# Logging Cheat Sheet - OWASP

Originalartikel

Backup

<html> <tr readability=„355“><td valign=„top“ readability=„258“> <p>Last revision (mm/dd/yy): <b>01/20/2016</b> </p> <div id=„toc“ class=„toc“> <ul><li class=„toclevel-1 tocsection-1“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Introduction“>1 Introduction</a></li> <li class=„toclevel-1 tocsection-2“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Purpose“>2 Purpose</a></li> <li class=„toclevel-1 tocsection-3“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Design.2C_implementation_and_testing“>3 Design, implementation and testing</a> <ul><li class=„toclevel-2 tocsection-4“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Event_data_sources“>3.1 Event data sources</a></li> <li class=„toclevel-2 tocsection-5“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Where_to_record_event_data“>3.2 Where to record event data</a></li> <li class=„toclevel-2 tocsection-6“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Which_events_to_log“>3.3 Which events to log</a></li> <li class=„toclevel-2 tocsection-7“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Event_attributes“>3.4 Event attributes</a></li> <li class=„toclevel-2 tocsection-8“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Data_to_exclude“>3.5 Data to exclude</a></li> <li class=„toclevel-2 tocsection-9“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Customizable_logging“>3.6 Customizable logging</a></li> <li class=„toclevel-2 tocsection-10“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Event_collection“>3.7 Event collection</a></li> <li class=„toclevel-2 tocsection-11“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Verification“>3.8 Verification</a></li> </ul></li> <li class=„toclevel-1 tocsection-12“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Deployment_and_operation“>4 Deployment and operation</a> <ul><li class=„toclevel-2 tocsection-13“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Release“>4.1 Release</a></li> <li class=„toclevel-2 tocsection-14“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Operation“>4.2 Operation</a></li> <li class=„toclevel-2 tocsection-15“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Protection“>4.3 Protection</a></li> <li class=„toclevel-2 tocsection-16“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Monitoring_of_events“>4.4 Monitoring of events</a></li> <li class=„toclevel-2 tocsection-17“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Disposal_of_logs“>4.5 Disposal of logs</a></li> </ul></li> <li class=„toclevel-1 tocsection-18“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Related_articles“>5 Related articles</a></li> <li class=„toclevel-1 tocsection-19“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Authors_and_Primary_Contributors“>6 Authors and Primary Contributors</a> <ul><li class=„toclevel-2 tocsection-20“><a href=„https://www.owasp.org/index.php/Logging_Cheat_Sheet#Other_Cheatsheets“>6.1 Other Cheatsheets</a></li> </ul></li> </ul></div> <p>This cheat sheet is focused on providing developers with concentrated guidance on building application logging mechanisms, especially related to security logging. Many systems enable network device, operating system, web server, mail

server and database server logging, but often custom application event logging is missing, disabled or poorly configured. It provides much greater insight than infrastructure logging alone. Web application (e.g. web site or web service) logging is much more than having web server logs enabled (e.g. using Extended Log File Format). </p><p>Application logging should be consistent within the application, consistent across an organization's application portfolio and use industry standards where relevant, so the logged event data can be consumed, correlated, analyzed and managed by a wide variety of systems. </p> <p>Application logging should be always be included for security events. Application logs are invaluable data for: </p> <ul><li> Identifying security incidents</li> <li> Monitoring policy violations</li> <li> Establishing baselines</li> <li> Assisting non-repudiation controls</li> <li> Providing information about problems and unusual conditions</li> <li> Contributing additional application-specific data for incident investigation which is lacking in other log sources</li> <li> Helping defend against vulnerability identification and exploitation through attack detection</li></ul><p>Application logging might also be used to record other types of events too such as: </p> <ul><li> Security events</li> <li> Business process monitoring e.g. sales process abandonment, transactions, connections</li> <li> Anti-automation monitoring</li> <li> Audit trails e.g. data addition, modification and deletion, data exports</li> <li> Performance