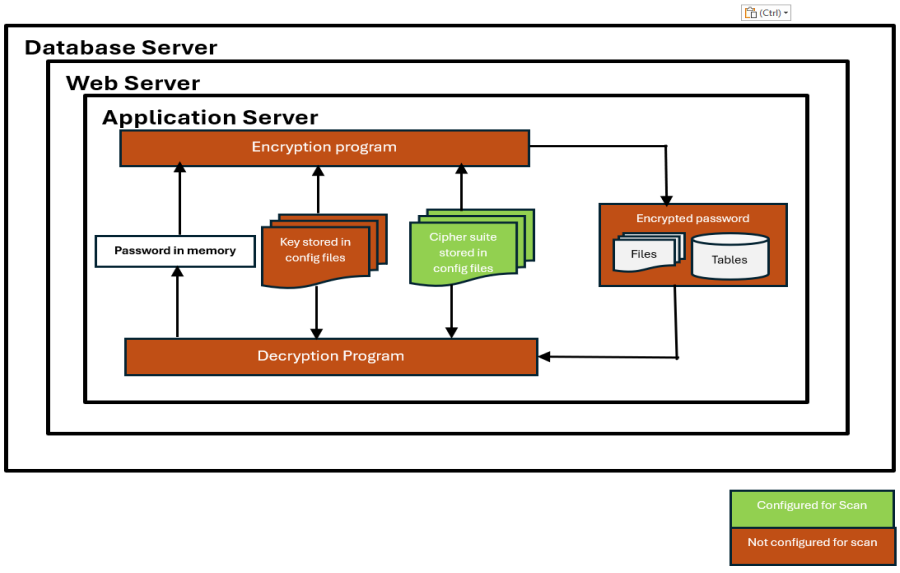
Encryption - The process of transforming the input plaintext to the ciphertext with the use of a cipher and a key.

Decryption - The reverse process of transforming the ciphertext to the plaintext with the use of a cipher suite and a key.

Our experience and the observed risk - Diagram 1 below depicts the deployment of components used to secure passwords or secrets in an application. Note that in this csae, the keys used are stored in the proprietary encryption or decryption program locally on an application server. Also, the configuration files maintain list of compatible encryption standards/cipher suites on the application server locally.

The diagram is having a single component marked in green whereas other components are marked in orange for a purpose of explanation of real experience indicating pitfalls in the AppDevSecOps process used. In this experience AppDev team is provided with a vulnerability report and AppDev team discovered that the scan was not configured to investigate many of the secrets generation components that are highlighted in orange in Diagram 1. Instead, policy and focus seems to center only on the cipher suites used in password encryption. As a result, many components are not fully scanned or investigated, leaving a large proportion of the application effectively unreviewed.

Diagram 1: Application critical components for secrecy



The initial scan report did, at least, reveal that it was needed to update the cipher suites to the latest available versions. The AppDev team was tasked with doing this – but, on investigating more deeply, AppDev team made the following discoveries about those other, unreviewed components:

- Cipher suites were listed in multiple places in the configuration files on the operating system of multiple server layers. Consequently, there was no single place or easy way to update them. This complexity required considerable efforts to fully test our resolution.
- The keys used to generate encrypted strings for the input passwords were hardcoded as clear texts in multiple places, also in the configuration files for the operating system of multiple server layers. Such poor key management could all-too-easily result in an application data breach, requiring AppDev team to resolve this too.
- The encryption/decryption programs, and the configuration files for keys and cipher suites, all had an elevated application account permissions, instead of being restricted to execute privileges only.
- Some instances of cleartext passwords were hardcoded and didn't use the intended encryption/decryption methods.
- The application supported symmetric encryption, which – due to the other shortcomings – could have enabled intruders to construct passwords.

With this experience it is recommended that every component in the diagram should be secured and it is suggested that the scanning tools should include a key management review, encryption/decryption program reviews, privilege review of critical components, and data access through API calls instead of hardcoded keys or secrets.

