Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Pentest-Report Dive CAE Web UI, API & Infra 09.2025

Cure53, Dr.-Ing. M. Heiderich, BSc. C. Kean, MSc. A. Schloegl, BSc. C. Mayr, L. Stockner, H. Crawford

Index

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Introduction

“We provide a compelling and fast cloud-native CAE experience combining smart simulation technologies with scalable cloud HPC. Our simple browser-based software hides complexity while giving engineers access to data and tools from anywhere in the world.”

From https://www.divecae.com/about

This report describes the results of a security assessment of the Dive CAE web application complex, focusing on its frontend UI and backend API endpoints, its SSO features, as well as its Azure and k8s setup. The project, which included source code audits and penetration tests conducted in both white- and gray-box manners, was conducted by Cure53 in September 2025.

The audit, registered as *DIV-03*, was requested by the Dive CAE (formerly dive solutions GmbH) in January 2025 and then scheduled to start in the late second quarter of the year to give both sides time to prepare.

The project is the third cooperation between Cure53 and Dive CAE on security matters. In fact, the preceding tests entailed investigations of some of the same aspects. These assessments took place back in May 2023 (see *DIV-01*), as well as again more recently, namely in August and September 2024 (see *DIV-02*).

In terms of the exact timeline and specific resources allocated to *DIV-03*, the Cure53 team has completed their research in CW37. In order to achieve the expected coverage for this task, a total of twelve days were invested. A team consisting of six senior testers was formed and assigned to the preparation, execution, documentation, and delivery of this project.

For optimal structuring and tracking of tasks, the assessment was divided into three separate work packages (WPs):

- **WP1**: Gray-box penetration tests & assessments of dive solutions SSO features
- **WP2**: White-box penetration tests & assessments of dive solutions web UI & API
- **WP3**: White-box penetration tests & reviews of dive solutions Azure & k8s setup

As the titles of the WPs indicate, mixed-methodology was overall used during *DIV-03*. More specifically, gray-box approaches were leveraged for inspections of the SSO features (WP1), while the web applications and the Azure and k8s setup (WP2 & WP3) were tested through white-box methods. Cure53 was provided with URLs, sources, test-user credentials, as well as all further means of access required to complete the tests.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

The project could be carried out without any major issues. To facilitate a smooth transition into the testing phase, all preparations were completed in CW36 of 2025. Throughout the engagement, communications were conducted through a private, dedicated, and shared Slack channel. Stakeholders - including Cure53 testers and the internal staff from Dive CAE - were able to participate in discussions in this space.

Cure53 did not need to ask many questions, and the quality of all project-related interactions was consistently excellent. Although the testers offered frequent status updates on the examination and emerging findings, live-reporting was not used during this project. Continuous communication contributed positively to the overall results of this project. Significant roadblocks were avoided thanks to clear and careful preparation of the scope, as well as through subsequent support.

The Cure53 team achieved very good coverage of the WP1-WP3 objectives. Of the seven security-related discoveries, only one was classified as a security vulnerability and six were classified as general weaknesses with low exploitation potential.

The overall small number of findings, as well as the general lack of issues above a Low severity rating, indicates that the inspected Dive CAE web application and features have already been correctly strengthened. Cure53 can confirm that good security measures have been crafted and put in place across the Dive CAE SSO features, web UI, API and infrastructure inspected during *DIV-03*.

Nevertheless, Cure53 recommends addressing all findings in a timely manner, even though they might carry risks from the lower-end of the threat spectrum. The current impact of the findings aside, it needs to be acknowledged that these kinds of flaws often become stepping stones for more severe and sophisticated attacks.

The following sections first describe the scope and key test parameters, as well as how the work packages were structured and organized.

Then, what the Cure53 team did in terms of attack attempts, coverage, and other test-related tasks is explained in a separate chapter on test methodology.

Next, all findings are discussed in grouped vulnerability and miscellaneous categories. The problems are then discussed chronologically within each category. In addition to technical descriptions, PoC and mitigation advice is provided where applicable.

