Threat Modeling Report

Created on 11/7/2018 7:04:31 PM

Threat Model Name:

Owner:

Reviewer:

Contributors:

Description:

Assumptions:

External Dependencies:

Threat Model Summary:

Not Started	0
Not Applicable	0
Needs Investigation	12
Mitigation Implemented	30
Total	42
Total Migrated	0

Diagram: Diagram 1

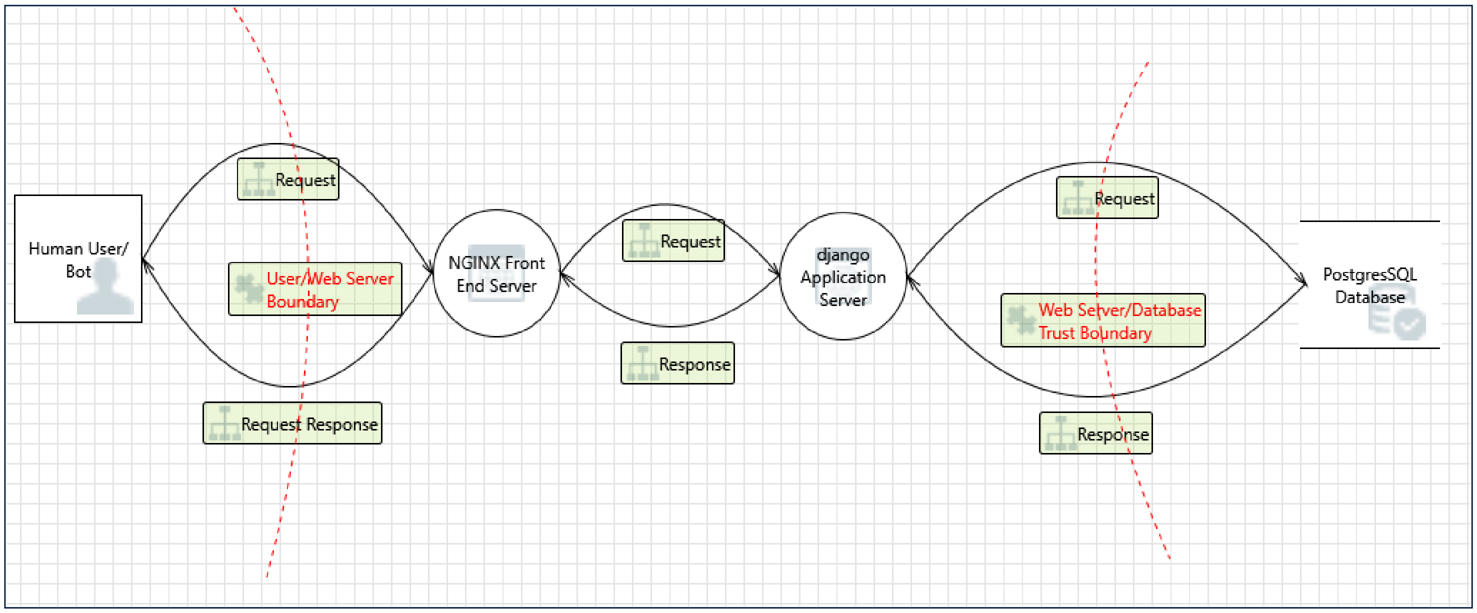
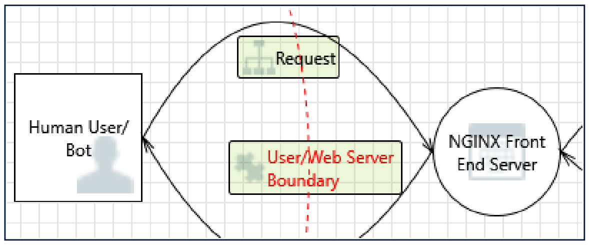


Diagram 1 Diagram Summary:

Not Started	0
Not Applicable	0
Needs Investigation	12
Mitigation Implemented	30
Total	42
Total Migrated	0

Interaction: Request



1. Spoofing the NGINX Front End Server Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing

Description: NGINX Front End Server may be spoofed by an attacker and this may lead to information disclosure by Human User/Bot. Consider using a standard authentication mechanism to identify the destination process.

Justification: Authentication is required for Humans and Bots.