2.1.2 Sensitive Personal Information stored in the database

Vendor usually works with customers admin users to identify PI/SPI elements in the database and mutually decide on methods to securely store such data. Following are the methods generally used and also discussed are real experiences encountered while implementing such controls -

- ***Data obfuscation*** –Databases provide built in functions or APIs to obfuscate data which is kind of replacing actual PI data elements with some randomly generated data of similar format as that of underlying PI elements. Real experiences encountered indicate that this generally works well for non production environments but production data is expected to be available to the business users as is which can still be a data security risk.
- ***Data encryption*** – Use keys, ciphers and custom encryption/decryption routines to generate encrypted strings to replace PI elements. Real experience observed with keys is listed in previous section and additionally this becomes a overhead to the process and increases surface attack.
- ***Data tokenization*** – Use tokens of data to replace PI data elements. This method is generally used for limited regulatory use cases and use of tokens does increase surface attack.

2.1.3 Sensitive data files on infrastructure servers

Applications as mentioned earlier may have 3 tiers architecture and 3 separate servers for each tier - application, web, database. Any sensitive files stored on the servers should be protected with following methods-

Least privilege and need to know principles – the file permissions and locations should be such that only people or groups of people requiring access to such data are permitted to access.

Password protection of files – Files should be password protected with password rotated periodically and shared with only permitted people.

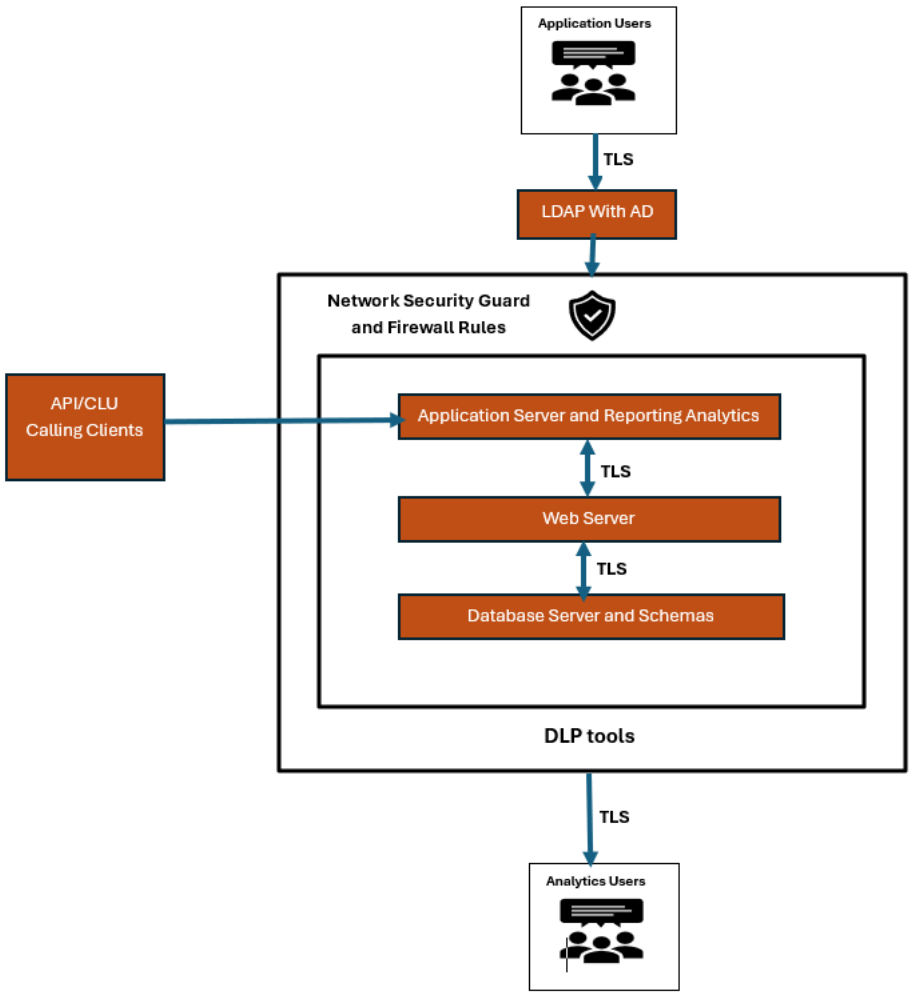
Secure data access through application – Application built role based access allowing access to such data to only permitted people.

