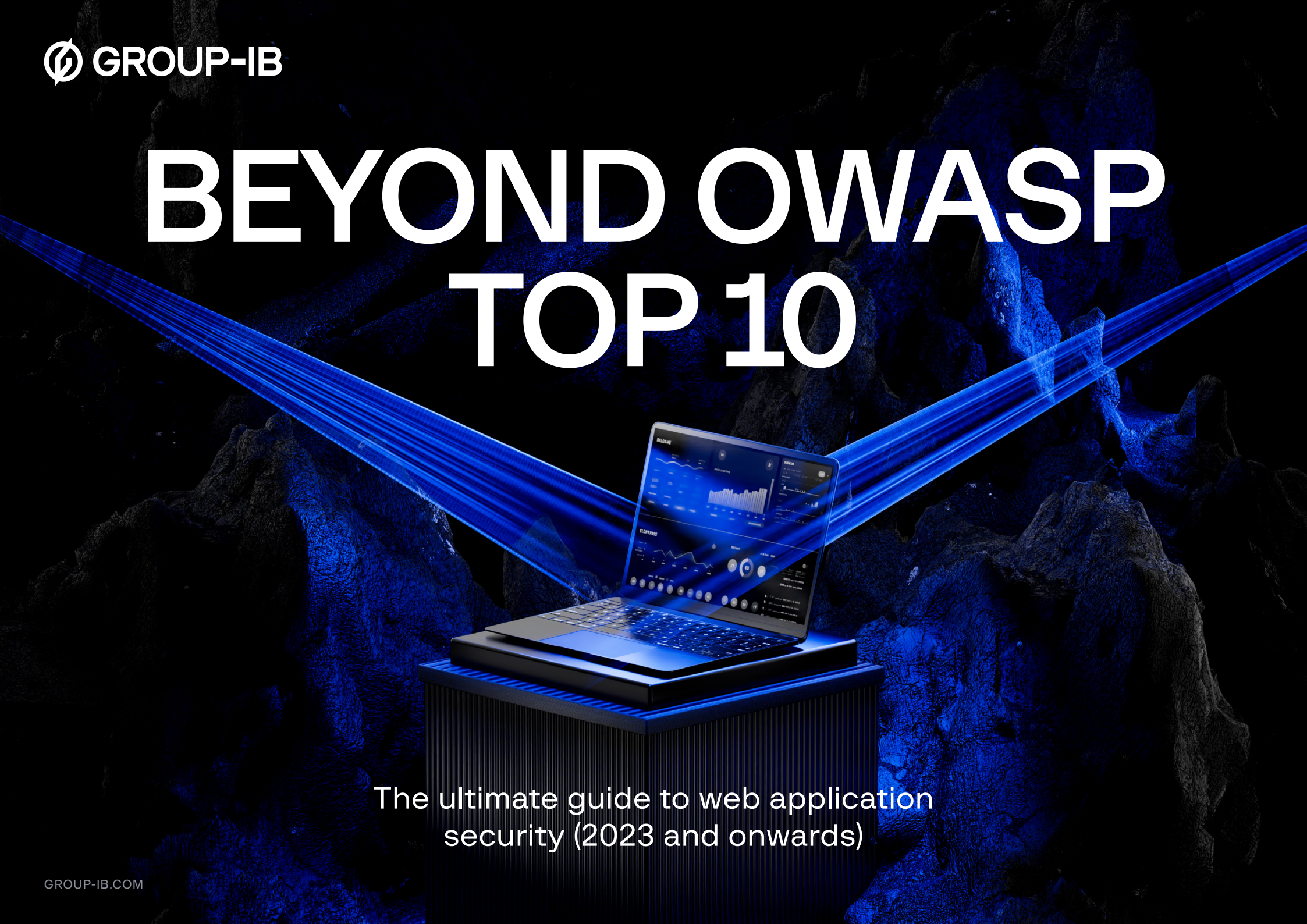
GROUP-IB

BEYOND OWASP TOP 10



The ultimate guide to web application security (2023 and onwards)

GROUP-IB

1. INTRODUCTION

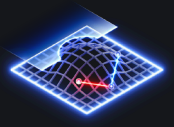
All digital businesses today run on code. From establishing an online presence and engaging with customers to streamlining operations, everything depends on software systems that are not just well-functioning but also robustly secured. The digital footprint of a business — its websites, applications, and the entire interface through which it interacts and transacts with customers — needs to be bulletproof.

Our comprehensive e-guide provides **web developers, security auditors, cybersecurity specialists, and business leaders with a detailed analysis of the most critical web application security risks based on OWASP criteria.**