monitoring e.g. data load time, page timeouts</li> <li> Compliance monitoring</li> <li> Data for subsequent requests for information e.g. data subject access, freedom of information, litigation, police and other regulatory investigations</li> <li> Legally sanctioned interception of data e.g application-layer wire-tapping</li> <li> Other business-specific requirements</li></ul><p>Process monitoring, audit and transaction logs/trails etc are usually collected for different purposes than security event logging, and this often means they should be kept separate. The types of events and details collected will tend to be different. For example a PCIDSS audit log will contain a chronological record of activities to provide an independently verifiable trail that permits reconstruction, review and examination to determine the original sequence of attributable transactions. It is important not to log too much, or too little. Use knowledge of the intended purposes to guide what, when and how much. The remainder of this cheat sheet primarily discusses security event logging. </p> <h2>Event data sources</h2> <p>The application itself has access to a wide range of information events that should be used to generate log entries. Thus, the primary event data source is the application code itself. The application has the most information about the user (e.g. identity, roles, permissions) and the context of the event (target, action, outcomes), and often this data is not available to either infrastructure devices, or even closely-related applications. </p><p>Other sources of information about application usage that could also be considered are: </p> <ul><li> Client software e.g. actions on desktop software and mobile devices in local logs or using messaging technologies, JavaScript exception handler via Ajax, web browser such as using Content Security Policy (CSP) reporting mechanism</li> <li> Embedded instrumentation code</li> <li> Network firewalls</li> <li> Network and host intrusion detection systems (NIDS and HIDS)</li> <li> Closely-related applications e.g. filters built into web server software, web server URL redirects/rewrites to scripted custom error pages and handlers</li> <li> Application firewalls e.g. filters, guards, XML gateways, database firewalls, web application firewalls (WAFs)</li> <li> Database applications e.g. automatic audit trails, trigger-based actions</li> <li> Reputation monitoring services e.g. uptime or malware monitoring</li> <li> Other applications e.g. fraud monitoring, CRM</li> <li> Operating system e.g. mobile platform</li></ul><p>The degree of confidence in the event information has to be considered when including event data from systems in a different trust zone. Data may be missing, modified, forged, replayed and could be malicious &#8211; it must always be treated as untrusted data. Consider how the source can be verified, and how integrity and non-repudiation can be enforced. </p> <h2>Where to record event data</h2> <p>Applications commonly write event log data to the file system or a database (SQL or NoSQL). Applications installed on desktops and on mobile devices may use local storage and local databases, as well as sending data to remote storage. Your selected framework may limit the available choices. All types of applications may send event data to remote systems (instead of or as well as more local storage). This could be a centralized log collection and management system (e.g. SIEM or SEM) or another application elsewhere. Consider whether the application can simply send its event stream, unbuffered, to stdout, for management by the execution environment. </p> <ul><li> When using the file system, it is preferable to use a separate partition than those used by the operating system, other application files and user generated content <ul><li> For file-based logs, apply strict permissions concerning which users can access the directories, and the permissions of files within the directories</li> <li> In web applications, the logs should not be exposed in web-accessible locations, and if done so, should have restricted access and be configured with a plain text MIME type (not HTML)</li></ul></li> <li> When using a database, it is preferable to utilize a separate database account that is only used for writing log data and which has very restrictive database , table, function and command permissions</li> <li> Use standard formats over secure protocols to record and send event data, or log files, to other systems e.g. Common Log File System (CLFS), Common Event Format (CEF) over syslog, possibly Common Event Expression (CEE) in future; standard formats facilitate integration with centralised logging services</li></ul><p>Consider separate files/tables for extended event information such