The report ends with general conclusions relevant to this summer 2025 project. Based on the test team's observations and the evidence collected, Cure53 elaborates on the overall impressions and reiterates the verdict. The final section also includes tailored hardening recommendations for the inspected components within the Dive CAE complex, specifically the project’s frontend UI and backend API endpoints, SSO features, as well as Azure and k8s setup.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Scope

- **Penetration tests & assessments of selected components and aspects**
  - **WP1:** Gray-box penetration tests & assessments of dive solutions SSO features
    - **URL (staging):**
      - https://app.preview.dive-solutions.de
  - **WP2:** White-box penetration tests & assessments of dive solutions web UI & API
    - **URLs:**
      - **Staging application:**
        - https://app.preview.dive-solutions.de
      - **Admin panel:**
        - https://app.preview.dive-solutions.de/admin
      - **REST API:**
        - https://app.preview.dive-solutions.de/api
      - **(Old) "Clipper" websocket API:**
        - wss://app.preview.dive-solutions.de/fetch
      - **(New) "Pontoon" websocket API (explicit focus):**
        - wss://app.preview.dive-solutions.de/pontoon
      - **Knowledge base (Hubspot):**
        - https://help.dive-solutions.de
      - **Production application (EU):**
        - https://eu.divecae.app
  - **WP3:** White-box penetration tests & reviews of dive solutions Azure & k8s setup
    - **Access via invite to Intra ID tenant**:
      - **Domain:**
        - https://www.divecae.com/
      - **Account invites:**
        - U: alex@rs.cure53.de
        - U: christian@rs.cure53.de
  - **Credentials for test-users**
    - **Admin** (staging):
      - E: christian@rs.cure53.de
      - E: alex@rs.cure53.de
    - **Contributor** (staging)
      - E: chris@cure53.de
  - **Test-supporting material was shared with Cure53**
  - **All relevant sources were shared with Cure53**

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Test Methodology

Since this penetration testing iteration did not reveal any vulnerabilities that could be exploited, this section provides a more detailed description of Cure53's testing methodology. On the one hand, this aims to bring more transparency into the overall work that was performed during this *DIV-03* security assessment. On the other hand, it also provides assurance of the extensive coverage achieved and the work completed within the given scope of this project.

The testing methodology employed during this assessment explicitly covers the designated focus areas, as listed by the Dive CAE team. These included the inspection of the Security Assertion Markup Language (SAML) integration for B2B customers' identity providers, the newly introduced *Pontoon* websocket API, and the implementation of strict user separation between the knowledge base and main application. The following sections outline the approaches used in each of the work packages delineated in the Scope of Work (SOW) documentation.

WP1: Gray-box penetration tests & assessments of dive solutions SSO features

In order to audit the integration of customers' identity providers (IdP), a custom Auth0 instance was set up by Cure53 and linked to the pentesting organization. This integration utilizes SAML to anchor trust and ensure authenticity.

Cure53 audited the SAML exchange to ensure all requests are replay-protected and properly verified for authenticity. Attempts to inject or falsify data during the SAML login process were rejected correctly on the basis of invalid signatures.

Overall, the usage of Auth0 as a third-party authentication provider was judged positively, as it left little room for error or misconfiguration with regard to B2B IdP integrations.

Finally, it was tested whether the application only accepts login tickets from the IdP where users already exist in the dive database. This ensures improperly configured IdPs will not jeopardize the security of the dive ecosystem as a whole.

WP2: White-box penetration tests & assessments of dive solutions web UI & API

For both the web frontend and the backend, Dive CAE provided source code access to Cure53, thus upgrading the engagement from a grey-box to a white-box assessment. This approach can be seen as more robust when it comes to excluding hidden or deeply nested risks. For the backend, the assessment focused on analyzing the codebase for common API security pitfalls[1].