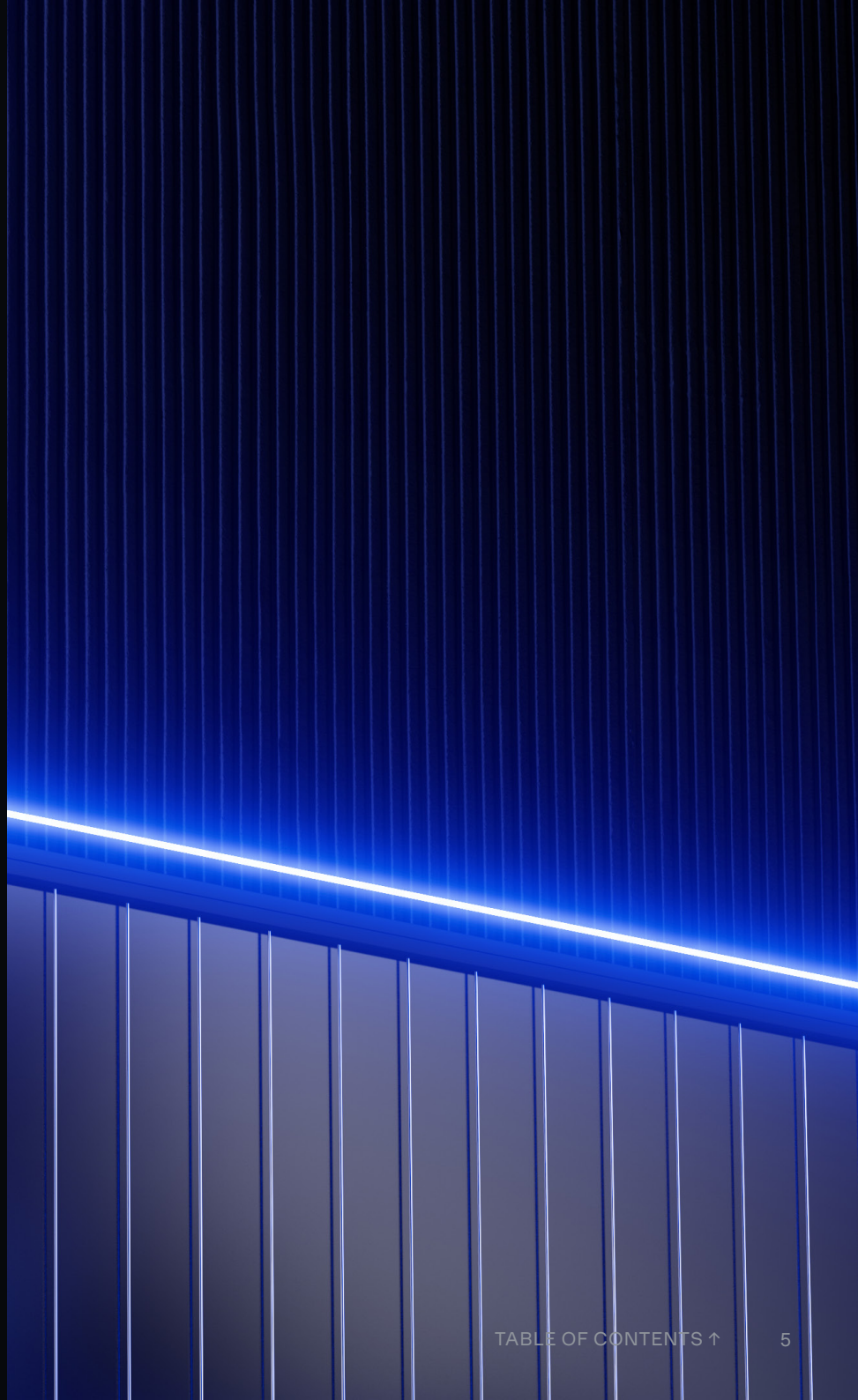
This resource goes beyond the basics by incorporating additional insights on emerging risks (gathered through our proprietary threat intelligence) mentioned by Group-IB auditors and penetration testers as well as information uncovered by CERT team members who analyze real-life attacks.

Using the knowledge shared in this guide, businesses can make their critical internet applications more secure and thereby safeguard their reputation, retain their customers, and protect their applications. It also ensures that operations remain uninterrupted in today's ever-evolving threat landscape.

To further increase their resilience, organizations can leverage Group-IB's audit and consulting technology and services to perform simulated application attacks and receive a guided audit report with significant discoveries, related concerns, and a recommendation checklist pertaining to their web-based applications.



In addition, Group-IB's Attack Surface Management (ASM) helps organizations proactively manage end-to-end asset security and reduce the external attack surface of web applications.

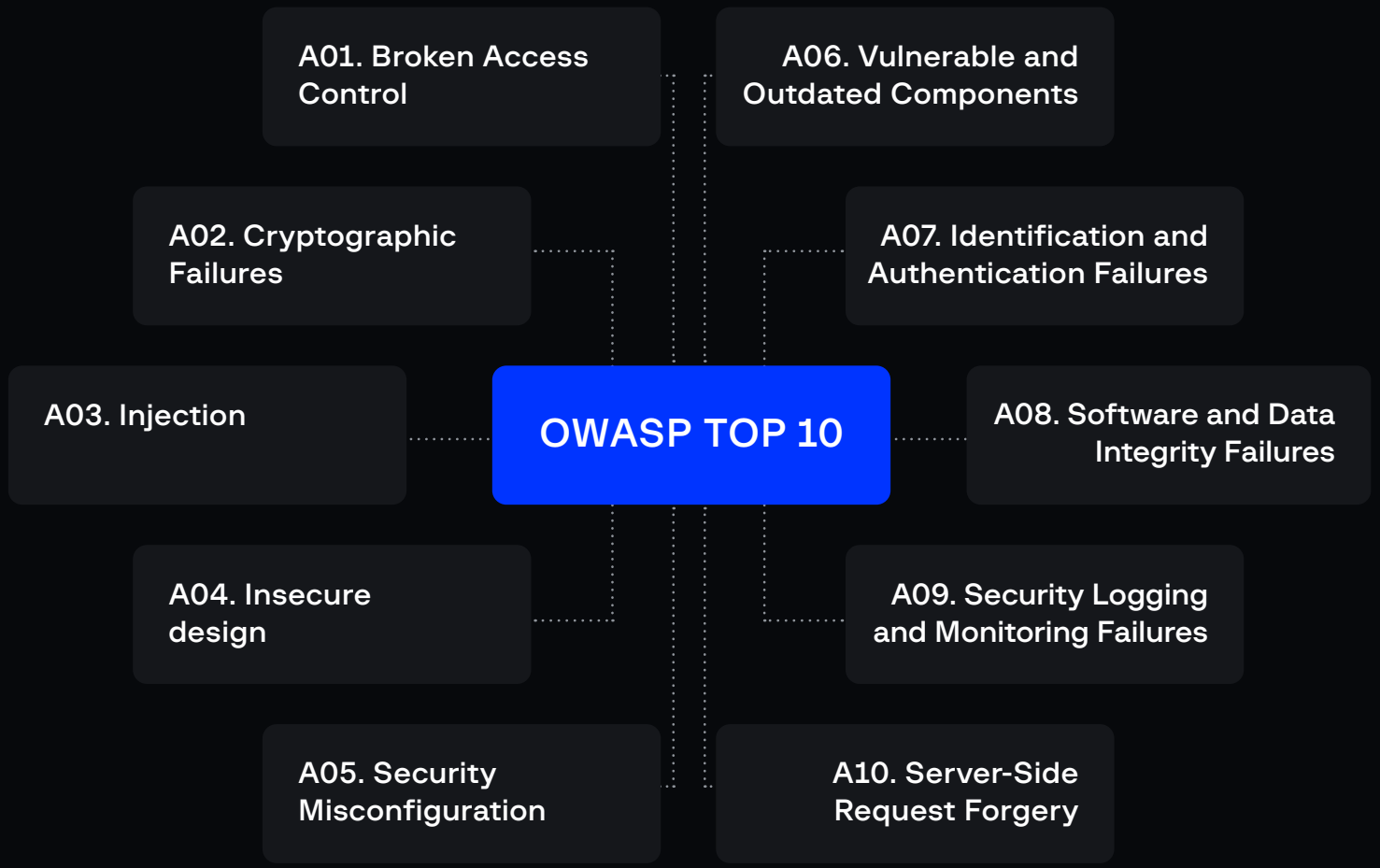


GROUP-IB

2. OWASP TOP 10, 2021: THE FUNDAMENTALS OF WEBAPP SECURITY

OWASP (Open Web Application Security Project) is a non-profit organization with a mission to enhance software security by periodically releasing an updated list of top web application security risks.