as error stack traces or a record of HTTP request and response headers and bodies. </p> <h2>Which events to log</h2> <p>The level and content of security monitoring, alerting and reporting needs to be set during the requirements and design stage of projects, and should be proportionate to the information security risks. This can then be used to define what should be logged. There is no one size fits all solution, and a blind checklist approach can lead to unnecessary „alarm fog“ that means real problems go undetected. Where possible, always log: </p> <ul><li> Input validation failures e.g. protocol violations, unacceptable encodings, invalid parameter names and values</li> <li> Output validation failures e.g. database record set mismatch, invalid data encoding</li> <li> Authentication successes and failures</li> <li> Authorization (access control) failures</li> <li> Session management failures e.g. cookie session identification value modification</li> <li> Application errors and system events e.g. syntax and runtime errors, connectivity problems, performance issues, third party service error messages, file system errors, file upload virus detection, configuration changes</li> <li> Application and related systems start-ups and shut-downs, and logging initialization (starting, stopping or pausing)</li> <li> Use of higher-risk functionality e.g. network connections, addition or deletion of users, changes to privileges, assigning users to tokens, adding or deleting tokens, use of systems administrative privileges, access by application administrators,all actions by users with administrative privileges, access to payment cardholder data, use of data encrypting keys, key changes, creation and deletion of system-level objects, data import and export including screen-based reports, submission of user-generated content - especially file uploads</li> <li> Legal and other opt-ins e.g. permissions for mobile phone capabilities, terms of use, terms &amp; conditions, personal data usage consent, permission to receive marketing communications</li></ul><p>Optionally consider if the following events can be logged and whether it is desirable information: </p> <ul><li> Sequencing failure</li> <li> Excessive use</li> <li> Data changes</li> <li> Fraud and other criminal activities</li> <li> Suspicious, unacceptable or unexpected behavior</li> <li> Modifications to configuration</li> <li> Application code file and/or memory changes</li></ul><h2>Event attributes</h2> <p>Each log entry needs to include sufficient information for the intended subsequent monitoring and analysis. It could be full content data, but is more likely to be an extract or just summary properties. The application logs must record „when, where, who and what“ for each event. The properties for these will be different depending on the architecture, class of application and host system/device, but often include the following: </p> <ul><li> When <ul><li> Log date and time (international format)</li> <li> Event date and time - the event time stamp may be different to the time of logging e.g. server logging where the client application is hosted on remote device that is only periodically or intermittently online</li> <li> Interaction identifier [Note A]</li></ul></li> <li> Where <ul><li> Application identifier e.g. name and version</li> <li> Application address e.g. cluster/host name or server IPv4 or IPv6 address and port number, workstation identity, local device identifier</li> <li> Service e.g. name and protocol</li> <li> Geolocation</li> <li> Window/form/page e.g. entry point

URL and HTTP method for a web application, dialogue box name</li> <li> Code location e.g. script name, module name</li></ul></li> <li> Who (human or machine user) <ul><li> Source address e.g. user's device/machine identifier, user's IP address, cell/RF tower ID, mobile telephone number</li> <li> User identity (if authenticated or otherwise known) e.g. user database table primary key value, user name, license number</li></ul></li> <li> What <ul><li> Type of event [Note B]</li> <li> Severity of event [Note B] e.g. {0=emergency, 1=alert, ..., 7=debug}, {fatal, error, warning, info, debug, trace}</li> <li> Security relevant event flag (if the logs contain non-security event data too)</li> <li> Description</li></ul></li></ul><p>Additionally consider recording: </p> <ul><li> Secondary time source (e.g. GPS) event date and time</li> <li> Action - original intended purpose of the request e.g. Log in, Refresh session ID, Log out, Update profile</li> <li> Object e.g. the affected component or other object (user account, data resource, file) e.g. URL, Session ID, User account, File</li> <li> Result status - whether the ACTION aimed at the OBJECT was successful e.g. Success, Fail, Defer</li> <li> Reason - why