The actual experience encountered with security of such objects is that object numbers turn out to be huge increasing surface attack and also deep into the application that are typically not uncovered by the policy makers and implementers without AppDev teams knowledge.

2.2 Components and gaps in Data in Transit (DIT) DevSecOps process

Diagram 2 below depicts the application components or endpoints that require security considerations for Data in Transit (DIT). Subsequent sub sections include discussion on these components with real experiences for some of the pitfalls observed in the DevSecOps process and the discussion includes the practices that should be followed to strengthen the DevSecOps process.

Diagram 2: Data in Transit (DIT) Endpoints



2.2.1 Network security and firewall rules

Sertup and components - Application under review hosts frontend services including services required for data processing on application server, whereas application backend is hosted on a web server. The database is hosted on another server in a three tier architecture. Connectivity and the data movement across these hosts is configurable through the application configuration files and requires specific list of ports to be opened for communication. It is ensured that only these ports are open for communication between all components and all other ports including default ports are blocked for any traffic. Additionally, the servers are protected by enterprise specific firewall rules and other secure tools or mechanisms, that organization requires by policy to intercept and block internal and external unwanted traffic.

Our experience and the observed risk - In the application used, it is observed that the application contains many services that interact with each other behind the scene and implicitly require multiple ports to be open. This implicit requirement if not identified and not mitigated can lead to a security threat. The AppDev teams started to collaborate witb SecOps teams to establish a procedure to monitor traffic on all the ports used by such services, to document all the ports requirements and to periodically audit and recertify firwall rules configured for these ports.

2.2.2 TLS / https protocols and associated certificates

Setup and components - Further the DIT between the servers in the application provide following specific configuration options-

- Application under consideration has an UI that is rendered by the web container which provisions LDAP based authentication via AD and Role Based Access Controls for authorization. The LDAP integration is done using TLS protection to the traffic over TLS port and using TLS / https URL.
- The protected web container resources are configured to use TLS port and TLS / https

URL. This ensures that all the data that is exchanged across all servers and with users is always encrypted and tamper proof.

- Application configuration used a Oracle database schema that hosts the setup and configuration of applications and another schema that hosts the customer transactions data. Both schemas are configured with the roles and privileges such that those are restricted to the need to know and least required levels. Also, the database client's connection URL is configured over a listener port that allows TLS connection only.
- The TLS configuration discussed above requires use of certificates for supporting key exchanges. The certificates are ensured to be within their validity period and are following standard formats supported by latest security standards.

Our experience and the observed risk - One of the pitfall observed by AppDev team is that some users used non SSL ports for client to database connections even though server to server communicatins are secured to only use SSL ports. Such practice is decided to be a risk as it could open the data to security threats and is resolved in time. Similar e.g. was observed in recent exploitation of Oracle's E-business suite of applications with CVE-2025-61884 which is found to be remotely exploitable that allowed unauthenticated attacker with network access via HTTP [2]. The attack exposed critical data of multiple customers and signifies importance of the controls mentioned above.

2.2.3 API or Command Line Utilities requiring secrets

Setup and components - Application in this scenario provides many command line utilities for e.g. object migrations across environments, external batch scheduler interface, etc. These utilities require a connection through an application native account with roles to perform the underlying tasks of object migration or batch executions.

Our experience and the observed risk - As shown in diagram 2 these utilities or APIs are usually called from client tools offered by different vendors and managed by teams other than application teams. Communication between these client tools and the application can lead to a risk and AppDevSecOps teams are suggested to follow TLS standards and to follow all the DIT standards for sharing secrets across all the application interfacing tools.

2.2.4 User Downloads of Sensitive Data

Setup and components - The application in this scenario comes integrated with tools for analytics reporting. It is ensured that reporting tools followed similar DAR/DIT standards.