The top 10 security risks on its 2021 list remain highly relevant for mitigating common vulnerabilities in web applications today and strengthening one's cybersecurity posture against potential attacks.



How is the OWASP Top 10 list created?

Rankings are based on:

Prevalence

The frequency and occurrence of the threat.

Exploitability

Attackers usually buy a list of credentials on the dark web. The credentials are often obtained from data breaches, social engineering attacks, phishing, and malware attacks.

Detectability

The likelihood of detecting the threat.

Impact

Business or technical consequences resulting from the threat

The document serves the following purposes:

- Raise awareness about common security vulnerabilities.
- Drive improvements in encryption practices and mechanisms in software applications.
- Highlight the importance of secure code practices in the early stages of the software development life cycle (SDLC).
- Foster a security-first mindset among developers, organizations, and professionals in the community.

A01:2021: Broken Access Control

Description

Web applications need access controls to allow users (with varying privileges) to use the application. Broken Access Control is when an application fails to enforce restrictions on who can access certain functions or data. When access controls are not correctly implemented, the vulnerability means that unauthorized users are able to **bypass intended restrictions** and **gainaccess to sensitive information or perform actions beyond what's permitted.**

Broken Access Control can lead to unauthorized privilege escalation and disclosure of confidential information.

Prevention

- Enable **permission-based access control** by verifying user permissions based on job roles or other criteria to ensure that access to specific data or actions is authorized.
- Implement a **'deny-by-default' strategy** that restricts access to all resources by default, excluding public resources. This approach allows for greater control over access privileges and strengthens security.
- **Limit Cross-Origin Resource Sharing (CORS) usage**, which allows for the controlled sharing of resources between different origins. Misconfigurations can lead to vulnerabilities, however. CORS must be configured and restricted correctly in order to prevent unauthorized access and potential attacks.
- Add an extra layer of control with **user behavior analysis (UBA).**

Example

If an unauthorized user gains access to a system's administrative functions, they can add or modify sensitive data as well as add new users to the system, all of which could compromise the system's integrity.

A02:2021: Cryptographic Failures

Description

Previously listed as **“Sensitive Data Exposure”**, this vulnerability is associated with failures related to cryptography. It **exploits the information-exchange process** using cryptography, a technique used to protect the confidentiality, authenticity, and integrity of the data and/or ciphertexts shared between two parties such as passwords and sensitive business/trade information.

As the threat environment evolves, conventional obfuscation and encryption techniques are susceptible to compromise, which exposes sensitive data through various vulnerabilities called “cryptographic failures.” Security flaws that lead to cryptographic failures **include improper cryptographic key management, transferring sensitive data in easy-to-decipher text, the use of outdated cryptographic algorithms, padding methods, and deprecated hash functions.**

Prevention

- Maintain a **comprehensive inventory of all data** handled by the application.
- Periodically **review and discard any unused data** to minimize the potential risk of unauthorized access and data breaches.
- **Disable caching for sensitive data,** which prevents it from being exposed to caching vulnerabilities.
- Implement **strong and unique initialization vectors (IVs)** when using cryptographic algorithms.
- Use **up-to-date cryptographic functions, algorithms, and protocols.**
- Enforce **key rotation** to limit the potential impact of a compromised key and enhance overall security.
- Use **authenticated encryption** for both confidentiality and integrity checks.
- **Continuously detect** publicly available assets for appropriate configurations, including weak configurations or lack of cryptography.

Example

In an application, credit card numbers are stored in a database and encrypted using an automatic database encryption mechanism. When this data is retrieved from the database, however, it is decrypted automatically.

The automatic decryption creates a vulnerability, namely a SQL injection flaw, which can be exploited to retrieve credit card numbers in plain, readable text. Essentially, attackers can manipulate the application’s input to execute malicious SQL queries that bypass the encryption and retrieve sensitive credit card information directly from the database.

A03:2021: Injections

Description

Injection attacks are a wide class of attack vectors. Apart from SQL injections, there can be code injections, lightweight directory access protocols (LDAP), cross-site scripting (CSS) email header injections, OS command injections, and more. Such attacks occur when a **perpetrator submits untrusted input to an application**, which is then processed by an interpreter as part of a command or query. As a result, the perpetrator **manipulates or alters the program's execution.**

Injection attacks are among the **oldest and most threatening** vulnerabilities to modern web applications. They can have serious consequences such as data theft, data loss, compromised data integrity, denial of service, and even complete system compromise.

Prevention

- Keep **user-supplied data separate from commands and queries** executed by the application. This can be done by using proper input validation and handling techniques.
- Instead of dynamically assembling queries with user-supplied data, **use parameterized queries or prepared statements** provided by the database framework. This approach separates the query structure from the input data, thereby preventing malicious injections. If dynamic queries are unavoidable, ensure that user input is correctly escaped or sanitized to neutralize any potential injection attempts. Use the escape syntax specific to the target interpreter or database to prevent injection attacks.
- **Implement positive server-side input validation** to ensure that it conforms to the expected format, type, and length.
- Use the **"least privilege" strategy for the database accounts** in the application.
- **Regularly update and patch software**
- **Perform security testing** (including pen testing) **and code reviews** to identify and mitigate any injection vulnerabilities.
- Enable a web application firewall and **bot protection** through a preventive proxy that protects against bad bot activity, including vulnerability scanners.

Example

A web application allows users to submit feedback forms. The application processes user feedback and stores it in a database. However, the application fails to properly validate, filter and sanitize user input.

Threat actors can take advantage of this vulnerability and tamper or even remove user records/data from the database, which disrupts the application's functionality.