[1] https://owasp.org/www-project-api-security/

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

This entailed - but was not limited to - checking for Broken Object Level Authorization (BOLA), authentication weaknesses, authentication in general and security misconfigurations. Server-Side Request Forgery (SSRF) vulnerabilities were considered as well.

To this end, static analysis tools were first employed to evaluate the overall security posture, and to identify low-complexity findings through code flow analysis with predefined rules. This process also covered configuration files related to the building and deployment of the backend. Static code analysis did not yield any findings or security recommendations.

Next, the object-level authorization controls in the API were audited. Overall, the access control for creating, updating, and deleting items across the dive application was found to be reasonably secure. All but one of the tested endpoints correctly check the permissions of the current user before granting access. The one exception documented in this report is a minor information leak documented in DIV-03-001. The issue stems from the fact that one endpoint delivers information for two web pages that are only access-controlled by the user interface.

Another key area of focus was the implemented authentication mechanism. For access to simulation and documentation applications, dive implements Auth0, which provides a single method of sign-in across their applications. A specific area of concern was whether a self-registering user who signed up via the documentation application, could in any way access the main simulation application. Cure53 validated that the implemented controls are appropriate and robust to prevent users from accessing other applications in this scenario.

During the sign-in process, all successful authentication attempts are assessed to check whether the user has the application attribute of *"knowledgebase only*" assigned to their account. Since this attribute cannot be modified by users, and its validation logic runs prior to the issuance of an *authentication* token, no risks could be noted.

In the end, the authorization logic was deemed to be secure. This was further validated with the support of dynamic testing, reviewing whether the application logic is vulnerable to injection or timing attacks. This manual testing confirmed that the implementation of this control is appropriate.

While auditing the dive solutions web UI and API, several other configuration-related issues were identified. While they are lower in impact, it could not be excluded that they still introduce unnecessary risk if left unaddressed.

Some applications lacked explicit controls to prevent indexing by search engines (DIV-03-004), signifying that potentially sensitive or internal-facing endpoints could appear in public search results, if discovered. In addition, there were misconfigurations and omissions in the use of security-related HTTP headers (DIV-03-005).

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Common issues included missing *Referrer-Policy* headers, which may allow older browsers to leak sensitive referrer information.

The absence of consistent anti-framing protections across applications was raised as it means that pages are potentially exposed to clickjacking. Only one application host implemented a limited *Content-Security Policy*, offering little meaningful defense-in-depth against modern browser-based attacks (DIV-03-007). Similarly, while *HSTS* was present, it was not configured with preload or applied to subdomains, which weakens its effectiveness against downgrade and first-visit attacks. Taken together, these gaps do not represent immediately critical risk, but they reflect an overall lack of hardened baseline across configurations.

The WebSocket APIs were closely audited for injection vulnerabilities in client-to-server messages. The main focus was laid on the newly added *pontoon* notification API. The entire websocket stack and configuration was audited for opportunities to inject commands or malicious data.

In this portion of the audit, Cure53 found proper authentication to be in place for all messages. A type-checked, library-assisted parsing of user data is performed before handling any data. Only *heartbeat* message types are accepted in the first place, and their handling was observed to be free of vulnerabilities. All other message types are immediately rejected.

The user interface frontend was also reviewed for common security pitfalls and weaknesses. The various parameters of user-controlled input supplied to the application were populated with HTML markup to check for the possibility of unsanitized rendering within the various sinks of the application. However, all submitted payloads were either blocked or correctly output-encoded, resulting in no findings concerning HTML injection or Cross-Site Scripting (XSS).

WP3: White-box penetration tests & reviews of dive solutions Azure & k8s setup

Dive's Azure environment was audited through live access, as well as the Terraform Infrastructure-as-Code (IaC) sources provided by the commissioning team. The audit showed a cloud environment that evidently considers security as a core principle.

