HashEye

// SMART CONTRACT SECURITY AUDIT

Eclipse Mosquitto

Security assessment by HashEye · prepared for AppSec

HASHEYE AUDITED

PROJECT	Eclipse Mosquitto
CLIENT	AppSec
CATEGORY	Blockchain
PUBLISHED	March 1, 2023
REPORT ID	research-eclipse-mosquitto-2023-03-01-kg9l83

This report was produced under HashEye's layered review process — **automated detection**, **pattern correlation**, and **senior manual verification** — with every finding signed off by a human reviewer. Full findings detail and on-chain attestation are available on the report page at hasheye.io/audits/research-eclipse-mosquitto-2023-03-01-kg9l83.

Eclipse Mosquitto Threat Model February 24, 2023 Prepared for: Eclipse Foundation Organized by Open Source Technology Improvement Fund, Inc. Prepared by: Kelly Kaoudis, Shaun Mirani, Spencer Michaels

About HashEye Founded in 2012 and headquartered in New York, HashEye provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high- end security research with a real -world attacker mentality to reduce risk and fortify code. With 80+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel. We maintain an exhaustive list of publications at https://github.com/hasheye-io/publications , with links to papers, presentations, public audit reports, and podcast appearances. In recent years, HashEye consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon. We specialize in software testing and code review projects, supporting client organizations in the technology, defense, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom. HashEye also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash. To keep up to date with our latest news and announcements, please follow hasheye on Twitter and explore our public repositories at https://github.com/hasheye-io . To engage us directly, visit our "Contact" page at https://www.hasheye.io/contact , or email us at info@hasheye.io . HashEye, Inc. 228 Park Ave S #80688 New York, NY 10003 https://www.hasheye.io info@hasheye.io HashEye 1 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

 Analysis Coverage Disclaimer All activities undertaken by HashEye in association with this project were performed in accordance with a statement of work and agreed upon project plan. Threat modeling projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase. HashEye 2 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Table of Contents HashEye 3 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Executive Summary Engagement Overview OSTIF engaged HashEye to conduct a lightweight threat model of the Eclipse Mosquitto project. From February 13 to February 17, 2023, a team of three consultants met with the client with three person-weeks of eff ort to evaluate relevant components and system architecture , drawing from the Mozilla "Rapid Risk Assessment" methodology and the National Institute of Standards and Technology's (NIST) guidance on data-centric threat modeling (NIST 800-154). Details of the project's timeline, test targets, and coverage are provided in subsequent sections of this report. Project Scope Our assessment focused on the identification of security control flaws that could result in a compromise of confidentiality, integrity, or availability of the target system, especially with respect to the controls noted in the category breakdown table below. An exhaustive list of security control types and their defi nitions can be found in appendix B . Summary of Findings The audit uncovered flaws impacting system confidentiality, integrity, and availability. A summary of the findings and details on notable findings are provided below. FINDINGS BY SEVERITY Severity Count High 5 Medium 2 Low 1 FINDINGS BY

CONTROL TYPE Category Count Access Controls 4 Denial of Service 2 System and Information Integrity 2 HashEye 4 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Notable Findings Significant security control flaws that impact system confidentiality, integrity, or availability are listed below. • TOB-MOSQ-3 Since there is no clear broker-global way to configure and enforce rate limiting for client or bridge connection and authentication attempts, an attacker could brute-force the password(s) of one or more users, potentially resulting in denial of service to other broker clients. Having authenticated with the brute-forced credentials, the attacker would then have the ability to publish to and consume any topics for which the credentials can legitimately be used, including $CONTROL or $SYS topics. • TOB-MOSQ-5 Since there is no functionality to prevent infinitely looping messages between bridged brokers, an attacker could bridge a malicious broker A to a broker they wish to overwhelm, B, with the intention of exhausting B's system resources and B's clients' resources. • TOB-MOSQ-8 Since there are no restrictions on publishing "last will and testament" (LWT) messages to Dynamic Security plugin $CONTROL topics or those of any other plugin, an attacker could set the LWT for a client they control to alter the broker security-related plugin configuration on their behalf when the LWT is published, after the attacker's client has lost ACL permissions to make such changes directly. HashEye 5 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Project Summary Contact Information The following project manager was associated with this project: Jeff Braswell , Project Manager jeff .braswell@hasheye.io The following engineering director was associated with this project: Anders Helsing , Engineering Director, Application Security anders.helsing@hasheye.io The following consultants were associated with this project: Kelly Kaoudis, Consultant Shaun Mirani, Consultant kelly.kaoudis@hasheye.io shaun.mirani@hasheye.io Spencer Michaels, Consultant spencer.michaels@hasheye.io Project Timeline The significant events and milestones of the project are listed below. Date Event February 9, 2023 Pre-project kickoff call February 14, 2023 Discovery meeting #1 February 16, 2023 Discovery meeting #2 February 23, 2023 Delivery of report draft and report readout meeting March 24, 2023 Delivery of final report HashEye 6 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Project Coverage During a lightweight threat modeling assessment, engineers generally aim to cover the entire target system as a coherent whole. In some cases, however, certain components may be either unnecessary to examine, or impossible to review thoroughly. Security Controls The following security controls were used to evaluate the project targets during threat modeling exercises. For further information regarding security controls, see appendix B . • Access Controls • Audit and Accountability • Cryptography • Denial of Service • Identification and Authentication • System and Information Integrity Exclusions We explicitly excluded the following components from the assessment scope: • mosquitto-go-auth , a third-party, Go-based alternative to Mosquitto's Dynamic Security plugin • HAProxy , which explicitly supports MQTT and is commonly deployed with Mosquitto in production environments, but is not part of the Mosquitto project • The Certifi cate Authority , which is an end user–controlled component and not part of the Mosquitto project • The openssl command-line utility, which is referred to throughout the Mosquitto documentation as the preferred way to create certificates and other cryptographic data, but is not itself part of the Mosquitto project • Any SSL/TLS library with which the client developer or infrastructure administrator configures a Mosquitto broker or client, though Mosquitto's usage of such libraries is explicitly in scope HashEye 7 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Limitations Because of the time-boxed nature of testing work, it is common to encounter coverage limitations. During this project, we were unable to perform comprehensive review of the following areas, which may warrant further review: • Implementation details of SSL/TLS library usage , which will be explored during the secure code review portion of this assessment • Deployment functionality , including the following: ○ Platform-specific functionality related to installation or packaging ○ Mosquitto Dockerfile environments HashEye 8 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