A04:2021: Insecure Design

Description

A new category added to the OWASP Top 10 is Insecure Design, which is a broad category that covers critical design and architectural flaws in an application. Also termed “Missing or Ineffective Control Design,” it occurs when **security controls are implemented incorrectly** and when **threats are evaluated inadequately** during the code design phase, which strips the application of its ability to defend against specific attacks.

An insecure design means that developers must use secure design patterns, planned threat modeling, and reference architectures that keep the application free of security gaps.

Prevention

- Continuously **assess and evaluate potential threats** and vulnerabilities and stay on top of best security best practices during both the application design process and the maintenance process.
- Implement **secure coding practices** to develop resilient code and deploy it securely by configuring servers properly, using encrypted connections, and protecting sensitive data.
- Use tools such as **static code analysis, vulnerability scanners, and automated testing frameworks** to identify and address security issues early in the development process.
- Conduct **threat modeling exercises** to identify and analyze risks and vulnerabilities, with a keen focus on critical flows like access control and business logic.
- Compile **use cases and misuse cases for each application tier** to identify potential abuse or exploitation of the application’s features

Example

An online ticketing platform offers discount codes for event tickets. If the application has an insecure design with improper security controls, attackers can exploit vulnerabilities and abuse the discount code feature. They might generate or guess valid discount codes even if they do not intend to use them. By repeatedly applying these unauthorized codes or sharing them with others, the attackers are able to obtain discounted or even free tickets, resulting in financial loss for the platform.

A05:2021: Security Misconfiguration

Description

Security Misconfiguration is not a fundamental security deviation in how an application is built, but a misconfiguration of the security controls available. Misconfiguration attacks exploit vulnerabilities arising from improperly configured settings in web applications or vulnerabilities due to outdated software. Typical misconfiguration vulnerabilities stem from the use of default settings without changing passwords, certificates, or installation configurations.

Other common misconfigurations include deprecated protocols and encryption methods, open database instances enabling directory listing, displaying error messages containing sensitive information, misconfigured cloud settings, and leaving unnecessary features (such as pages, ports, or command injection) enabled.

Certain developer features (such as debug and QA functionalities) are crucial during development but pose significant risks if left active in live production environments. Attackers can leverage these features to bypass authentication mechanisms and gain unauthorized access to sensitive information, potentially with elevated privileges.

Prevention

- Implement a **segmented app architecture** that provides secure separation between components and tenants.
- Ensure that all **debugging features are disabled when transitioning the application** from development to production.
- Develop an **automated hardening process** that can easily be applied to new environments in order to efficiently deploy secure environments and reduce the risk of misconfigurations.
- **Disable access to administration tools by default** and grant access to authorized users only.
- **Prevent directory listing and review permissions** assigned to separate folders and files to ensure adequate security settings.
- **Continuously detect** publicly available assets for appropriate configurations.

Example

An example of a security misconfiguration through default credentials is when a network device, such as a router or a firewall, is deployed with default usernames and passwords. Default credentials are well-known and often publicly documented, making it easy for attackers to gain unauthorized access to the device's administrative interface.

A06:2021: Vulnerable and Outdated Components

Description

Vulnerable and outdated components within an application refer to components that are no longer maintained or supported by the developers. Such components often lack the necessary security updates and patches, which leaves them exposed to vulnerabilities. To effectively address this risk, it is essential to fully understand all the components used in both the client-side and the server-side of the application. This includes libraries, frameworks, modules, plugins, and any other third-party dependencies.

An outdated, unsupported software component not only increases the attack surface but also poses a potential security gap that can be exploited by attackers. Hackers search for and exploit vulnerabilities in outdated software as they are aware that these vulnerabilities are unlikely to be patched.

Prevention

- Remove **unused dependencies and unnecessary features and components** from your application. Use the OWASP **dependency check.**
- Maintain an **inventory of components** to remain aware of their status and security implications.
- **Monitor, triage, and updateweb applications on an ongoing basis** to prevent components from becoming outdated or vulnerable.
- Leverage intrusion detection and prevention systems, log monitoring, and real-time alerting mechanisms to help detect any unauthorized activities or exploit attempts in your web application.
- Enable a web application firewall and **continuously detect** publicly available assets for appropriate configurations, including vulnerable software.

Example

An outdated and vulnerable component is the Log4j library (specifically versions prior to 2.15.0), which in December 2021 was discovered to have a critical vulnerability known as “Log4Shell” (CVE-2021-44228). This vulnerability allowed attackers to execute arbitrary code on servers running applications that used the vulnerable Log4j version.

The Log4Shell vulnerability garnered significant attention and posed a widespread threat due to the extensive use of Log4j in various software applications and systems. Exploiting the vulnerability involved sending specially crafted requests to the Log4j server and enabling remote code execution. This could have severe consequences, including full system compromise, unauthorized access to sensitive data, and data theft. The impact of the Log4Shell vulnerability was significant and it affected many organizations across various industries. It required action from system administrators, who had to either update their Log4j versions to the patched release (2.15.0) or implement mitigations to safeguard their systems.

A07:2021: Identification and Authentication Failures

Description

Ensuring that the user's identity, authentication, and session management are properly handled is essential for preventing authentication-related attacks. Authentication measures act as the front edge of the defense and can help confirm whether a user is legitimate based on their access credentials.

Authentication vulnerabilities that organizations should look out for in their applications include:

- If the application permits brute force attacks and automated attacks
- If the application permits credential stuffing (adversaries are able to obtain usernames and passwords through data leaks and dark web forums in order to launch secondary attacks)
- If the application permits weak, default, or well-known passwords
- Inadequate multi-factor authentication (MFA)
- If the application does not invalidate session IDs following prolonged inactivity or timeouts

Prevention

- Enable **multi-factor authentication.**
- **Implement the 'deny-by-default' strategy to** restrict access to all resources except public ones.
- Use **strong passwords and do not reuse passwords** for multiple accounts.
- Use **server-side built-in session managers** to make user sessions more secure.
- Leverage **Threat Intelligence** to detect compromised account access sold on the dark web and to prevent fraud through user behavior analysis and passwordless authentication.

Example

A user logs into an application but forgets to log out or close the browser window. If the application lacks appropriate session timeout settings, the user's session may remain active indefinitely, leaving it vulnerable to unauthorized access.

A08:2021: Software and Data Integrity Failures

Description

This risk arises due to inadequate integrity verification of software updates, critical data, and CI/CD pipelines. Software and data integrity failures occur when code and infrastructure lack the necessary protection against integrity violations. This could be the case when an application relies on plugins or libraries sourced from untrusted platforms, for instance.