2. Spoofing the Human User/Bot External Entity [State: Needs Investigation] [Priority: High]

Category: Spoofing

Description: Human User/Bot may be spoofed by an attacker and this may lead to unauthorized access to NGINX Front End Server. Consider using a standard authentication mechanism to identify the external entity.

Justification: Authentication is required for Humans and Bots. However, if credentials are compromised, it would still be vulnerable. Need to evaluate MFA.

3. Potential Lack of Input Validation for NGINX Front End Server [State: Mitigation Implemented] [Priority: High]

Category: Tampering

Description: Data flowing across Request may be tampered with by an attacker. This may lead to a denial of service attack against NGINX Front End Server or an elevation of privilege attack against NGINX Front End Server or an information disclosure by NGINX Front End Server. Failure to verify that input is as expected is a root cause of a very large number of exploitable issues. Consider all paths and the way they handle data. Verify that all input is verified for correctness using an approved list input validation approach.

Justification: Traffic is encrypted

4. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category: Tampering

Description: The web server 'NGINX Front End Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.

Justification: Input is santized

5. Potential Data Repudiation by NGINX Front End Server [State: Mitigation Implemented] [Priority: High]

Category: Repudiation

Description: NGINX Front End Server claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

Justification: All transactions are logged

6. Data Flow Sniffing [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure

Description: Data flowing across Request may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.

Justification: HTTPS traffic required

7. Potential Process Crash or Stop for NGINX Front End Server [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

Description: NGINX Front End Server crashes, halts, stops or runs slowly; in all cases violating an availability metric.

Justification: Needs investigation

8. Data Flow Request Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: Needs investigation

9. Elevation Using Impersonation [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: NGINX Front End Server may be able to impersonate the context of Human User/Bot in order to gain additional privilege.
Justification: User Access Controls are Implemented

10. NGINX Front End Server May be Subject to Elevation of Privilege Using Remote Code Execution [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: Human User/Bot may be able to remotely execute code for NGINX Front End Server.
Justification: Command injection is not allowed

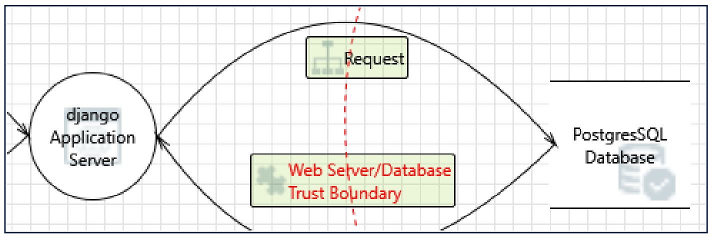
11. Elevation by Changing the Execution Flow in NGINX Front End Server [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into NGINX Front End Server in order to change the flow of program execution within NGINX Front End Server to the attacker's choosing.
Justification: It is only a proxy server and does not process commands

12. Cross Site Request Forgery [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: Cross-site request forgery (CSRF or XSRF) is a type of attack in which an attacker forces a user's browser to make a forged request to a vulnerable site by exploiting an existing trust relationship between the browser and the vulnerable web site. In a simple scenario, a user is logged in to web site A using a cookie as a credential. The other browses to web site B. Web site B returns a page with a hidden form that posts to web site A. Since the browser will carry the user's cookie to web site A, web site B now can take any action on web site A, for example, adding an admin to an account. The attack can be used to exploit any requests that the browser automatically authenticates, e.g. by session cookie, integrated authentication, IP whitelisting, … The attack can be carried out in many ways such as by luring the victim to a site under control of the attacker, getting the user to click a link in a phishing email, or hacking a reputable web site that the victim will visit. The issue can only be resolved on the server side by requiring that all authenticated state-changing requests include an additional piece of secret payload (canary or CSRF token) which is known only to the legitimate web site and the browser and which is protected in transit through SSL/TLS. See the Forgery Protection property on the flow stencil for a list of mitigations.
Justification: Zulip requires CSRF for all API interactions

Interaction: Request



13. Spoofing the django Application Server Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing
Description: django Application Server may be spoofed by an attacker and this may lead to unauthorized access to PostgresSQL Database. Consider using a standard authentication mechanism to identify the source process.
Justification: Zulip user is the only connection allowed to database

14. Spoofing of Destination Data Store PostgresSQL Database [State: Mitigation Implemented] [Priority: High]

Category:	Spoofing
Description:	PostgresSQL Database may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of PostgresSQL Database. Consider using a standard authentication mechanism to identify the destination data store.
Justification:	Zulip user is the only connection allowed to database