Our experience and the observed risk - The users having access to these reports can potentially download the data to their desktops and pose the risk of data exposure. The access of such users is suggested to be configured with need to know and least privilege principles. Also, appropriate audit and network security rules are suggested to be applied across the network boundaries allowing the data to cross, and also it is suggested to use data loss prevention tools.

3. Recommendations, best practices and process improvements

3.1 Recommendations and best practices

The process of controlling applications entry point, network interfaces and the associated service accounts/identities access, involves many implicit or explicit components. These are crucial for secure DAR/DIT channels of communication. With the advent of advanced computing methods these critical components become more vulnerable and the application security will largely depend on how securely these components are stored, access protected and managed. Here are some recommendations and best practices to protect these components -

- Configure scanning tools to provide the most granular information about the vulnerabilities so that the application teams can isolate the attack surface. For e.g. configure the scan for operating systems file paths and database tables storing vulnerable configuration of keys, cipher suites, encryption programs.
- Maintain inventory of critical components like cipher suites used, keys used, and methods used to secure keys or passwords and ensure they follow requisite security standards or policies. These items should be periodically updated to follow latest standards.
- Parameterize or use a common and secure store for all the critical components instead of repeating configuration in multiple places and tiers of applications that require usage of these components.
- As an additional layer of security encrypt all key configuration files or tables where the critical components are configured.
- Privileges to these critical components should be very restrictive and should be managed by the super users and should not be accessible

with full permissions to the application accounts or non-privileged accounts.

- Where possible or supported rely on public/private key or asymmetric methods instead of symmetric methods.
- Use stronger and latest encryption methods that application supports.
- Use certificates that are valid and are not expired and implement secure and latest formats.
- Use wallets or vaults for storing these critical components and use the application supported API interfaces with these wallets or vaults. This will help in automating periodic rotation of these critical components in a secure way and reduce the vulnerability attack.
- Remove default ports and default passwords configuration of admin accounts from all places in application. Implement a password policy for all user accounts and service accounts.
- As an additional oversight, firewall rules should be reviewed periodically and certified by system owners.

These best practices and recommendations could be used for multiple applications by the information security team which will help them tighten the organization's security practices.

3.2 Process improvements

Here are the recommendations to impve AppDevSecOps process that are based on the lessons learned –

- Employ a template so that multiple applications can identify and implement such critical components in a homogeneous way.
- Hold peer reviews at every major milestone of the application lifecycle, so that awareness and compliance becomes the focus of the application teams.
- Publish periodic scans of each application's critical components with granular details, assign resolutions to respective teams and get feedback from the application teams for improvements in the vulnerability identification and remediation process.
- Train application teams on the scanning processes and understand the parameters of application security posture. This softens the resistance of AppDev teams by helping them to understand that identifying and mitigating security risk is a necessity rather than a burden and can provide valuable learning experiences.

4. Conclusion

The entire exercise helped us identify following listed components for an applications data security - **Data at Rest (DAR)** involves following components in addition to generally considered encryption standards

- Application data stored within database schemas,
- Application server's sensitive configuration files,
- Web server's sensitive configuration files.
- Data encryption standards, cipher suites, encryption/decryption keys and programs, storage of these components,
- API's and Command line utilities.

Data in Transit (DIT) involves following components –

- Client tools TLS or SSL connectivity to the database schemas,
- Infrastructure server's SSL connectivity to the database schemas,
- Infrastructure servers (application, web, database) to and from traffic using SSH or TLS,
- Network security and firewall rules
- Certificates associated with these components

Also, the experiences shared in this article can help others build a more robust vulnerability management process by using deeper insights by AppDev SME's, and an increased collaboration with policy makers and rest of the SecOps teams. The recommendations and guidelines derived from the lessons learned are listed in this article as well and we hope that others will benefit from our experiences.

References

[1] "Verizon 2025 Data Breach Investigations Report", Page 17
[2] https://www.bleepingcomputer.com/news/security/oracle-silently-fixes-zero-day-exploit-leaked-by-shinyhunters/, https://nvd.nist.gov/vuln/detail/CVE-2025-61884