In addition, the auto-update functionality found in many applications constitutes a vulnerability as the updates are applied to previously trusted components without integrity verification. Such failures can lead to unauthorized access, compromised data, and the introduction of malicious code.

Prevention

- Use **digital signatures or similar mechanisms** to verify that the data comes from a trusted source.
- Building and deploying code often occurs in a continuum so **ensure that CI/CD pipeline has proper integration, configuration, and access control.**
- **Introduce a review process for code and configuration** changes to minimize the risk of introducing vulnerabilities.
- **Verify your software supply chain** by carefully selecting trusted repositories and sources for obtaining libraries, dependencies, and tools (use the OWASP **dependency check**).

Example

Software and data integrity failure occurs when a computer system uses pirated or counterfeit software obtained from unreliable sources, which makes the system vulnerable to various integrity-related risks. The software may contain hidden malware or backdoors that can compromise the system's security. Additionally, these unauthorized versions often lack essential updates and patches from legitimate software vendors, leaving critical vulnerabilities unaddressed.

Attackers can exploit these vulnerabilities to gain unauthorized access, steal sensitive data, compromise the system, and even launch attacks on other systems within the network.

A09:2021: Security Logging and Monitoring Failures

Description

This risk arises due to inadequate integrity verification of software updates, critical data, and CI/CD pipelines. Software and data integrity failures occur when code and infrastructure lack the necessary protection against integrity violations. This could be the case when an application relies on plugins or libraries sourced from untrusted platforms, for instance.

In addition, the auto-update functionality found in many applications constitutes a vulnerability as the updates are applied to previously trusted components without integrity verification. Such failures can lead to unauthorized access, compromised data, and the introduction of malicious code.

Prevention

- Structure and standardize logs to facilitate easy interpretation and seamless integration with other log management solutions.
- Preserve the integrity of logs to support forensic analysis and investigations, thereby ensuring that the logged information is accurate and reliable.
- Include sufficient contextual information in logs so that malicious activities can be identified more easily.
- Introduce security measures such as access control, encryption, and regular backups to safeguard the confidentiality, integrity, and availability of log data.

Example

An e-commerce website handles a large volume of customer data, including personal information, payment details, and order history. Despite the sensitive nature of this data, the organization does not have effective logging and monitoring practices.

In such cases, attackers could exploit a vulnerability in the website's payment processing system and gain unauthorized access to customer data. Due to the lack of comprehensive logging and monitoring, however, the organization could remain unaware of the breach for a potentially long time, and customer data would continue to be compromised.

A10:2021: Server-Side Request Forgery (SSRF)

Description

SSRF is a web security vulnerability that enables attackers to manipulate a server-side application into making HTTP requests to the domains of their choice. In other words, SSRF vulnerabilities occur when an attacker has full or partial control over the request sent by the web application.

An SSRF vulnerability arises when a web application fails to properly validate user-provided URLs before fetching them to remote resources. This vulnerability allows attackers to bypass security measures such as firewalls, VPNs, and network access control lists and trick the application into sending unauthorized requests to unexpected destinations.

Prevention

- Properly **validate user input** to prevent SSRF attacks, ensuring that only **authorized hostnames or IP addresses** are accessed.
- **Do not rely on blacklists.** Instead, introduce effective input validation measures, disallowing requests to private or non-routable IP addresses.
- **Validate the "Content-Type" header of response data** to ensure that it matches the expected type, preventing potential data manipulation or injection through SSRF attacks.
- Disable **unused URL schemas**, allowing only necessary and safe schemas like HTTP or HTTPS for requests and reducing the risk of potential SSRF vulnerabilities.
- Implement **network segmentation to separate critical internal resources from public-facing servers**, limiting their exposure to potential SSRF attacks.
- Enable a web application firewall and **continuously detect** publicly available assets for appropriate configurations, including vulnerable software.

Example

A web application has the functionality to retrieve data from a designated URL based on user input. However, the application lacks proper validation and sanitization measures for the user-supplied URL.

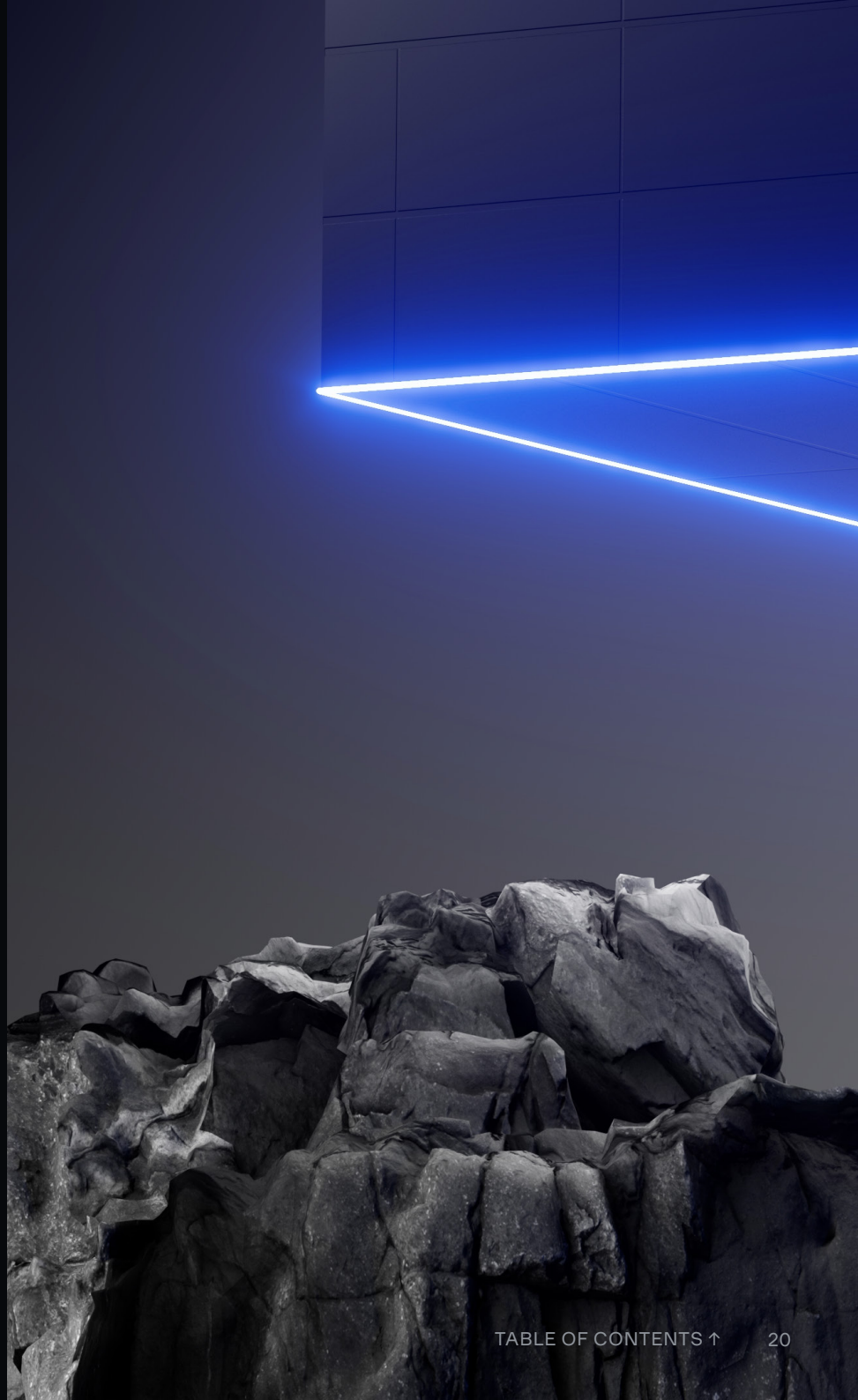
A perpetrator could exploit this vulnerability and manipulate the system by submitting a malicious URL that directs the server to access internal resources, which could include confidential databases.

