Aiven Oy Source Code Audit Retest

FINAL REPORT

limitless innovation. no compromise.

Prepared for: James Arlen
CISO

Aiven Oy
Antinkatu 1
001000, Helsinki, Finland

August 03, 2022

Version:	Final
Prepared for:	Aiven Oy
Date:	August 03, 2022

Table of Contents

Executive Summary

Aiven Oy engaged Leviathan Security to perform a time-bound security assessment of the Aiven Postgres Gateway module source code, build configuration, and compiled library. We performed this assessment from July 05, 2022 until July 08, 2022. Retests were conducted on August 03, 2022.

Our objectives were to review the Aiven Postgres Gateway module code, build configuration, and compiled library to identify any malicious or extraneous functionality. We used automated and manual testing informed by highly detailed discussions with the project team on the application's design, security controls, and other implementation details. We were provided with source code, application design diagrams, a set of test credentials for each application role, previous threat models or security reviews, and other resources to give us the greatest advantage in identifying major weaknesses within the service; this is colloquially known as white-box methodology.

Our review uncovered no findings of vulnerability.

Observations

We confirmed that the Aiven Postgres Gateway module does not include any malicious or extraneous functionality. The coding style used is clear, concise, and well documented with comments. The logic flow is linear and avoids complexity, making it robust against threats. We identified three minor instances in which coding best practices are not being followed. However, they do not constitute any vulnerability or potential exploit in the code. There are multiple uses of strcmp() and a failure to check memory allocations that use malloc(); both of these factors could cause the program to crash. In addition, we found a memory leak that could eventually cause a denial-of-service (DoS) condition. Although these items do not pose a security risk, they present an opportunity to improve coding practices.

Recommendations

Implement checks for buffer length, and substitute vector operations that do not check bounds with ones that do.

Ensure that memory allocations are checked before dereferencing the pointer that is returned from the allocation.

Address the memory-allocation issues, with a specific focus on error-handling paths and reference counting, as detailed in finding 107333.

Vulnerability Classification

Impact

When we find a vulnerability, we assign it one of five categories of severity, describing the potential impact if an attacker were to exploit it:

Informational – Does not present a current threat but could pose one in the future if certain changes are made. To protect against future vulnerabilities, fixing the condition is advisable.

Low – May allow an attacker to gain information that could be combined with other vulnerabilities to carry out further attacks. May allow an attacker to bypass auditing or minimally disrupt availability, resulting in minor damage to reputation or financial loss.

Medium – May allow an attacker inappropriate access to business assets such as systems or servers. There may be impact to the confidentiality or integrity of data, or limited disruption of availability, resulting in moderate damage to reputation or financial loss.

High – May allow an attacker inappropriate access to business assets such as systems or servers. There may be substantial or widespread impact to the confidentiality or integrity of particularly sensitive data, or disruption of availability, resulting in significant damage to reputation or financial loss.

Critical – May allow an attacker to gain persistence, or imminently disrupt functionality or disclose data, resulting in severe reputational damage or financial loss.

Skill Level to Exploit

When we find a vulnerability, we assess how skilled an attacker must be to exploit it:

Simple – Requires minimal understanding of the underlying technology. Tools/ techniques for exploiting the vulnerability can be easily found on the internet.

Moderate – Requires significant expertise, possibly in proprietary information, or access to tools that are not readily available to individuals. The unwitting cooperation of a victim or target may also be required.

Advanced – Requires insider access or access to tools that are not publicly available. Successful exploitation of another vulnerability may be required. Direct interaction with the victim or target may also be required.

Impact Rating (Weight)	Skill Level to Exploit Rating (Weight)		
	Advanced (1)	Moderate (2)	Simple (3)
Critical (4)	4	8	12
High (3)	3	6	9
Medium (2)	2	4	6
Low (1)	1	2	3

Severity	
Critical	10-12
High	7-9
Medium	4-6
Low	1-3

Vulnerability Index

This section represents a quick view into the vulnerabilities discovered in this assessment.

ID	INITIAL SEVERITY	CURRENT SEVERITY	TITLE	COMPONENT
107332	Info	**Fixed**	Failure to check if malloc() failed (remedied)	Aiven Postgres Gatekeeper
107333	Info	**Fixed**	Memory leak (remedied)	Aiven Postgres Gatekeeper
107337	Info	**Fixed**	strcmp() in use (remedied)	Aiven Postgres Gatekeeper

Activity Index

This section represents a quick view into the activities performed in this assessment.

COMPONENT	TITLE	STATUS
Aiven Postgres Gatekeeper	Insecure functions	Complete
Aiven Postgres Gatekeeper	Coding best practice	Complete

Observations & Analysis

For the purposes of evaluation, we grouped this project into a single component based on design documentation and discussions with the service team.

Aiven Postgres Gatekeeper

Aiven has implemented a Postgres preload library module that restricts access to actions that require privileged root access on the underlying operating system. It does this by adding utility functions that act as a gateway and is written in C.

Threat Analysis

Potential issues include memory management; for example, when buffers are not bound correctly, an attacker might be able to overwrite areas of memory. If memory is not cleared when it is not in use, an attacker could exploit the residual values. Incorrect allocation and freeing of system memory can lead to instability of the program.