the status above occurred e.g. User not authenticated in database check ..., Incorrect credentials</li> <li> HTTP Status Code (web applications only) - the status code returned to the user (often 200 or 301)</li> <li> Request HTTP headers or HTTP User Agent (web applications only)</li> <li> User type classification e.g. public, authenticated user, CMS user, search engine, authorized penetration tester, uptime monitor (see „Data to exclude“ below)</li> <li> Analytical confidence in the event detection [Note B] e.g. low, medium, high or a numeric value</li> <li> Responses seen by the user and/or taken by the application e.g. status code, custom text messages, session termination, administrator alerts</li> <li> Extended details e.g. stack trace, system error messages, debug information, HTTP request body, HTTP response headers and body</li> <li> Internal classifications e.g. responsibility, compliance references</li> <li> External classifications e.g. NIST Security Content Automation Protocol (SCAP), Mitre Common Attack Pattern Enumeration and Classification (CAPEC)</li></ul><p>For more information on these, see the „other“ related articles listed at the end, especially the comprehensive article by Anton Chuvakin and Gunnar Peterson. </p><p>Note A: The „Interaction identifier“ is a method of linking all (relevant) events for a single user interaction (e.g. desktop application form submission, web page request, mobile app button click, web service call). The application knows all these events relate to the same interaction, and this should be recorded instead of losing the information and forcing subsequent correlation techniques to re-construct the separate events. For example a single SOAP request may have multiple input validation failures and they may span a small range of times. As another example, an output validation failure may occur much later than the input submission for a long-running „saga request“ submitted by the application to a database server. </p><p>Note B: Each organisation should ensure it has a consistent, and documented, approach to classification of events (type, confidence, severity), the syntax of descriptions, and field lengths &amp; data types including the format used for dates/times. </p> <h2>Data to exclude</h2> <p>Never log data unless it is legally sanctioned. For example intercepting some communications, monitoring employees, and collecting some data without consent may all be illegal. </p><p>Never exclude any events from „known“ users such as other internal systems, „trusted“ third parties, search engine robots, uptime/process and other remote monitoring systems, pen testers, auditors. However, you may want to include a classification flag for each of these in the recorded data. </p><p>The following should not usually be recorded directly in the logs, but instead should be removed, masked, sanitized, hashed or encrypted: </p> <ul><li> Application source code</li> <li> Session identification values (consider replacing with a hashed value if needed to track session specific events)</li> <li> Access tokens</li> <li> Sensitive personal data and some forms of personally identifiable information (PII) e.g. health, government identifiers, vulnerable people</li> <li> Authentication passwords</li> <li> Database connection strings</li> <li> Encryption keys and other master secrets</li> <li> Bank account or payment card holder data</li> <li> Data of a higher security classification than the logging system is allowed to store</li> <li> Commercially-sensitive information</li> <li> Information it is illegal to collect in the relevant jurisdictions</li> <li> Information a user has opted out of collection, or not consented to e.g. use of do not track, or where consent to collect has expired</li></ul><p>Sometimes the following data can also exist, and whilst useful for subsequent investigation, it may also need to be treated in some special manner before the event is recorded: </p> <ul><li> File paths</li> <li> Database connection strings</li> <li> Internal network names and addresses</li> <li> Non sensitive personal data (e.g. personal names, telephone numbers, email addresses)</li></ul><p>Consider using personal data de-identification techniques such as deletion, scrambling or pseudonymization of direct and indirect identifiers where the individual's identity is not required, or the risk is considered too great. </p><p>In some systems, sanitization can be undertaken post log collection, and prior to log display. </p> <h2>Customizable logging</h2> <p>It may be desirable to be able to alter the level of logging (type of events based on severity or threat level, amount of detail recorded). If this is implemented, ensure that: </p> <ul><li> The default level must provide sufficient detail for business