System Diagrams The following diagrams depict the relationships between Mosquitto's various components and trust zones, as well as the potential paths that threat actors can take within them. Data Types Generally, the Mosquitto broker and ctrl utilities communicate via publishing to and subscribing to MQTT topics. There are no restrictions on the format or types of data that can be communicated over MQTT. The Mosquitto broker and ctrl utilities also consume and edit configuration files in the local filesystem. High-Level Data Flow HashEye 9 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Local Data Flow HashEye 10 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Components Eclipse Mosquitto is an open-source MQTT 5, 3.1.1, and 3.1 broker implementation packaged with client, client-library, and broker-plugin foundation code. The following table

describes each Mosquitto component or external dependency and notes through an asterisk (*) whether the component or dependency is not in scope. We explored the implications of threats involving out of scope components which directly affect in-scope components, but we do not consider threats to out of scope components. Component Description Broker This is the Mosquitto MQTT broker service. Dynamic Security broker plugin This is the Mosquitto default authentication/authorization plugin that enforces the access-control and authorization rules defi ned in configuration files stored on the broker host's filesystem. mosquitto-go-auth (*) This is the third-party Mosquitto authentication/authorization plugin that is compatible with a variety of data sources and formats, such as mysql, Redis, and JWT. This component was out of scope. Bridge When a broker connects to another broker to send, receive, or bidirectionally exchange reproduced messages and/or topics, the connection is called a bridge. Infrastructure administrators commonly build trees of brokers using bridges. HAProxy (*) This is an MQTT-aware reverse proxy commonly used with Mosquitto brokers. If the broker does not directly terminate client TLS and is deployed behind HAProxy, HAProxy performs TLS termination and load balancing on behalf of the broker. This component was out of scope. Certificate Authority (*) This is the authoritative party that holds the private key for signing Mosquitto broker and client certificates, as well as the public key for verifying them. This component was out of scope. SSL/TLS library (*) This is the library handling cryptographic operations in the Mosquitto broker and in the libmosquitto API. The library is either LibreSSL or OpenSSL . This component was out of scope. HashEye 11 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