15. Potential SQL Injection Vulnerability for PostgresSQL Database [State: Needs Investigation] [Priority: High]

Category:	Tampering
Description:	SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker.
Justification:	Needs investigation

16. The PostgresSQL Database Data Store Could Be Corrupted [State: Mitigation Implemented] [Priority: High]

Category:	Tampering
Description:	Data flowing across Request may be tampered with by an attacker. This may lead to corruption of PostgresSQL Database. Ensure the integrity of the data flow to the data store.
Justification:	Access control is in place

17. Data Store Denies PostgresSQL Database Potentially Writing Data [State: Mitigation Implemented] [Priority: High]

Category:	Repudiation
Description:	PostgresSQL Database claims that it did not write data received from an entity on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Justification:	All transactions are logged

18. Data Flow Sniffing [State: Mitigation Implemented] [Priority: High]

Category:	Information Disclosure
Description:	Data flowing across Request may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.
Justification:	all traffic is encrypted

19. Weak Credential Transit [State: Mitigation Implemented] [Priority: High]

Category:	Information Disclosure
Description:	Credentials on the wire are often subject to sniffing by an attacker. Are the credentials re-usable/re-playable? Are credentials included in a message? For example, sending a zip file with the password in the email. Use strong cryptography for the transmission of credentials. Use the OS libraries if at all possible, and consider cryptographic algorithm agility, rather than hardcoding a choice.
Justification:	Strong passwords are required and all traffic is encrypted

20. Potential Excessive Resource Consumption for django Application Server or PostgresSQL Database [State: Needs Investigation] [Priority: High]

Category:	Denial Of Service
Description:	Does django Application Server or PostgresSQL Database take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout.
Justification:	Needs investigation

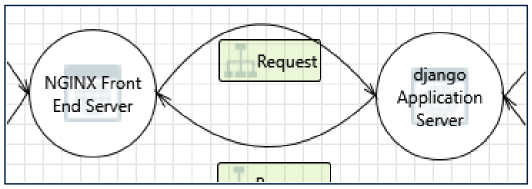
21. Data Flow Request Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category:	Denial Of Service
Description:	An external agent interrupts data flowing across a trust boundary in either direction.
Justification:	Needs investigation for DoS attacks

22. Data Store Inaccessible [State: Needs Investigation] [Priority: High]

Category: Denial Of Service
Description: An external agent prevents access to a data store on the other side of the trust boundary.
Justification: Needs investigation on DoS attacks

Interaction: Request



23. NGINX Front End Server Process Memory Tampered [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: If NGINX Front End Server is given access to memory, such as shared memory or pointers, or is given the ability to control what django Application Server executes (for example, passing back a function pointer.), then NGINX Front End Server can tamper with django Application Server. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.
Justification: Direct memory access is not allowed

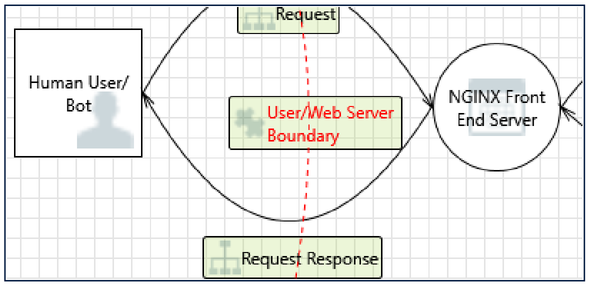
24. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: The web server 'django Application Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: Input is sanitized

25. Elevation Using Impersonation [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: django Application Server may be able to impersonate the context of NGINX Front End Server in order to gain additional privilege.
Justification: Password mitigations and security are enabled to prevent the elevation of accounts

Interaction: Request Response



26. Spoofing of the Human User/Bot External Destination Entity [State: Needs Investigation] [Priority: High]

Category: Spoofing
Description: Human User/Bot may be spoofed by an attacker and this may lead to data being sent to the attacker's target instead of Human User/Bot. Consider using a standard authentication mechanism to identify the external entity.
Justification: Authentication is required for Humans and Bots. However, if credentials are compromised, it would still be vulnerable. Need to evaluate MFA.