More precisely, various security features offered by Azure, like KeyVaults for secret management, are properly utilized. Concerns in the dive ecosystem are also properly separated using microservices. This leaves each individual microservice with a small and easy-to-understand attack surface.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

In the course of auditing the Azure environment, it was discovered that many resources, among them sensitive key vaults, are open to the public Internet. While this is apparently known to the dive development team, a ticket discussing the associated risks has still been created (see DIV-03-006).

VPC and especially ingress configurations were audited using automated tooling and manual reviews of the results. This highlighted the fact that isolation of dive's VPC is currently incomplete, and many internally used services (and KeyVaults) are openly accessible from the internet. Although protections from Azure RBAC rules still apply, a cleaner separation of the VPC, potentially combined with a dedicated VPN into the VPC, would be preferable in this context.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Identified Vulnerabilities

The following section lists all vulnerabilities and implementation issues identified during the testing period. Notably, findings are cited in chronological order rather than by degree of impact, with the severity rank offered in brackets following the title heading for each vulnerability. Furthermore, each ticket has been given a unique identifier (e.g., *DIV-03-001*) to facilitate any follow-up correspondence, if needed.

DIV-03-001 WP2: Missing ACL grants access to subscription data *(Low)*

The dive solution web application offers two user-roles to manage and restrict access. These are the less-privileged *Contributor* role and the privileged *Admin* role. While reviewing the restrictions imposed on the *Contributor* role, it was revealed that it can access *Subscription* data from the underlying *Subscription* API endpoints. This goes against what the landing page for those API endpoints appears to do, as access restrictions should apply.

The impact of this issue was evaluated as *Low* because only some data regarding the subscription is disclosed beyond what can be already found in the *Usage & Budget* endpoint. At the same time, the flaw demonstrates a misimplementation of access control in which access is only restricted cosmetically in the UI, without being enforced in the API.

Landing page:
https://app.preview.dive-solutions.de/settings/subscription

Affected API endpoint:
https://app.preview.dive-solutions.de/api/organizations

The data is only accessible to *Contributors* in the *Subscription* landing page and not in the *Usage & Budget* page.

To mitigate this issue, Cure53 recommends enforcing the access control mechanisms implied within the user interface on the server-side. This may include separating the information from the *Organizations* endpoint into two separate endpoints, one for *Usage & Budget* and one for *Subscription* information. This approach would match the separation found in the user interface. The *OWASP Authorization Cheat Sheet*[2] can be reviewed for further guidance in the context of hardening this aspect of the application.

[2] https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Miscellaneous Issues

This section covers any and all noteworthy findings that did not incur an exploit but may assist an attacker in successfully achieving malicious objectives in the future. Most of these results are vulnerable code snippets that did not provide an easy method by which to be called. Conclusively, while a vulnerability is present, an exploit may not always be possible.

DIV-03-002 WP2: SSH service hardening for employed algorithms *(Info)*

An automated review utilizing *ssh-audit*[3] revealed several minor misconfigurations concerning the underlying SSH configuration utilized for the SFTP server. The component in question is deployed via the Virtual Machine settings on *app.preview.dive-solutions.de* and hosted on *udxzlpjaloesrqhf.dev.divesolutions.de*.

The SSH service exhibited several opportunities for hardening. Improvements can be achieved by using stronger algorithms and omitting insecure cryptographic parameters, such as weak elliptic curves or hashing algorithms.

Nevertheless, the impact of this issue was merely deemed *Info*, since successful exploitation of these weak cryptographic parameters within an SSH protocol context tends to require both significant resources and Man-in-the-Middle (MitM) capabilities.