openssl (*) This is the command-line utility (indicated by documentation) to generate certificates for Mosquitto broker and client operations. This component was out of scope. Client This is an MQTT client, either based on libmosquitto or third-party client software (the latter was out of scope), that publishes to or subscribes to topics coordinated by a Mosquitto broker instance. Custom broker plugin This is a third-party plugin built with Mosquitto-provided components for the purposes of configuring the Mosquitto broker. mosquitto_ctrl This is a command-line tool to simplify the reconfiguration of the MQTT broker at runtime. It optionally reads an options file, which stores the command-line configuration, from the local filesystem. mosquitto_ctrl_dynsec This is a command-line tool for configuring the Mosquitto broker's Dynamic Security plugin (module). Generally, this tool simplifies the publication to the Dynamic Security plugin's $CONTROL MQTT topics. mosquitto_passwd This is a command-line tool for managing Mosquitto broker password files in the local filesystem. mosquitto_pub This is a command-line tool that can publish simple messages to a given Mosquitto-brokered topic. mosquitto_sub This is a command-line tool that can subscribe to a Mosquitto-brokered topic and will print all messages it receives. mosquitto_rr This is a command-line request/response client that can both receive and publish MQTT messages. HashEye 12 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Trust Zones Systems include logical "trust boundaries" or "zones" in which components may have different criticality or sensitivity. Therefore, to further analyze a system, we decompose components into zones based on shared criticality, rather than physical placement in the system. Trust zones capture logical boundaries where controls should or could be enforced by the system and allow designers to implement interstitial controls and policies between zones of components as needed. Zone Description Included Components Private Network The network zone(s) under the control of Mosquitto broker administrators and client operators • libmosquitto clients • Third-party MQTT clients • Broker, when configured to terminate client TLS itself Public Network The network zone for data that crosses the broader internet (e.g., between a client and a broker) and, therefore, must cross third-party networks before reaching its destination • HAProxy, when situated in front of a broker • Broker, when configured to accept direct client connections • Certificate Authority • libmosquitto clients • Third-party MQTT clients • Remote Mosquitto test clients (e.g., mosquitto_rr) Localhost The network and system-level zone for data that may persist on the same host as a component (e.g, the host running a Mosquitto broker) • Broker • Broker plugins • Mosquitto ctrl and Dynamic Security plugin ctrl utilities • Local Mosquitto test clients (e.g., mosquitto_rr) Process The running broker process on the host • Broker • Broker plugins HashEye 13 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Trust Zone Connections At a design level, trust zones are delineated by the security controls that enforce the differing levels of trust within each zone. As such, it is necessary to ensure that data cannot move between trust zones without first satisfying the intended trust requirements of its destination. We enumerate such connections between trust zones below. Originating Zone Destination Zone Data Description Connection Type Authentication Type Private Network 1 Public Network Data sent from a client to HAProxy: • Username, password • MQTT control packets • Published messages Unencrypted TCP, TLS, WS, WSS Username/ password, client certificate, anonymous access Public Network Private Network 2 Data sent from HAProxy or a broker downstream to a client (e.g., subscribed messages) Unencrypted TCP, TLS, WS, WSS Username/ password, client certificate, PSK, anonymous access Private Network 2 Public Network Data sent between bridged brokers: • MQTT control packets • Bridged messages • Username, password Unencrypted TCP, TLS Username/ password, client

certificate, PSK, anonymous access Private Network Localhost Data sent from MQTT clients to the broker: • Username, password Unencrypted TCP, TLS, WS, WSS Username/ password, client certificate, anonymous access HashEye 14 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

• MQTT control packets • Published messages Localhost Broker Process Configuration data loaded from the local filesystem at broker runtime or sent to the running broker via Mosquitto ctrl utilities Also, local test clients' published messages sent to the local broker Unencrypted TCP, TLS, WS, WSS Username/ password, client certificate, anonymous access Localhost Private Network Published messages delivered from the broker to clients on the local network Unencrypted TCP, TLS, WS, WSS Username/ password, client certificate, anonymous access HashEye 15 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Threat Actors Similarly to establishing trust zones, defi ning malicious actors when conducting a threat model is useful in determining which protections, if any, are necessary to mitigate or remediate a vulnerability. We will use these actors in all subsequent findings from the threat model. Additionally, we defi ne other users of the system who may be impacted by, or induced to undertake, an attack. For example, in a confused deputy attack such as cross-site request forgery, a normal user would be both the victim and the potential direct attacker, even though that user would be induced to undertake the action by a secondary attacker. Actor Description External Attacker An attacker on the public network who can eavesdrop on and potentially modify (MitM) other users' connections that route through the public network Internal Attacker An attacker on a private network who can eavesdrop on and potentially modify other users' connections that route through that private network Local Attacker An attacker who controls a process or user account on the same host as the Mosquitto broker and can affect the environment or filesystem Client Developer Integrates libmosquitto in custom MQTT client applications Client Has full control of the client device connected to a broker Infrastructure Administrator Can read from or, as appropriate, publish to broker and broker plugin $CONTROL and $SYS topics; has full access to the server or container running the Mosquitto broker and ctrl utility software Proxy Operator Has full administrative access to a reverse proxy (e.g., HAProxy) that terminates client TLS for a Mosquitto broker Contributor A regular contributor to Mosquitto source code Maintainer A gatekeeper controlling additions to the source code Certificate Authority A signer and validator of broker and client certificates HashEye 16 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Threat Actor Paths Additionally, defi ning attackers' paths through the various zones is useful when analyzing potential controls, remediations, and mitigations that exist in the current architecture. Originating Zone Destination Zone Actor Description Public Network Public Network External Attacker An external attacker suitably positioned on the public network between a client and broker is able to read and tamper with unencrypted traffic. Private Network Private Network Internal Attacker An internal attacker suitably positioned on the private network of either a client or broker is able to read and tamper with unencrypted traffic. Public Network Public Network Certificate Authority A malicious or compromised Certificate Authority can sign fake certificates to enable MitM attacks on encrypted traffic by an external or internal attacker. Private Network Private Network Proxy Operator The operator of a HAProxy instance that terminates client TLS for a Mosquitto broker is able to inspect and modify all traffic between the client and broker. Localhost Localhost Local Attacker An attacker gains control of a user account on the broker host machine, or compromises another process running on the host, and is able to 1) make changes to the host environment that affect broker behavior and 2) access broker configuration data and logs. Localhost Localhost Local Attacker An attacker obtains superuser access on the broker host machine, or compromises another process running on the host as root, and is able to monitor and intercept all broker traffic. Localhost Localhost Local Attacker An attacker obtains access to the user account under which Mosquitto is running on the broker host machine. The attacker uses methods such as ptrace(2) to monitor network traffic sent and HashEye 17 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