GROUP-IB

3. COMBINING AUTOMATED AND MANUAL ANALYSIS FOR CRITICAL VULNERABILITY DETECTION

Although OWASP provides a comprehensive list of web application security risks, it serves as a base-level guide and may not cover all emerging threats that exist in the current landscape. The OWASP Top 10 is a standard awareness document used by information security vendors and there are vulnerability scanning tools and specialized companies that perform automated OWASP TOP 10 security analyses.

Manual analysis along with automated analysis can be crucial in identifying and eliminating critical vulnerabilities in web applications. While automated tools are often used to scan for common vulnerabilities, analytical analysis provides a more in-depth understanding of the application's architecture, business logic, and potential attack vectors.



Group-IB's audit and consulting department has encountered many instances where critical vulnerabilities were identified only through manual intervention and analysis. **Examples include:**

A vulnerability was found during a project for a US company that allowed files to beread from a server and **resources to be accessed within the local network.** The vulnerability was found in a module responsible for **rendering a PDF file from HTML.**

Incorrect processing of a regular expression in one of the tested applicationsresulted in a load of server capacity when a **deliberately invalid request was sent.** Approximately 100 such requests resulted in a denial of service, which in turn resulted in aReDoS.

One of the applications tested **allowed for files preloaded on the server to be downloaded** as an archive on which the file was unpacked. Using links in the archive made it possible to **access and download one of the sensitive files** located along the standard path.

A **Race condition vulnerability** was found while analyzing a financial application, caused due to separation of withdrawal and depositing operations, potentially leading to multiple deposits occurring between accounts within a single withdrawal transaction.

During a project in the APAC region, a **CVE-2022-44268 vulnerability** wasdiscovered in imagemagic. Exploiting the vulnerability allowed for sensitive files to be read.

Manual interventions ensure that critical vulnerabilities are not left undetected, which provides organizations with a higher level of protection against potential threats.

Let's examine a **"Local File Read" vulnerability** that automated tools failed to identify. When an expert conducts an **assessment of a "single page" web application** that includes a file upload function, the vulnerability is created due to the feature within the application that allows users to upload archives, which are then uncompressed on the remote server. Users can subsequently download the files contained within by clicking corresponding buttons in another section of the page. It's important to note that archives can contain symbolic links, which are file-system objects that point to other file-system objects.

During the assessment, the expert creates a zip archive that includes a symbolic link to a Linux password default file.

By uploading this archive to the application, the expert can exploit the flaw and download the actual password file from the remote server.

This exploit leverages the vulnerability in how the application handles symbolic links.

To illustrate an authorization issue, consider a scenario where a user account page features a button that generates a POST request. The request includes a specific identifier and a parameter for the desired reply type, with the aim of generating a user activity report. Automated tools might not flag the vulnerability if a modified value in this request allows the user to obtain an activity report from another user's account. An expert, however, would identify this as a security flaw and note that the server lacks proper authorization checks.

To achieve the most comprehensive results in an assessment, a combination of automated tools and analytical expertise is crucial. While automated tools can provide valuable insights, they might not detect certain complex attack vectors or nuanced vulnerabilities. Proficient experts, who rely on their experience and knowledge, are able to thoroughly evaluate the different aspects of an application's functions and identify potential security risks. This approach offers a holistic view of the system's security, which ensures a more comprehensive assessment.

To summarize, combining automated tools with a manual analysis by an expert allows for a more robust assessment, which helps to identify vulnerabilities that could otherwise go unnoticed. This approach provides a broader perspective and enhances the system's overall security.

Group-IB has come across numerous examples, like the ones above, of critical vulnerabilities being identified. Penetration testing (pentest) and security analyses should be carried out every year to expose vulnerabilities.

However, vulnerabilities differ from system to system. The OWASP Top 10 details the most common vulnerabilities, which could be considered the first echelon of security testing. Group-IB's audit and consulting technology and services have helped enterprises in various industries to identify vulnerabilities in their web applications. Our success lies in our established three-step process:

1	2	3
Preparation • Planning work • Collecting initial data • Analyzing internal documentation	**Examination and analysis** • Conduction interviews • Collecting audit evidence • Analyzing the data collect	**Report Generation** • Preparing a report • Drafting recommendation on how to eliminate incinsistencies

Group-IB has been a trusted partner of business worldwide for security compliance, with over 50 critical compliance checks performed every year.

Every member of our consulting team is certified auditor with extensive experience in compliance auditing for corporatioons in fields such as healthcare, financial services, critical services and maufacturing.