Command:

```
./ssh-audit.py udxzlpjaloesrqhf.dev.divesolutions.de
[...]
# algorithm recommendations (for OpenSSH 8.9)
(rec) -ecdh-sha2-nistp256                    -- kex algorithm to remove
(rec) -ecdh-sha2-nistp384                    -- kex algorithm to remove
(rec) -ecdh-sha2-nistp521                    -- kex algorithm to remove
(rec) -ecdsa-sha2-nistp256                   -- key algorithm to remove
(rec) -hmac-sha1                             -- mac algorithm to remove
(rec) -hmac-sha1-etm@openssh.com             -- mac algorithm to remove
(rec) -diffie-hellman-group14-sha256         -- kex algorithm to remove
(rec) -hmac-sha2-256                         -- mac algorithm to remove
(rec) -hmac-sha2-512                         -- mac algorithm to remove
(rec) -umac-128@openssh.com                  -- mac algorithm to remove
(rec) -umac-64-etm@openssh.com               -- mac algorithm to remove
(rec) -umac-64@openssh.com                   -- mac algorithm to remove
```

To mitigate this issue, Cure53 advises reviewing the items enumerated above and disabling those considered surplus to requirement. A tightened access strategy will help minimize the attack surface incurred by weak cryptographic parameters.

[3] https://github.com/jtesta/ssh-audit

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

DIV-03-003 WP2: Remote VM hardening recommendations *(Low)*

While auditing the post-processing environment in dive, the Cure53 testers noted that the test VMs allow access via SFTP, and thus via SSH. This can lead to unforeseen actions of potentially malicious users operating on these VMs.

Based on communication between Cure53 and dive, it seems that the main intent behind this setup is the possibility of offering customers the option to post-process simulation results without first pulling the intermediate files to their network. The SFTP access mechanism seems to also be based on the requirements associated with some customers.

Nevertheless, some hardening recommendations can be made in this context. While the VMs are properly isolated from other components in the dive ecosystem, no isolation towards the public Internet or the Azure VPC internal components seems to be in place.

This means that the other dive components are adequately separated and protected from attacks. However, attackers can still gain information about the Azure deployment using the Azure *metadata* endpoint. In the same vein, illicit or illegal activity can also be performed from the VMs. Allowing this could drag dive into potential litigation.

To increase the robustness of the post-processing VMs, Cure53 recommends filtering *outbound* connections, such that traffic can only travel the already established routes. Thus, data can flow via the connection established to the VM, but no further requests can be made, neither into the Azure environment nor to the public Internet.

DIV-03-004 WP2: Lack of search engine protection *(Info)*

While reviewing application server configurations via dynamic testing, it was observed that multiple applications lack proper search engine protection. Both the production and staging environments of the dive CAE applications fail to implement *robots.txt* or equivalent mechanisms to restrict automated indexing.

As a result, more sensitive routes such as */admin* and */api* may be indexed by search engines and found via OSINT activities. Although search engine protection decreases the risk of unintended data exposure, there is a negligible security impact because web crawlers do not have to respect this configuration. This finding has been included for information-sharing purpose only.

Affected URIs:

- *https://app.preview.dive-solutions.de*
- *https://eu.divecae.app*

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Cure53 recommends implementing proper search engine protection mechanisms - such as configuring *robots.txt* files and *meta* tags - to prevent unauthorized indexing of sensitive directories and endpoints. This measure will reduce the risk of exposing critical application content through public search engines, concurrently strengthening the overall security posture.

DIV-03-005 WP2: Lack of general HTTP security headers *(Low)*

During the review of the in-scope applications and web servers, several issues were noted with regard to security-related headers. None of the applications set a *Referrer-Policy* header, which could result in the browser including sensitive information in the *Referrer* header.

What is more, anti-framing controls through the *X-Frame-Options* were generally absent, though they could be spotted on the authentication page. Finally, while *HSTS* was implemented, it was missing the *includeSubDomains* directive in the *help.dive-solutions.de* application.

It is recommended to generally review the composition of security headers, paying particular attention to the following options.

- ***X-Frame-Options****:* This header specifies whether the web page is allowed to be framed. Although this header is known to prevent clickjacking attacks, a plethora of alternative breach strategies are achievable when a web page is frameable[4]. Cure53 recommends configuring the value to either **SAMEORIGIN** or **DENY**.