Activities Performed

INSECURE FUNCTIONS

Scope

Review code to see if insecure functions are being used.

Methodology

Review the code for uses of potentially insecure functions, such as `strcpy`, `strcmp`, and `strcat`, and determine whether or not they are being used in an insecure manner.

Observations

The code is well written and generally avoids using insecure functions. The use of `strcmp()` was observed, but the function is used in a defensive manner that does not expose an attack surface as it does not utilize user-provided values.

Related Findings

107337: strcmp() in use

CODING BEST PRACTICE

Scope

Validate that coding best practices are being observed.

Methodology

Verify that all return values are being validated and that when necessary, memory is correctly cleared. Confirm that things such as file descriptors are used rather than performing file operations by name and that preprocessor macros and the like are used in a safe way.

Observations

In general, the code is robustly written and well documented. The logic and state flow is straightforward with no unnecessary complexity or extraneous functions, which minimizes the available attack surfaces. A few issues with coding best practices were observed in relation to the use of malloc() and strcmp().

Related Findings

107332: Failure to check if malloc() failed
107333: Memory leak

Vulnerabilities

REMEDIED - FAILURE TO CHECK IF MALLOC() FAILED

ID	107332
Component	Aiven Postgres Gatekeeper
Original Severity	**Fixed** (initially Info)
Impact / Skill Level	Informational/Advanced
Reference	https://cwe.mitre.org/data/definitions/690.html
Location	aiven_gatekeeper.c: Line 340

Observation

Memory allocation calls can fail, for example, if there is not sufficient available memory to grant the request. When software assumes that an allocation succeeds and does not check the return code, it creates conditions that could lead to an application crash.

The code calls **`malloc()`** and assumes that the call will succeed, which may result in the pointer and value of **`reserved_func_oids`** to be incorrect and the program to crash.

Impact Rationale:
In rare circumstances, when NULL is equivalent to the 0x0 memory address and privileged code can access it, then writing or reading memory is possible, which may lead to code execution because the code in use has superuser privileges when executed.

Difficulty Rationale:
An attacker would need to modify data defined initially in the code as a static const char structure and subsequently compiled into the text or bss segment of the binary, which is extremely unlikely.

Recommendation

Ensure that memory allocations are checked before dereferencing the pointer that is returned from the allocation.

Description of Remedy

Calls to malloc are now properly checked. If malloc fails, the set_reserved_oids function returns false and none of the hooks are registered.

REMEDIED - MEMORY LEAK

ID	107333
Component	Aiven Postgres Gatekeeper
Original Severity	**Fixed** (initially Info)
Impact / Skill Level	Informational/Advanced
Reference	https://cwe.mitre.org/data/definitions/401.html
Location	aiven_gatekeeper.c: Line 340

Observation

C and C++ programs must manage their memory usage manually, explicitly allocating memory for as long as it is needed, and then releasing it when no longer in use. A memory leak occurs when the program is no longer using allocated memory but forgets to release it. Chronically doing this will cause the program's memory usage to continuously grow until the program crashes. This becomes a denial-of-service (DoS) vulnerability if attackers can induce the program to grow its memory usage quickly.

When the **`aiven_gatekeeper`** function **`_PG_init`** is called, it in turn calls **`set_reserved_oids`**, which invokes **`malloc()`** to reserve memory for **`reserved_func_oids`**. The memory remains allocated, and there is no corresponding **`free()`** nor are there any count references to make sure all allocated memory has been unallocated when the library is unloaded from Postgres. This results in a small amount of memory being allocated every time **`set_reserved_oids()`** is called.

Impact Rationale:
DoS may occur due to memory pressure on the OS over time as the gateway module is initialized and uninitialized.

Difficulty Rationale:
When **`malloc()`** is called without a corresponding **`free()`** it causes a memory leak that constrains free memory available on a system. To exploit this, an attacker must exercise the memory leak by attempting to run commands excessively exercising the library to cause the software to crash. This is a very difficult threat surface to exploit.

Recommendation

Free memory when it is no longer necessary in **`_PG_fini()`** and the library is unloaded. Give specific attention to error-handling paths to ensure they do not branch past deallocation. Make sure to count references when **`malloc()`** has been called and avoid calling **`free()`** twice.

Description of Remedy

If it was initially allocated successfully, and is not a null pointer, **`reserved_func_oids`** is now freed in **`_PG_fini`**.

REMEDIED - STRCMP() IN USE

ID	107337
Component	Aiven Postgres Gatekeeper
Original Severity	**Fixed** (initially Info)
Impact / Skill Level	Informational/Advanced
Reference	https://owasp.org/www-community/attacks/Buffer_overflow_attack
Location	aiven_gatekeeper.c: Lines 220, 234, 238, 325

Observation

In type-unsafe languages such as C/C++, array bounds (buffers) are not automatically checked, so insufficient or nonexistent array bounds checking can result in oversize input to the program. This can corrupt the program's internal state, often giving an attacker full control of the program. Functions such as strcmp() use NULL termination as a bounds check to stop comparing values. If the NULL is not present, a buffer overrun may occur and any memory reference to an unintended address is undefined behavior.