needs</li> <li> It should not be possible to completely inactivate application logging or logging of events that are necessary for compliance requirements</li> <li> Alterations to the level/extent of logging must be intrinsic to the application (e.g. undertaken automatically by the application based on an approved algorithm) or follow change management processes (e.g. changes to configuration data, modification of source code)</li> <li> The logging level must be verified periodically</li></ul><h2>Event collection</h2> <p>If your development framework supports suitable logging mechanisms use, or build upon that. Otherwise, implement an application-wide log handler which can be called from other modules/components. Document the interface referencing the organisation-specific event classification and description syntax requirements. If possible create this log handler as a standard module that can is thoroughly tested, deployed in multiple application, and added to a list of approved &amp; recommended modules. </p> <ul><li> Perform input validation on event data from other trust zones to ensure it is in the correct format (and consider alerting and not logging if there is an input validation failure)</li> <li> Perform sanitization on all event data to prevent log injection attacks e.g. carriage return (CR), line feed (LF) and delimiter characters (and optionally to remove sensitive data)</li> <li> Encode data correctly for the output (logged) format</li> <li> If writing to databases, read, understand and apply the SQL injection cheat sheet</li> <li> Ensure failures in the logging processes/systems do not prevent the application from otherwise running or allow information leakage</li> <li> Synchronize time across all servers and devices [Note C]</li></ul><p>Note C: This is not always possible where the application is running on a device under some other party's control (e.g. on an individual's mobile phone, on a remote customer's workstation which is on another corporate network). In these cases attempt to measure the time offset, or record a confidence level in the event time stamp. </p><p>Where possible record data in a standard format, or at least ensure it can be exported/broadcast using an industry-standard format. </p><p>In some cases, events may be relayed or collected together in intermediate points. In the latter some data may be aggregated or summarized before forwarding on to a central repository and analysis system. </p> <h2>Verification</h2> <p>Logging functionality and systems must be included in code review, application testing and security verification processes: </p> <ul><li> Ensure the logging is working correctly and as specified</li> <li> Check events are being classified consistently and the field names, types and lengths are correctly defined to an agreed standard</li> <li> Ensure logging is implemented and enabled during application security, fuzz, penetration and performance testing</li> <li> Test the mechanisms are not susceptible to injection attacks</li> <li> Ensure there are no unwanted side-effects when logging occurs</li> <li> Check the effect on the logging mechanisms when external network connectivity is lost (if this is usually required)</li> <li> Ensure logging cannot be used to deplete system resources, for example by filling up disk space or exceeding database transaction log space, leading to denial of service</li> <li> Test the effect on the application of logging failures such as simulated database connectivity loss, lack of file system space, missing write permissions to the file system, and runtime errors in the logging module itself</li> <li> Verify access controls on the event log data</li> <li> If log data is utilized in any action against users (e.g. blocking access, account lock-out), ensure this cannot be used to cause denial of service (DoS) of other users</li></ul> <h2>Release</h2> <ul><li> Provide security configuration information by adding details about the logging mechanisms to release documentation</li> <li> Brief the application/process owner about the application logging mechanisms</li> <li> Ensure the outputs of the monitoring (see below) are integrated with incident response processes</li></ul><h2>Operation</h2> <p>Enable processes to detect whether logging has stopped, and to identify tampering or unauthorized access and deletion (see protection below). </p> <h2>Protection</h2> <p>The logging mechanisms and collected event data must be protected from mis-use such as tampering in transit, and unauthorized access, modification and deletion once stored. Logs may contain personal and other sensitive information, or the data may contain information regarding the application's code and logic. In addition, the collected information in the logs may itself have business value (to