- ***Strict-Transport-Security***: The absence of the HSTS header may encourage a Man-in-the-Middle (MitM) to attempt channel downgrade attacks using readily available tools such as *sslstrip*[5]. In this scenario, the attacker would simply proxy clear-text traffic to the victim user and establish an SSL connection with the targeted website, stripping all cookie security flags if required. Cure53 recommends configuring the header as follows:

  Strict-Transport-Security: max-age=31536000; includeSubDomains;

  In cases where HSTS was implemented, Cure53 observed a lack of use of the *includeSubDomains* directive. Pertinently, the HSTS *preload* flag has been omitted since it is considered a risk-inducing implementation[6].

- ***Referrer-Policy****:* This header allows a site to restrict how much referrer information is included in requests. Failing to set this header correctly might inadvertently have

[4] https://cure53.de/xfo-clickjacking.pdf
[5] https://github.com/moxie0/sslstrip
[6] https://www.tunetheweb.com/blog/dangerous-web-security-features/

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

a user leak potentially sensitive information. The risk is carried through the referrer header upon navigating to another site.

Failure to incorporate beneficial security headers is suboptimal and should be avoided. To mitigate this issue, Cure53 advises inserting the aforementioned headers into every server response, including error responses such as 4xx items.

Generally speaking, it is important to deploy all HTTP headers at a specific, shared, and central location, rather than randomly assigning them. This should either be handled by a load-balancing server or a similar infrastructure. If this is deemed infeasible, then remediation can be achieved by deploying a web server configuration and a matching module.

DIV-03-006 WP3: Open ingress in Azure resources *(Low)*

Note: *This issue is known to the dive developers, yet it was decided to include it in the report for tracking.*

While auditing the Azure environment of dive, testers noticed promiscuous ingress rules, permitting access from any IP address. The listed resources still have RBAC in place; however, limiting access to operations within a VPC/VPN's CIDR range would reduce the likelihood of credential spraying and brute-force attacks.

Affected registry:

/subscriptions/9d1f24f0-6f5f-4e56-8849-be60dee36027/resourceGroups/Development-Sandbox/providers/Microsoft.ContainerRegistry/registries/divesandbox

To mitigate this issue, Cure53 recommends establishing a VPC and limiting access to sensitive components, especially key vaults. For situations where direct access to the resources is required, a VPN connection into the VPC should be used to provide an IP from the appropriate CIDR range.

DIV-03-007 WP2: Weak *Content Security Policy* configuration *(Low)*

While testing, Cure53 observed that usage of the *Content SecurityPolicy (CSP)* header was very limited in dive. Only one site deployed a minimal policy (*upgrade-insecure-requests*). Consequently, a more comprehensive *CSP* should be adopted to restrict the sources of scripts, styles, and other assets, reducing the risk of XSS and other injection-based attacks.

Configured CSP:
Content-Security-Policy: upgrade-insecure-requests

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Affected resources:

- *https://app.preview.dive-solutions.de*
- *https://help.dive-solutions.de*
- *https://eu.divecae.app*

Without a *script-src* directive, the browser has no restriction on where JavaScript can be loaded from, which means an attacker could inject and execute scripts from malicious domains. Similarly, no *style-src* or *font-src* directive means that CSS and fonts can be loaded from anywhere, opening the door to malicious styling or information leaks.

Without *img-src* or *media-src*, images and videos can also be pulled from untrusted sources, which attackers sometimes exploit for tracking or exfiltration. The absence of *frame-ancestors* means the site could be embedded in a hostile *iframe*. The latter means the extended attack options for clickjacking. Leaving out *object-src* further allows dangerous legacy plugins to run. In essence, while HTTPS upgrades are enforced, the policy is missing nearly all of the content restrictions that actually reduce XSS and injection risks.

A robust CSP should extend beyond *upgrade-insecure-requests* and explicitly define trusted content sources. A *default-src 'self'* directive should be added to block all external content by default, with exceptions granted only where necessary.