received by the broker process. Private Network Private Network Internal Attacker A compromised/malicious bridge or client operator is able to consume excessive broker system resources and potentially negatively affect other clients and other bridged brokers via excessive published messages, authentication attempts, or connection attempts. HashEye 18 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Summary of Recommendations Throughout the engagement, HashEye identified a number of threat scenarios that pose risk to Mosquitto deployments and clients. HashEye recommends that the Mosquitto maintainers and contributors address the findings detailed in this report, especially prioritizing the steps below to further build upon threat modeling exercises: • Simplify the ACL system. Removing manual priority specification from the Dynamic Security ACL system entirely will make evaluation order more consistent. ○ For a combination of a username, an action, and a topic path, ACL rules should be evaluated in order from those that apply to the most specific/narrowest

applicable path (e.g., parent/foo/bar/stuff) to the least specific/broadest applicable path (e.g., parent/#). This will ensure that the most specific rules always apply first. ○ Only allow access control configuration file/runtime configuration changes to come from the ctrl utilities run as the mosquitto user (or root). ○ Deny all client/username access by default until the infrastructure admin intentionally allows a particular client access to a given topic or set of topics. • Improve the fuzzing coverage. Particularly with regard to MQTT packet parsing code, additional fuzzing coverage obtained via internal or external audits will help determine how, for example, MQTT packet parsers within Mosquitto handle their respective acceptable input ranges. ○ Log more extensive information on errors such as crashes, hangs, and unexpected exits or syscalls within the Mosquitto broker and Dynamic Security plugin. This will help broker administrators produce better issue reports in the event of a problem at runtime in their deployments. • Leverage static analysis. Run a targeted set of CodeQL or other static analysis rules built from known-bad patterns against pull requests to help prevent regressions and “low-hanging fruit” vulnerabilities from being introduced. ○ Instead of running a large SAST query set, which could lead to an overwhelming number of false-positive or less-useful alerts, scan each PR against develop and master branches with a small, tailored CodeQL query suite based on past Mosquitto security flaws, security issues in similar MQTT projects, and potentially general known-bad security patterns. HashEye 19 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

○ Determine an upper bound for false positives reported over a period of time (e.g., 24 hours or one week), after which a particular CodeQL query is primarily not reporting helpful information; modify or remove any query that produces too many false positives by this upper bound. ○ Exclude test-related code and folder paths from CodeQL to further reduce false-positive alerts. ○ Require each PR author to resolve PR-specific CodeQL findings (require status checks to pass) before allowing PRs to be merged into develop or master . • Implement comprehensive ACL exploration functionality. Enable infrastructure administrators to easily validate per-topic and per-client what combinations of access control decisions will apply (and in what order) at runtime. • Do not store passwords or other sensitive data in plaintext in configuration files. This will reduce the potential attack surface and information desirable to an attacker on the broker host. ○ Only store (or allow use of) hashed and uniquely salted broker passwords . ○ Enable administrators to provide sensitive data, such as a bridge connection remote_password , at runtime so it is only read into broker memory and not stored on the broker host filesystem. • Refactor the handwritten parsers. Reimplement each parser in an human-readable specification format such as ASN.1. Use a parser generator on these specifications to create the actual code to link into Mosquitto. This will enable formal verification; handwritten parser routines are more likely than formally verified, automatically generated parsers to contain subtle, unintended issues. ○ Additionally, consider more clearly separating the parsing code from the input validation and packet handling code to facilitate verification, testing, and fuzzing of critical paths. HashEye 20 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Summary of Findings The table below summarizes findings during the review, including type and severity details. ID Title Type Severity 1 Insufficient default configuration file permissions Access Controls High 2 Unclear ACL, role, group enforcement priority Access Controls Medium 3 Missing global connection rate limiting Denial of Service High 4 Plaintext password storage and handling System and Information Integrity High 5 Bridge→broker→bridge message looping Denial of Service High 6 Broker does not check configuration filesystem permissions Access Controls Medium 7 Configuration reload may cause inconsistent behavior System and Information Integrity Low 8 Clients can publish last will messages to $CONTROL topics Access Controls High HashEye 21 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Detailed Findings 1. Insucient default configuration file permissions Severity: High Difficulty: High Type: Access Controls Finding ID: TOB-MOSQ-1 Target: Mosquitto Broker Description Mosquitto broker settings can come from local user-managed files such as mosquitto.conf . On a system with Linux capabilities, any user with CAP_KILL capabilities for the broker parent process can force the broker to reload all configuration files by sending the SIGHUP signal to the broker process via kill() . Mosquitto utilities such as mosquitto_ctrl do not disallow or discourage manual edits to these configuration files, nor do these utilities check that filesystem permissions are sufficiently restrictive before updating the file in question. Threat Scenario An infrastructure administrator creates broker configuration files such as acl_file with broad write permissions. A malicious user on the machine where the broker runs lacks the capability to SIGHUP the broker process, but can edit configuration files to enable anonymous access in mosquitto.conf , change group membership in acl_file , and add an unexpected psk_file entry, among other actions. The next time the administrator makes a benign change to any one of the configuration files and SIGHUP s the broker, the broker accepts all changes (including the attacker’s) across all configuration files. Docker also lacks default container filesystem protections, which means running the broker in a container does not prevent or protect against this scenario. Any container-host user account with