27. External Entity Human User/Bot Potentially Denies Receiving Data [State: Mitigation Implemented] [Priority: High]

Category: Repudiation

Description: Human User/Bot claims that it did not receive data from a process on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

Justification: All transactions are logged

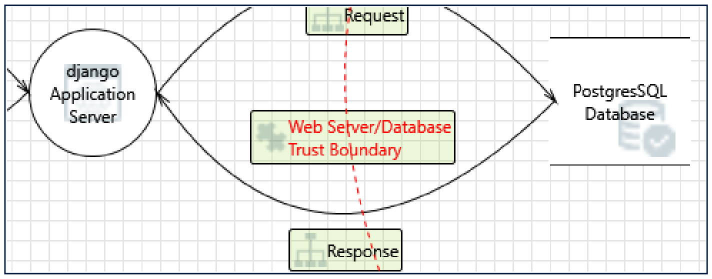
28. Data Flow Request Response Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: Needs investigating

Interaction: Response



29. Spoofing the django Application Server Process [State: Mitigation Implemented] [Priority: High]

Category: Spoofing

Description: django Application Server may be spoofed by an attacker and this may lead to information disclosure by PostgresSQL Database. Consider using a standard authentication mechanism to identify the destination process.

Justification: Zulip user is the only user that can access the database and the traffic is encrypted

30. Spoofing of Source Data Store PostgresSQL Database [State: Mitigation Implemented] [Priority: High]

Category: Spoofing

Description: PostgresSQL Database may be spoofed by an attacker and this may lead to incorrect data delivered to django Application Server. Consider using a standard authentication mechanism to identify the source data store.

Justification: Zulip user is the only user that can access the database and the traffic is encrypted

31. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category: Tampering

Description: The web server 'django Application Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.

Justification: Only trusted input is allowed, otherwise the request fails

32. Persistent Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category: Tampering

Description: The web server 'django Application Server' could be a subject to a persistent cross-site scripting attack because it does not sanitize data store 'PostgresSQL Database' inputs and output.

Justification: Only trusted input is allowed, otherwise the request fails

33. Potential Data Repudiation by django Application Server [State: Mitigation Implemented] [Priority: High]

Category: Repudiation

Description: django Application Server claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

Justification: all transactions are loged

34. Weak Access Control for a Resource [State: Mitigation Implemented] [Priority: High]

Category: Information Disclosure
Description: Improper data protection of PostgresSQL Database can allow an attacker to read information not intended for disclosure. Review authorization settings.
Justification: Access controls are implemented

35. Potential Process Crash or Stop for django Application Server [State: Needs Investigation] [Priority: High]

Category: Denial Of Service
Description: django Application Server crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification: Needs investigation on DoS attacks

36. Data Flow Response Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service
Description: An external agent interrupts data flowing across a trust boundary in either direction.
Justification: Needs investigation of DoS attacks

37. Data Store Inaccessible [State: Needs Investigation] [Priority: High]

Category: Denial Of Service
Description: An external agent prevents access to a data store on the other side of the trust boundary.
Justification: Needs investigation of DoS attacks

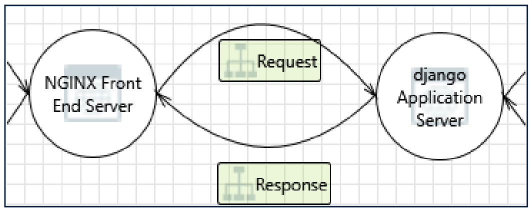
38. django Application Server May be Subject to Elevation of Privilege Using Remote Code Execution [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: PostgresSQL Database may be able to remotely execute code for django Application Server.
Justification: no remote code is available to be executed via HTTPS calls allowed

39. Elevation by Changing the Execution Flow in django Application Server [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: An attacker may pass data into django Application Server in order to change the flow of program execution within django Application Server to the attacker's choosing.
Justification: Access controls are implemented to prevent this

Interaction: Response



40. django Application Server Process Memory Tampered [State: Mitigation Implemented] [Priority: High]

Category: Tampering
Description: If django Application Server is given access to memory, such as shared memory or pointers, or is given the ability to control what NGINX Front End Server executes (for example, passing back a function pointer.), then django Application Server can tamper with NGINX Front End Server. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it.
Justification: no direct access to memory

41. Cross Site Scripting [State: Mitigation Implemented] [Priority: High]

Category: Tampering

Description: The web server 'NGINX Front End Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification: Inputs are sanitized

42. Elevation Using Impersonation [State: Mitigation Implemented] [Priority: High]

Category: Elevation Of Privilege
Description: NGINX Front End Server may be able to impersonate the context of django Application Server in order to gain additional privilege.
Justification: Password mitigations and security are enabled to prevent the elevation of accounts