Next up, a *script-src* directive should be used to limit JavaScript execution to approved domains. This needs to be done while remembering to avoid '*unsafe-inline*' and '*unsafe-eval*' whenever possible. Similarly, directives such as *style-src* and *font-src* should be configured to restrict stylesheet and font loading, while *img-src* and *media-src* should be specified to control image and media sources.

To prevent clickjacking, *frame-ancestors 'none'* or a list of trusted parents should be included, and *object-src 'none'* should be set to block legacy plugins. Finally, a *report-uri* or *report-to* directive should be implemented, so that policy violations can be monitored. The latter would also assist the dive team with refining the configuration over time.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Conclusions

This *DIV-03* assignment represents the third iteration of Cure53 being tasked with security testing the Dive CAE web application. As noted in the *Introduction,* this September 2025 engagement concludes positively, as only one *Low*-ranking vulnerability could be spotted and confirmed. Otherwise, just minor recommendations could be formulated in terms of additional hardening of the dive complex.

To reiterate, the Dive CAE components provide applications and infrastructure for computational fluid dynamics (CFD) software. Such software is designed to run computationally intensive simulations for dive customers.

As part of this assessment, Cure53 reviewed the simulation application in both the production and staging environments, as well as looked at the supporting infrastructure. The latter included a Microsoft Azure cloud subscription, documentation application, associated APIs, as well as two WebSocket APIs.

Due to the nature of the time-boxed security audit, Cure53 restricted the testing focus to tackling key security concerns. These can be enumerated as unauthorized access, cross-account access between customers, arbitrary code execution, as well as privilege escalation within the simulation environment. Notably, the in-scope applications and infrastructure were found to be well protected from such attacks.

Cure53 must underline that the authorization policy for the main API (*jellyfish*) was reviewed and found to be appropriate. As such, Cure53 shifted to producing hardening recommendations that would provide additional layers of security, in line with the defense-in-depth approach.

It was positively noted that the application does not contain serious server-side flaws like code execution or SQL injection. This indicates that the exposed attack surface is kept as small as possible. Moreover, the outcomes show that security is taken seriously at Dive CAE. The same holds for the client-side web application.

With regard to web and API security, the dive application makes a robust impression and is observably effective in minimizing the attack surface. This is further underlined by the limited severities ascribed to the reported problems, specifically not exceeding *Low*-risk levels.

The documented issues mainly concern hardening measures for services such as SSH, the remote VM, or HTTP headers, as well as access control. The proposed measures should be interpreted as suggestions rather than necessary steps. At the same time, they will help harden the dive application further.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

The Pontoon websocket API, newly introduced to provide a push mechanism for server-to-client messages, was thoroughly audited for injection capabilities, but none were found. Only heartbeats are handled by the server, where the authentication header is properly validated before proceeding. Further, deserialization of the payload is done using *Pydantic*, leaving little room for development errors.

Generally, the usage of Oso and Auth0 to handle authorization and authentication was observed to be well thought-out and properly integrated. As expected, the correct use of established third-party libraries reduces the system's attack surface. This stands in contrast to common implementations of authentication mechanisms from scratch, which risks introducing security flaws.

The business SSO integration was also found to be safely implemented. The Cure53 testing team uncovered no viable attack vectors for privilege escalation or cross-account impersonation over the course of this audit.

Last but not least, the Azure and Kubernetes configurations were found to be robust within the scope of this assessment. Some security recommendations were identified, and it is advised to review them and, ideally, follow the proposed guidance on additional improvements.

In conclusion, the results of this September 2025 assessment paint a positive picture of the system's overall security posture. The Dive CAE components benefit from security controls implemented across both frontend and backend realms, making the project capable of effectively mitigating a wide range of potential threats. By continuing to conduct regular security assessments, Dive CAE can maintain a strong security posture and minimize emerging risks.

Cure53 would like to thank Nicholas Greenall, Johannes Gutekunst and Ivo Simonsmeier from the Dive CAE (formerly dive solutions GmbH) team for their excellent project coordination, support, and assistance, both before and during this assignment.