the necessary filesystem access permissions can still directly edit broker configuration files located within a Docker container. Since the attacker must obtain a user account on the broker host machine, we consider this scenario to be of high difficulty. Since the changes a malicious user makes can broadly affect trust between the broker and all connected clients, we consider this issue to be of high severity. HashEye 22 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Recommendations Short term, modify the Mosquitto documentation to clearly recommend strict default configuration file permissions such as 640 or even 600 , and ensure all Mosquitto Dockerfiles set the default configuration folder and file permissions to disallow reads and writes from users who do not directly own them. Long term, make the ctrl utilities the primary way to interact with any configuration files for the Mosquitto broker. Ctrl utilities should automatically create all Mosquitto configuration files; if a given file does not already exist, create it empty or with a default deny-all ruleset with strict default access permissions (e.g., 600). Similarly to the recommendations provided for TOB-MOSQ-6 , ctrl utilities should refuse to load or modify broker configuration files that lack strict-enough filesystem access permissions, akin to ssh ’s restrictions. References ● Adam Shostack: Fail-Safe Defaults HashEye 23 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

2. Unclear ACL, role, group enforcement priority Severity: Medium Difficulty: Low Type: Access Controls Finding ID: TOB-MOSQ-2 Target: Dynamic Security plugin Description It is unclear from the documentation and examples within the documentation which ACLs will be checked first, or the priority order in which rules will apply, if multiple (user-wise, group-wise, or role-wise) rules apply to a particular client (user) publishing to or subscribing to a given topic. Threat Scenario For a broker with many clients and multiple roles/groups, the infrastructure administrator creates overlapping access rules, including a general allow-all-actions rule for all usernames for topic/# and a narrowly scoped rule intended to make subscribing to, publishing to, and receiving messages from topic/secret available only to members of the secret group. The admin assumes the topic/secret group-based deny rule will take precedence over the allow-everything rule when a client attempts topic/secret access. Since ‘ # ’ has a lower ordinal value than other characters that are possible in topic names, and neither rule has a priority value assigned, the administrator assumes the deny rule will apply before the publish-allow rule for all topics under topic/ . However, since client-role-related rules are always checked before group-related rules in acl_check() and the method returns once a matching rule is found for the action type and client, an attacker-controlled client not in the secret group can subscribe to topic/secret and receive messages published to it, contrary to the administrator’s assumptions. We consider this scenario to be of low difficulty, since the malicious client can by default receive messages on topic/secret . We consider this scenario to be of medium severity since potentially sensitive user information published to the topic is compromised. Recommendations Short term, clarify the ACL documentation and examples regarding the interaction of the administrator-configurable priority parameter with user-associated, role-associated, and group-associated rules. Document the implicit precedence ordering of user and group-associated rules due to acl_check() structure. HashEye 24 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Long term, to reduce complexity, take the following actions: ● Remove the configurable priority rule parameter entirely. ● Refactor the ACL checking code so that user and group-specific rules are equally likely to apply to a username/client, action, and topic combination: ○ Determine and then check the full set of ACL rules that apply for a user and the groups to which they belong for the given topic. ○ Order rule checking from the narrowest (longest) topic specifier to the broadest (mainly, shortest) topic specifier: ■ Since rules applying to wildcarded (# , +) topic specifiers are broader than other rules applying to a comparable number of topic path segments, rules containing wildcards should be checked as the last of each set at a given level of the topic tree. ■ For example, to determine whether a client may subscribe to topic/foo/secret , first determine the branch(es) of the tree of topics within which the topic in question falls (e.g., topic/ , topic/foo/). Then, check only within the applicable topic-tree branch for rules relating to the narrowest/most closely related path (e.g., topic/foo/secret) first, returning if an applicable more-specific rule is found before any broader rules may apply. ■ Thus, rules applying only to topic/foo/secret are checked before rules that cover, for example, topic/foo/# and topic/# . ○ If the Dynamic Security plugin checks ACL rules following this topic-tree ordering, it should also be possible to remove the current lexicographic fallback rule ordering. Ordering rule checks by topic should be less error-prone and should prove more intuitive to broker and client administrators. HashEye 25 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

3. Missing global connection rate limiting Severity: High Difficulty: Low Type: Denial of Service Finding ID: TOB-MOSQ-3 Target: Broker Description A number of per-listener maximum values are configurable in Mosquitto, including the maximum number of client connections, the global maximum client packet size, and the global publishable payload size. However, there are no rate-limiting

