For the purposes of this report, I will be exploring the topic of Database Security from the perspective of an Application Developer, Database Administrator and Security Engineer. In doing so I will discuss three topics from the OWASP Cheat Sheet series: Database Security, SQL Injection Attack Prevention and User Privacy Protection.

First, with regards to Database Security, the OWASP recommendations focus primarily on network traffic encryption and database hardening which are done using myriad of techniques and procedures that are generally applicable to most databases but vary in their specifics according to the type of database being used. What is applicable to all types of databases is the need to secure TCP and the need to restrict access in accordance with the principle of least privilege.

On the topic of network security, one option that is available to many DBAs that can contribute positively to database security is disabling network access altogether, instead routing all access over a socket file or named pipe, which are two alternative ways of performing inter-process communications that don’t require network connectivity.

***Note:*** *named pipes in Linux can work using the mkfifo command which creates a queue data structure as a file and the echo > commands on one side and the cat <> commands on the other side. Both must be engaged in order for the transfer to work as expected. Named pipes should be created with a secured Access Control List (ACL).*

Alternatively configuring the database to only allow connections that are encrypted with modern ciphers can also help to protect the database connection to the network. This can help to mitigate the threat of an attacker collecting network packets in transit. Another way to mitigate against this threat is to configure the database such that it only binds to the local host. This helps ensure that the acceptable avenues of access are not circumvented by an attacker trying to access the database directly on a client machine within the Local Area Network (LAN). Furthermore, restricting access to the network port to only specific hosts with firewall rules can also help to protect against this threat by allowing only authorized database users to connect from devices that have been vetted and deemed secure. While these measures are all important contributors individually, when taken together they can have a compounding holistic effect that creates a more secure networking environment for the database to perform its work. This concept is known as a layered defense or defense in depth and these methods are by no means exhaustive of a defense in depth approach.

In parallel to the application of these techniques it is also important to ensure that authentication and authorization are dealt with in a conservative manner, using Windows authentication mode over Mixed Authentication Mode whenever possible. For users that do have access as authorized users it is important to be specific in determining exactly how much access is given using roles that correspond to their respective job responsibilities. These decisions should always be made in accordance with the principle of least privilege such that a user is given the rights needed to perform their duties and only those rights. One thing to keep in mind is that most applications only need SELECT, UPDATE and DELETE permissions to a database to perform their respective duties. Not only does restricting access help to protect against the potential for malicious use by a disgruntled employee, but it also helps to protect against compromised accounts that are used by bad actors to access sensitive information or to pivot in such a way as to further elevate their level of access, potentially engaging in IP theft or causing damage to the company’s core infrastructure such as by tampering with or deleting a database.

The credentials used by authorized users to log in should meet a specified security threshold in terms of length and complexity and passwords should be subject to change and review in recurring intervals, thus reducing the window of time that an attacker may potentially have to crack or guess a user’s password. Furthermore, authorized accounts should also be restricted to connect only with specific databases and on whitelisted hosts so that the account is not exposed to the potential threats that may exist outside the scope of the corporate network. In addition to restricting account access, it is also important to restrict access to configuration files by using file encryption and storing transaction logs on a separate drive from the main database files. Furthermore, no matter how many precautionary measures are taken it is important to remember that there is always a chance that something was overlooked, or something was done incorrectly as a result of human error. Therefore, it is also important to ensure to keep regular backups on a frequent basis and that said backups are encrypted and protected with the appropriate permission settings.

In relation to general Database Security, SQL Injection attacks are a unique type of database risk that warrants its own discussion. These types of attacks occur when a malicious user of an application inserts data into a form such that they can circumvent the application logic and send an SQL query to the backend database that might either modify, retrieve or delete otherwise protected data. SQL Injections are unfortunately very common but are relatively simple to mitigate against.

To avoid SQL Injection vulnerabilities, the goals of the developer should be to overcome the habit of writing dynamic queries, and to prevent user supplied input which contains malicious SQL from affecting the logic of the executed query. The four primary ways of achieving this are: 1) using prepared statements with parameterized queries, 2) using stored procedures, 3) whitelisting input validation, and 4) escaping all user supplied input. However, the effectiveness of these more direct techniques would be additionally supported by performing some general database security as mentioned previously, including enforcing the principle of least privilege as well as performing whitelist input validation, which is the practice of detecting unauthorized input before it is passed to the SQL query. However, input validation cannot account for all cases so it should not be relied on as a primary method.

Finally, the topic of User Privacy Protection is one that spans across several IT sectors. As such I will focus on the most effective guidelines to mitigate against the core vectors used to de-anonymize an individual. First and foremost in the fight to preserve user privacy is the use of strong cryptography. It is important to use strong cryptography to encrypt sensitive data both in transit and at rest. User secrets such as passwords are an example of data in storage or data at rest that might be protected in this manner to mitigate the risk of exposing user credentials. For data in transit, it is important that developers implement the TLS/SSL best practices. As a general rule of thumb private data should be encrypted, and user credentials should be hashed for storage.

Another good measure to take in the fight to protect user privacy would be to implement Support HTTP Strict Transport Security (HSTS), which is a header set by the server that restricts the pool of acceptable connections to only those that use HTTPS. While it may be impractical to force this configuration on all users it is good practice to at least have this as an option that users can choose to enable. This coupled with remote session invalidation would give users a good layer of protection against Man-in-the-Middle (MitM) Attacks and unwanted session monitoring.

In addition to SSL and HSTS, the practice of Digital Certificate Pinning is a useful way of reducing the scope of visibility that a CA might have so that only predefined certificates are used, even when others have been granted access. This is akin to using a firewall restrict the scope of access to a port to only whitelisted servers. This is a good way of controlling what outside parties might have access to your data.

Another way to protect user privacy is to provide connections from your application to anonymous networks and prevent IP leaks. One of the most popular methods used by activists, journalists and whistle-blowers around the world is by using anonymous networks such as the Tor Project or I2P which provide people with a means to circumvent censorship, report injustices and send and receive news that may otherwise be inaccessible from within the confines of oppressive regimes. This comes with its own set of drawbacks however and is not likely to be viable for an enterprise environment due to the drastically reduced network speeds. Although the existence of these networks cannot be relied upon as a privacy silver bullet, developers can make the effort to ensure that their apps are at least compatible with these networks so that people fleeing persecution might be able to communicate to the outside world using social media or other platforms. Having that level of compatibility may not be relevant to most users but helps the overall ecosystem by providing an avenue for reprieve from persecution for publication of state misconduct. Another tool that often goes hand in hand with privacy is the use of a VPN which can help to obfuscate the source IP of individuals and protect their web traffic from unwanted surveillance. Finally with regards to any of these practices honesty should always remain at the forefront of policy decisions. If an application cannot in good faith provide legal and technological protections to their users, they should be informed as such so they can make an informed decision as to whether they choose to continue to use that product.