CYBER SECURITY EXCELLENCE AWARDS ★ WINNER ★ 2021

GDPR Compliance- Group-IB GDPR Compliance Assessment

GROUP-IB

4. GROUP-IB ATTACK SURFACE MANAGEMENT (ASM)

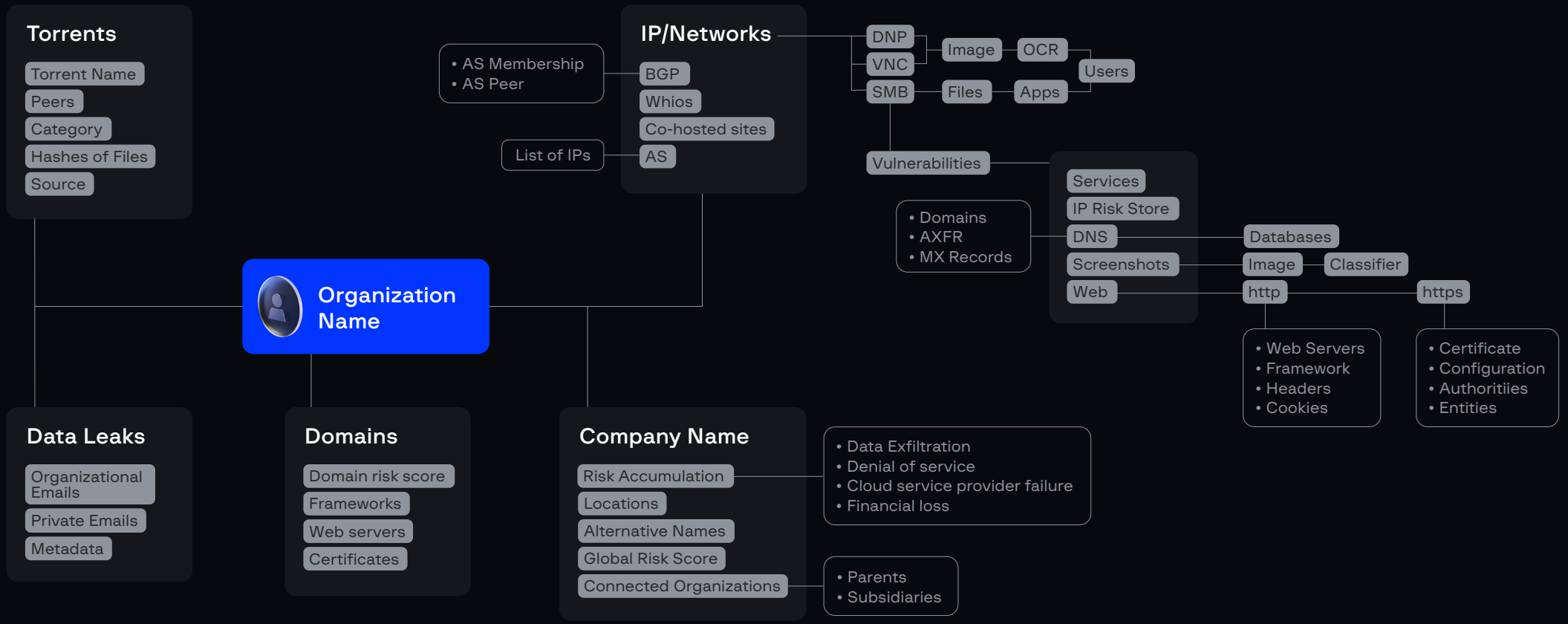
Web application security made cutting-edge and comprehensive.

Today's business landscape changes fast, which puts pressure on organizations to continually develop and deploy assets if they want to fuel their growth ambitions. However, this process often skips the step of carefully testing configuration parameters, vulnerability exposures, access control changes, keeping a log of shadow IT, etc. Neglecting such aspects can significantly expand a company's attack surface.

To ensure robust web application security, developers, security auditors, and business leaders must continuously evaluate their global internet assets as well as assess potential vulnerabilities and how likely they are to be exploited. While vulnerability scanners can help identify risk levels, their scope is often limited and the results can be either inaccurate or irrelevant. Furthermore, it can be challenging to make sense of the extensive data they generate.

TABLE OF CONTENTS ↑

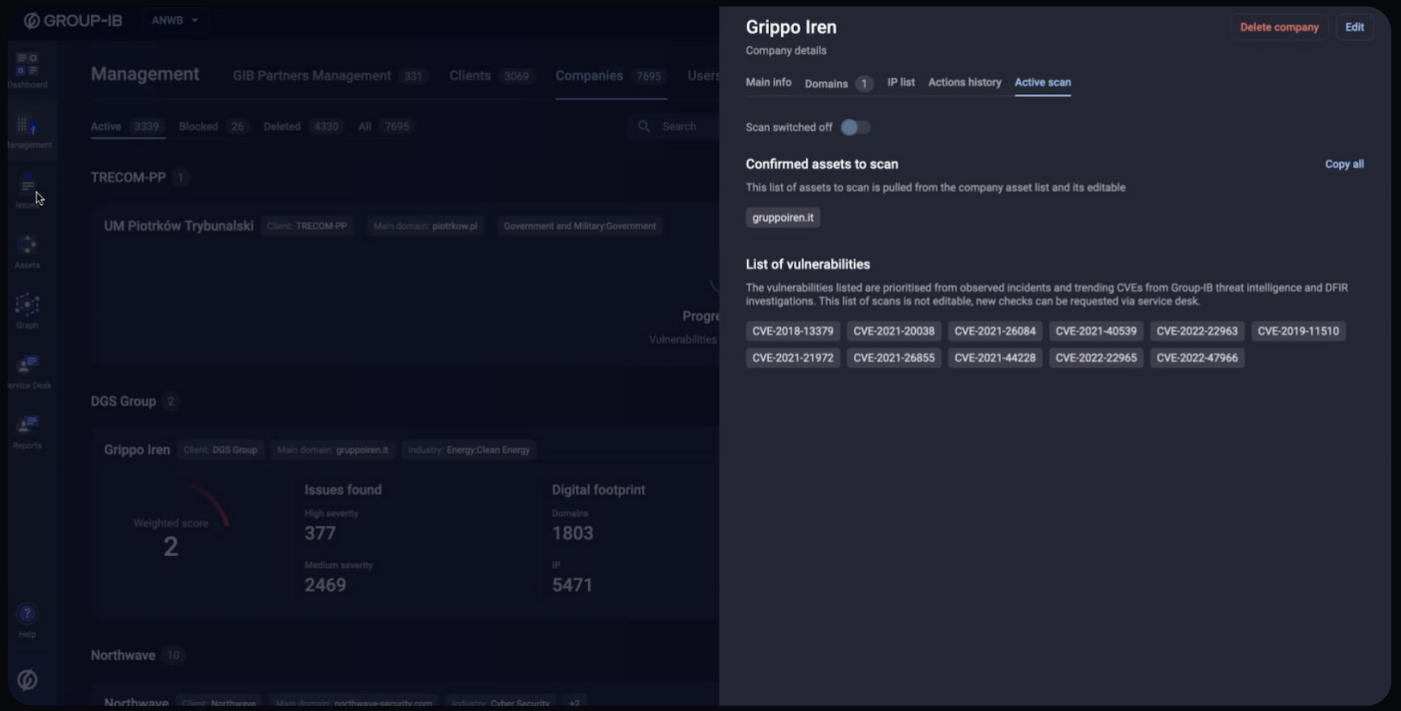
Group-IB's **Attack Surface Management** helps build a resilient and proactive stance against vulnerabilities in web applications and other assets by continuously monitoring and building actionable steps against critical risks and compliance violations across the whole infrastructure.