Multiple uses of **`strcmp()`** are present in the code, although all uses are using direct comparison to fixed strings directly coded, or to strings stored in a static const array of values. It is extremely unlikely that this use of **`strcmp()`** would lead to an exploitable condition. It is, however, considered best practice to instead use **`strncmp`**, which does not rely on NULL terminated strings.

Impact Rationale:
A buffer overflow, or buffer overrun, is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory. If data being compared is not NULL terminated, **`strcmp`** can cause a program crash and buffer overrun.

Difficulty Rationale:
An attacker would need change data in the text or BSS region of the binary in a manner to craft an exploit for a buffer overrun; the likelihood of success is extremely low.

Recommendation

Implement checks for buffer length. Avoid using "large enough" static buffers without checking. Substitute vector operations that do not check bounds (e.g., **`strcmp`**) with ones that do (e.g., **`strncmp`**).

Description of Remedy

All uses of **`strcmp()`** have been replaced with **`strncmp()`**.

Appendix A – Technical Services

Leviathan's Technical Services group brings deep technical knowledge to your security needs. Our portfolio of services includes software and hardware evaluation, penetration testing, red team testing, incident response, and reverse engineering. Our goal is to provide your organization with the security expertise necessary to realize your goals.

SOFTWARE EVALUATION We provide assessments of application, system, and mobile code, drawing on our employees' decades of experience in developing and securing a wide variety of software. Our work includes design and architecture reviews, data flow and threat modeling, and code analysis using targeted fuzzing to find exploitable issues.

HARDWARE EVALUATION We evaluate new hardware devices ranging from novel microprocessor designs, embedded systems, mobile devices, and consumer-facing end products, to core networking equipment that powers internet backbones.

PENETRATION & RED TEAM TESTING We perform high-end penetration tests that mimic the work of sophisticated attackers. We follow a formal penetration testing methodology that emphasizes repeatable, actionable results that give your team an understanding of the overall security posture of your organization as well as the details of discovered vulnerabilities.

SOURCE CODE-ASSISTED SECURITY EVALUATIONS We conduct security evaluations and penetration tests based on our code-assisted methodology, allowing us to find deeper vulnerabilities, logic flaws, and fuzzing targets than a black-box test would reveal. This methodology gives your team a stronger assurance that the most significant security-impacting flaws have been found, allowing your team to address them.

INCIDENT RESPONSE & FORENSICS We respond to our customers' security incidents by providing forensics, malware analysis, root cause analysis, and recommendations for how to prevent similar incidents in the future.

REVERSE ENGINEERING We assist clients with reverse engineering efforts. We provide expertise in investigations and litigation by acting as experts in cases of suspected intellectual property theft.

Appendix B – Risk and Advisory Services

Leviathan's Retained Services group is a supplement to an organization's security and risk management capability. We offer a pragmatic information security approach that respects our clients' appetites for security process and program work. We provide access to industry leading experts with a broad set of security and risk management skills, which gives our clients the ability to have deep technical knowledge, security leadership, and incident response capabilities when they are needed.

INFORMATION SECURITY STRATEGY DEVELOPMENT We partner with boards, directors, and senior executives to shape your enterprise's overall approach to meeting information security requirements consistently across an entire organization.

ENTERPRISE RISK ASSESSMENT We develop an information asset-centric view of an organization's risk that provides insight to your organization's Enterprise Risk Management capability. This service can be leveraged with annual updates, to account for your organization's changing operations, needs, and priorities.

PRIVACY & SECURITY PROGRAM EVALUATION We evaluate your organization's existing security program to give you information on compliance with external standards, such as ISO 27000 series, NIST CSF, HIPAA, or PCI-DSS. This is often most useful before a compliance event or audit and helps to drive the next phase of growth for your Security and Risk Management programs.

VENDOR RISK ASSESSMENT We assess the risk that prospective vendors bring to your organization. Our assessment framework is compatible with legislative, regulatory, and industry requirements, and helps you to make informed decisions about which vendors to hire, and when to reassess them to ensure your ongoing supply chain security.

NATIONAL & INTERNATIONAL SECURITY POLICY In 2014, we launched a public policy research and analysis service that examines the business implications of privacy and security laws and regulations worldwide. We provide an independent view of macro-scale issues related to the impact of globalization on information assets.

M&A/INVESTMENT SECURITY DUE DILIGENCE We evaluate the cybersecurity risk associated with a prospective investment or acquisition and find critical security issues before they derail a deal.

LAW FIRM SECURITY SERVICES We work with law firms as advisors, to address security incidents and proactively work to protect client confidences, defend privileged information, and ensure that conflicts do not compromise client positions. We also work in partnership with law firms to respond to their clients' security needs, including in the role of office and testifying expert witnesses.

SAAS AND CLOUD INITIATIVE EVALUATION We give objective reviews of the realistic threats your organization faces both by moving to cloud solutions and by using non-cloud infrastructure. Our employees have written or contributed to many of the major industry standards around cloud security, which allows their expertise to inform your decision-making processes.