mechanisms for broker-wide (global) connection and authentication attempts, which, if implemented, would prevent broker resource overuse by a particular client or bridge. These mechanisms cannot be clearly set with just one or a few configuration settings. Section 5.4.8 of the MQTT v5 specification (“Detecting abnormal behaviors”) encourages monitoring client behavior to detect repeated connection or authentication attempts and recommends adding misbehaving clients to a dynamic blocklist or rate-limiting list. Threat Scenario An attacker discovers a vulnerable Mosquitto broker through Shodan and obtains a valid client username for this broker. Through many failed attempts across multiple listeners (each listener could have different configuration settings, but all listeners rely on the same acl_file if it is present, and the same password_file), the user applies MQTT-pwn to obtain the correct password through brute force, as in issue #2076 . If acl_file does not exist, or existing ACL rules do not prevent such access from the credentials the attacker obtained, the now-authenticated attacker can publish to (and consume messages from) $CONTROL topics and other sensitive topics identified using MQTT-pwn. Alternatively, an improperly configured client simply overwhelms the broker by making many connection and/or authentication or authorization attempts across listeners. We consider this issue to be of low difficulty, since the attacker requires no specialized knowledge to exploit the issue. The severity is rated as high because this issue could affect all broker clients. HashEye 26 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Recommendations Short term, if such a combination of options exists currently, clearly document how to rate-limit connection attempts, authentication attempts, authorization attempts, and publish attempts from a particular client, across all listeners configured on a broker. Long term, implement and document a simple global (broker-wide) rate-limiting set of configuration options for connections, authentication, and publication. In particular, implement and document an option to specifically rate limit authentication attempts globally by user identity. Thus configured, the Mosquitto broker should uniquely identify and rate limit any individual client across listeners and individual connections. Additionally, either remove per-listener rate-limiting options or clearly document how per-listener rate limiting will interact with global rate limits. References ● CWE-307: Improper Restriction of Excessive Authentication Attempts HashEye 27 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

4. Plaintext password storage and handling Severity: High Difficulty: High Type: System and Information Integrity Finding ID: TOB-MOSQ-4 Target: Multiple Description The Mosquitto ctrl utilities and broker permit plaintext password storage and usage. For example, they allow a user to provide a plaintext password, such as the broker administrator password, as the value of a command-line utility configuration option, and they allow the storage of plaintext passwords in configuration files such as mosquitto.conf , password_file , and the mosquitto_ctrl options file. Threat Scenario An attacker obtains read access to the filesystem of the machine or container hosting the Mosquitto broker and copies the broker configuration files. From the mosquitto_ctrl saved options file, the attacker obtains a broker administrator user password and gains broker administrator access. Alternatively, an attacker gains read access on a client device from which the administrator has previously remotely configured the broker. The attacker recovers the broker administrator password from the device’s shell history file and gains broker administrator access. Alternatively, an attacker obtains the remote_password and remote_username that this broker uses to bridge “out” to another broker without TLS PSK or a bound local IP from mosquitto.conf , and impersonates this broker to the remote broker. Since these scenarios require filesystem read access, we consider this issue to be of high difficulty. The outcome is administrator-level broker takeover, which affects all clients and bridged brokers, so the severity is also high. Recommendations Short term, disallow all password storage in configuration files other than password_file . Disallow user creation through passing a plaintext password value directly on the command line to mosquitto_passwd . Prefer reading passwords into app memory through a secondary dialogue, where feasible without breaking existing functionality, instead of allowing password(s) directly as the value of a given command-line argument. HashEye 28 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Long term, also disallow plaintext passwords entirely in password_file , even when TLS support is not compiled into Mosquitto, so that the default choice is the most secure choice (with regard to password storage) that the broadest number of users will accept. Do not store or support existing unsalted and unhashed passwords. Following OWASP and NIST recommendations, either choose a more secure key stretching algorithm (i.e., argon2) or configure PBKDF2 by default with an appropriately large number of hash iterations, salt size, and so on. Write test cases to ensure the Mosquitto utilities and broker reject authenticated-user topic accesses from users with passwords that the broker has stored as plaintext in password_file until the administrator updates the password file. References ● OWASP Password Storage: PBKDF2 ● NIST SP 800-132 : Recommendation for Password-Based Key Derivation Part 1: Storage Applications ● NIST SP 800-63 : Digital Identity Guidelines ● CWE-

256: Plaintext Storage of a Password • CWE-260: Password in Configuration File HashEye 29 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

5. Bridge → broker → bridge message looping Severity: High Difficulty: Low Type: Denial of Service Finding ID: TOB-MOSQ-5 Target: Broker Description It is possible to bidirectionally bridge two Mosquitto brokers A and B and infinitely loop messages between them. It is also possible to rearrange a broker's topic tree when bridging A to B such that each broker's topics appear to contain no loops, but A will re-receive messages it published to B, and/or vice versa. Infinite loops could over-consume broker system resources. Threat Scenario An attacker bridges a broker they control, A, to a target broker with existing clients, B, and also subscribes to the topics brokered by B on which messages bridged from A to B propagate, intentionally creating a loop between A and B. Since there is no rate limiting for messages propagated across a bridge to clients that subscribe to bridge topics, any message "bridged" to B from A could be looped back through A to the target B as many times as A's resource allocation can support, consuming additional system resources of B with each message publish back to A and replay to B over the bridge. If the attacker provisions A with a greater resource allocation than B, A could continue amplifying the amount of messages looped through B up to or past B's resource limits, denying service to clients and other bridged connections of B, and potentially also knocking B offline. Topics imported from A and brokered by B to clients can no longer be consumed without causing client network-connection saturation. All existing clients of these topics are either pinned to their maximum resources or knocked offline. Since this scenario requires only the ability to bridge another broker A to the target broker B, we consider this attack to be of low difficulty. Since all clients and the broker B itself could be affected, we consider the severity to be high. HashEye 30 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Recommendations Short term, create topic-tree validation tooling that administrators of B can run to find and eliminate publish/subscribe loops that could degrade service for clients or bridges. Introduce global rate-limiting functionality across bridges and clients, as is also recommended for TOB-MOSQ-3 , to prevent any bridged-in broker A from overwhelming the broker in question, B, through intentionally looping messages. Long term, refactor the bridging code to wholly prevent any bridge A from directly (or indirectly, through topic/topic tree remapping) both producing and consuming the same remote topic on B. HashEye 31 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