competitors, gossip-mongers, journalists and activists) such as allowing the estimate of revenues, or providing performance information about employees. This data may be held on end devices, at intermediate points, in centralized repositories and in archives and backups. Consider whether parts of the data may need to be excluded, masked, sanitized, hashed or encrypted during examination or extraction. </p><p>At rest: </p> <ul><li> Build in tamper detection so you know if a record has been modified or deleted</li> <li> Store or copy log data to read-only media as soon as possible</li> <li> All access to the logs must be recorded and monitored (and may need prior approval)</li> <li> The privileges to read log data should be restricted and reviewed periodically</li></ul><p>In transit: </p> <ul><li> If log data is sent over untrusted networks (e.g. for collection, for dispatch elsewhere, for analysis, for reporting), use a secure transmission protocol</li> <li> Consider whether the origin of the event data needs to be verified</li> <li> Perform due diligence checks (regulatory and security) before sending event data to third parties</li></ul><p>See NIST SP 800-92 Guide to Computer Security Log Management for more guidance. </p> <h2>Monitoring of events</h2> <p>The logged event data needs to be available to review and there are processes in place for appropriate monitoring, alerting and reporting: </p> <ul><li> Incorporate the application logging into any existing log management systems/infrastructure e.g. centralized logging and analysis systems </li> <li> Ensure event information is available to appropriate teams</li> <li> Enable alerting and signal the responsible teams about more serious events immediately</li> <li> Share relevant event information with other detection systems, to related organizations and centralized intelligence gathering/sharing systems</li></ul><h2>Disposal of logs</h2> <p>Log data, temporary debug logs, and backups/copies/extractions, must not be destroyed before the duration of the required data retention period, and must not be kept beyond this time. Legal, regulatory and contractual obligations may impact on these periods. </p> <p>OWASP <a rel=„nofollow“ class=„external text“ href=„http://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API“>ESAPI Documentation</a> </p><p>OWASP <a rel=„nofollow“ class=„external text“ href=„https://www.owasp.org/index.php/Category:OWASP_Logging_Project“>Logging Project</a> </p><p>IETF <a rel=„nofollow“ class=„external text“ href=„http://tools.ietf.org/html/rfc5424“>syslog protocol</a> </p><p>Mitre <a rel=„nofollow“ class=„external text“ href=„http://cee.mitre.org/“>Common Event Expression (CEE)</a> </p><p>NIST <a rel=„nofollow“ class=„external text“ href=„http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf“>SP 800-92 Guide to Computer Security Log Management</a> </p><p>PCISSC <a rel=„nofollow“ class=„external text“ href=„https://www.pcisecuritystandards.org/security_standards/documents.php“>PCI DSS v2.0 Requirement 10 and PA-DSS v2.0 Requirement 4</a> </p><p>W3C <a rel=„nofollow“ class=„external text“ href=„http://www.w3.org/TR/WD-logfile.html“>Extended Log File Format</a> </p><p>Other <a rel=„nofollow“ class=„external text“ href=„http://arctecgroup.net/pdf/howtoapplogging.pdf“>How to Do Application Logging Right, Anton Chuvakin &amp; Gunnar Peterson, IEEE Security &amp; Privacy Journal</a> </p><p>Other <a rel=„nofollow“ class=„external text“

href=„http://taosecurity.blogspot.co.uk/2009/08/build-visibility-in.html“>Build Visibility In, Richard Bejtlich, TaoSecurity blog</a> </p><p>Other <a rel=„nofollow“ class=„external text“ href=„http://www.arcsight.com/solutions/solutions-cef/“>Common Event Format (CEF), Arcsight</a> </p><p>Other <a rel=„nofollow“ class=„external text“ href=„http://www.clerkendweller.com/2010/8/17/Application-Security-Logging“>Application Security Logging, Colin Watson, Web Security Usability and Design Blog</a> </p><p>Other <a rel=„nofollow“ class=„external text“ href=„http://msdn.microsoft.com/en-us/library/windows/desktop/bb986747(v=vs.85).aspx“>Common Log File System (CLFS), Microsoft</a> </p><p>Other <a rel=„nofollow“ class=„external text“ href=„http://www.symantec.com/connect/articles/building-secure-applications-consistent-logging“>Building Secure Applications: Consistent Logging, Rohit Sethi &amp; Nish Bhalla, Symantec Connect</a> </p> <p>Most of the information included is based on content in the articles referenced in the text and listed above, but the following people have provided their ideas, knowledge and practical experience: </p><p>Colin Watson - colin.watson[at]owasp.org<br/>Eoin Keary - eoin.keary[at]owasp.org<br/>Alexis Fitzgerald - alexis.fitzgerald[at]owasp.org </p> <h2>Other Cheatsheets</h2> </td></tr> </html>