Attack Surface Management is designed to give security leaders a comprehensive view of their organization's security landscape. Leveraging telemetry data helps uncover near real-time insights into ongoing web application activities and potential vulnerabilities.

What insights does Group-IB Attack Surface Management offer?

Security teams today are always trying to speed up their response times while struggling with alert fatigue. Using Attack Surface Management, they can gain complete visibility of their environment, which in turn makes it possible to quickly detect threats and respond to them. Better yet, an automated response to alert triage can considerably reduce response times (within hours), thereby helping businesses save face and money in the wake of a growing tendency for cyber attacks.



Through passive or implied data scanning, our Attack Surface Management (ASM) platform helps organizations quantify the volume and severity of issues within their selected assets, allowing them to address potential avenues of exploitation.

Moreover, Group-IB's Attack Surface Management platform goes beyond passive data scanning by also offering active scanning capabilities, as part of which our team of pen testers deliberately introduce vulnerabilities into the environment to simulate potential exploits. This proactive approach helps organizations identify vulnerabilities within their workflows and offers valuable insights into the security levels of their web applications.

Through a powerful combination of both passive and active scanning techniques, the Attack Surface Management platform empowers organizations to detect and address security flaws in their assets and ensure the integrity and protection of their systems. The value and benefits of ASM do not end there.

Attack Surface Management is powered by the world's largest feed of **Threat Intelligence (TI)**, which helps businesses map threats relevant to their business and take appropriate measures against business-critical vulnerabilities. ASM ensures a high-impact remediation process.

The Group-IB ASM platform has been designed to provide an easy-to-use entry into attack surface management while doubling as a partner platform with full multi-tenancy. Intended as an affordable solution that can be used in any organization, ASM combines 'just enough' information with a robust indexation service and expansive detection logic.

The platform continuously and automatically scans an organization's external-facing assets for any threats, vulnerabilities, and configuration issues that might ease a potential attacker's path into the company's infrastructure.

At the same time, Attack Surface Management customers benefit from other sources of information. If their domain credentials show up in a leak or our specialists discover that the company's name is being discussed on an underground forum, Attack Surface Management will feed that information into the dashboard, making ASM the ideal starting point to progress into more technical cybersecurity solutions based on the organization's needs.

Attack Surface Management monitoring points:

- Corporate network, domain discovery
- Technology discovery, identification (assets, services, devices, applications, cloud services, VPNs, etc.)
- Critical vulnerabilities
- Active and passive scans for risks
- Identification of infected hosts and geolocations
- Analysis of anomalous events and network behavior

Benefits of Attack Surface Management

Improved visibility

Discover all external assets, including shadow IT, forgotten infrastructure, and misconfigurations

Continuous discovery

Automate IT asset discovery and continuously map out your organization's external attack surface

An up-to-date inventory

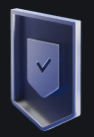
Confirm your organization's assets to generate an up-to-date IT asset inventory that keeps up with growth

Threat intelligence data

Gain insights into hidden risks like credential dumps, dark web mentions, botnets, malware, and more

Risk assessment

Check confirmed assets for common vulnerabilities & assign each one a risk score to prioritize remediation



Stronger security posture

Reduce risk and fix issues that provide measurable results for your security program

To see Group-IB Attack Surface Management in action

Reqest a demo ↗

About Group-IB

Group-IB is a leading provider of innovations and solutions for detecting and preventing cyberattacks, eliminating fraud, and protecting brands from digital risks worldwide.

1,400+
successful investigations of high-tech crimes in 60+ countries

250+
employees

650+
enterprise customers

60
countries

$1 bln
saved by our client companies through our technologies

#1*
Incident Response Retainer Provider

120+
patents and applications

17
inventors in our team

4
unique Digital Crime Resistance Centers (Europe, Asia-Pacific, Middle East)

* According to Cybersecurity Excellence Awards

Active partner in global investigations

INTERPOL

Europol

Recognized by top industry experts

FORRESTER®

kuppingercole ANALYSTS

Gartner

IDC

FROST & SULLIVAN

Technologies and innovations

Cybersecurity

- Threat intelligence
- Attack surface management
- Email protection
- Network traffic analysis
- Malware detonation
- EDR
- XDR

Anti-fraud

- Client-side anti-fraud
- Adaptive authentication
- Bot prevention
- Fraud intelligence
- User and entity behavior analysis

Brand protection

- Anti-phishing
- Anti-piracy
- Anti-scam
- Anti-counterfeit
- Protection from data leaks
- VIP protection

Intelligence-driven services

Audit & Consulting	• Security Assessment • Penetration Testing • Red Teaming • Compliance & Consulting
Education & Training	• For technical specialists • For wider audiences
DFIR	• Incident Response • Incident Response Retainer • Incident Response Readiness Assessment • Compromise Assessment • Digital Forensics • eDiscovery
Managed Services	• Managed Detection • Managed Threat Hunting • Managed Response
High-Tech Crime Investigation	• Cyber Investigation • Investigation Subscription

GROUP-IB

Preventing and investigating cybercrime since 2003

FIGHT AGAINST CYBERCRIME

GROUP-IB.COM
INFO@GROUP-IB.COM

APAC
+65 3159 3798

EU & NA
+31 20 226 90 90

MEA
+971 4 508 1605