6. Broker does not check configuration filesystem permissions Severity: Medium Difficulty: High Type: Access Controls Finding ID: TOB-MOSQ-6 Target: Broker Description Mosquitto can be provided with a password file containing username-password or username-hash pairs. However, when the broker loads the password file, it does not verify that the file's access permissions are sufficiently strict. Threat Scenario A naive user uses chmod to make an existing broker password_file writable to additional host system users (i.e., by making the file world- or group-writable). An attacker then obtains write access as a low-privileged user, writes a new entry to the file, and waits for the Mosquitto broker to eventually restart, which loads the attacker's new broker credentials. Since this scenario requires local write access, and the attacker's system user may not have the capability to force the broker to reload configuration files with SIGHUP , we consider this finding to be of high difficulty. We consider the severity to be medium because this example's broker relies only on the password file for client access control, resulting in the attacker gaining broad access to all brokered topics. Recommendations Short term, after addressing TOB-MOSQ-1 , when the broker loads a password file (both at startup and during SIGHUP -triggered reloads), check that the password file's permission flags are set to 0600 (writable/readable by user only). Display a warning message about password file permissions if they are not sufficient; consider ssh 's "permissions are too open" error message, reproduced below. @@@ @ WARNING: UNPROTECTED PRIVATE KEY FILE! @ @@@ Permissions 0644 for '/Users/example/.ssh/id_rsa.pub' are too open. It is required that your private key files are NOT accessible by others. This private key will be ignored. bad permissions: ignore key: /Users/example/.ssh/id_rsa.pub Permission denied (publickey,password). HashEye 32 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Long term, take the following actions: • If the security of the application depends on the fact that local configuration files are writable only by certain users (i.e., root and mosquitto), it is important not only to check that access permissions are sufficiently strict when loading such a file, but also that the broker fails to start if this invariant does not hold. • If filesystem permissions for a configuration file such as password_file are insufficient, do not allow the broker to start. It is important to fail to start the broker instead of simply failing to load the password file, to avoid unintentionally running with no usernames and passwords if password_file permissions are too open. This method is not foolproof, as there is also a possibility of a time-of-check to time-of-use (TOCTOU) race between the broker checking the permissions of a file and

another process completing a permissions or file contents modification; however, this method does help rule out the most common case of naive over-permissioning. References • OWASP: Filesystem Permissions • CWE-367: TOCTOU • Man: ssh HashEye 33 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

7. Configuration reload may cause inconsistent behavior Severity: Low Difficulty: High Type: System and Information Integrity Finding ID: TOB-MOSQ-7 Target: Broker Description The Mosquitto broker reloads its configuration upon receiving a SIGHUP signal. If settings that affect how requests are handled are changed, requests that are received during the configuration reload could be processed in unexpected or inconsistent ways. Note that plugins can be loaded only at startup and cannot be unloaded via a SIGHUP , so authentication race conditions are not a concern for this issue. Threat Scenario An administrator configures a broker with allow_zero_length_clientid=true . Later, they change that setting to false in the Mosquitto configuration file. Rather than risk downtime by restarting the broker entirely, they send a SIGHUP to the broker so that it reloads its configuration. A client that was initially allowed to connect with a zero-length client ID could still remain connected after this change, leading to unexpected behavior. We consider this issue to be of low severity since authentication plugins cannot be reloaded via the SIGHUP method. We consider the difficulty to be high since privileged local access is required to restart the broker. Recommendations Short term, pause the processing of incoming or queued requests during configuration reloads and resume only once all authentication methods are fully initialized. In addition, consider providing a configuration option to drop queued requests instead to accommodate client use cases in which a drop-and-retry approach would be preferable to blocking. Long term, to reduce similar ambiguities, ensure that all (sensitive) configuration-dependent functionality cannot execute while the configuration is in a transitional state, such as during a SIGHUP -induced reload. HashEye 34 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

8. Clients can publish last will messages to $CONTROL topics Severity: High Difficulty: Low Type: Access Controls Finding ID: TOB-MOSQ-8 Target: Broker Description On initial connection, an MQTT client can set a “last will and testament” (LWT) containing an arbitrary message that is limited only in length. The broker publishes this LWT to the chosen MQTT topic in the event the client unexpectedly loses its connection to the broker. While a client’s proposed will topic cannot be greater than a certain length and cannot contain invalid UTF-8 characters, the Mosquitto broker does not prevent a client from publishing its LWT to $CONTROL topics, which access control plugins such as Dynamic Security use as a configuration API. This could result in a scenario in which an attacker-controlled client is able to set in advance an LWT that will alter security-related plugin configurations when it is published, after the client has lost ACL permissions to make such changes directly. Threat Scenario A client connects to the Mosquitto broker and sets as its LWT a command to alter the Dynamic Security configuration, such as by adding a backdoor user account with administrative privileges. The client sets the $CONTROL topic for this Dynamic Security action as the LWT destination topic. While the client is connected, an administrator revokes its privileges. The client then disconnects without sending a DISCONNECT packet to the broker , causing the broker to publish the previously-set LWT to the previously-selected $CONTROL topic, creating a backdoor account through the Dynamic Security plugin API. The attacker maintains privileged broker access via this new account. We consider this issue to be of high severity since a malicious client’s ability to change Dynamic Security plugin configuration on abnormal disconnect could affect all other broker clients. We consider the difficulty to be low since no specialized knowledge or significant eff ort is required to set such an LWT. Recommendations Short term, as is currently done with $SYS topics, prevent all clients from setting any $CONTROL topic as the destination topic for an LWT. HashEye 35 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Long term, consider implementing a dedicated last will topic that consists primarily of LWTs. Although the MQTTv5 specification requires handling LWTs, it makes no restrictions on which topic(s) should receive and propagate such messages. HashEye 36 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

A. Methodology HashEye’ threat modeling assessments are intended to provide a detailed analysis of the risks facing an application at a structural and operational level, assessing the security of its design as opposed to its implementation details. During these assessments, engineers rely heavily on frequent meetings with the client’s developers, paired with extensive readings of any and all documentation the client can make available. Code review and dynamic testing are not an integral part of threat modeling assessments, although engineers may occasionally consult the codebase or a live instance to verify specific assumptions about the system’s design. Engineers begin a threat modeling assessment by identifying the system’s security controls , the safeguards and guarantees that are critical to maintaining the target system’s confidentiality, integrity, and availability. These security controls dictate the assessment’s overarching scope and are determined based on the specific requirements of the target system, which may include technical and reputational concerns, legal liability, regulatory compliance, and so on. With these security

controls in mind, engineers then divide the system into logical components —discrete elements that perform specific tasks—and establish trust zones around groups of components that lie within a common trust boundary. They identify the types of data handled by the system, enumerating the points at which data is sent, received, or stored by each component, as well as within and across trust boundaries. Having established a detailed map of the target system's structure and data flows, engineers then identify threat actors —anyone who might threaten the target's security, whether a malicious external attacker, a naive insider, or otherwise. Based on each threat actor's initial privileges and knowledge, threat actor paths are then traced out through the system, establishing which controls and data a threat actor might be able to improperly access, as well as which safeguards stand in the way of such compromise. Any viable attack path discovered in this way constitutes a finding , which is paired with design recommendations by which such gaps in the system's defenses can be remediated. After enumerating a list of findings, engineers rate the strength of each security control, indicating the general robustness of that type of defense against the full spectrum of possible attacks. HashEye 37 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

B. Security Controls and Rating Criteria The following tables describe the security controls and rating criteria used in this report. Security Controls Category Description Access Controls Authorization, session management, separation of duties, etc. Audit and Accountability Logging, non-repudiation, monitoring, analysis, reporting, etc. Awareness and Training Policy, procedures, and related capabilities Security Assessment and Authorization Assessments, penetration testing, authorization to deploy, etc. Configuration Management Inventory, secure baselines, configuration management, & change control Contingency Planning Disaster recovery, continuity, backups, testing, and related controls Cryptography The cryptographic controls implemented at rest, in transit, and in process Denial of Service The controls to defend against different types of denial-of-service attacks impacting availability Identification and Authentication User and system identification and authentication controls Incident Response Policy, process, handling, reporting, and related controls Maintenance Preventative and predictive maintenance, and related controls Media Protection Identification, storage, sanitization, and removal Personnel Security HR Processes, screening, and related controls Physical and Environmental Protection Controls to protect work sites and related assets Planning Security architecture, policy, procedures, management, etc. HashEye 38 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Program Management Assigned responsibility and commitment to plans for critical infrastructure, enterprise architecture, information security programs, plan of action and milestones, and risk management strategies. Risk Assessment Risk assessment policies, vulnerability scanning capabilities, and risk management solutions. System and Communications Protection Network level controls to protect data System and Information Integrity Software integrity, malicious code protection, monitoring, information handling, and related controls System and Services Acquisition Development lifecycle, documentation, supply chain, etc. Rating Criteria Rating Description Strong The security control was reviewed and no concerns were found. Satisfactory The security control had only minor issues; though it may lack certain non-critical operational procedures or security measures, their absence does not expose users to a significant degree of risk. Remediation in this area is suggested, but is not urgent. Moderate The security control had several issues or an impactful issue which may expose users to some degree of risk, albeit not to a severe degree. Remediation in this area is desired. Weak The security control had several significant issues which are likely to expose users to a substantial amount of risk. Remediation in this area should be prioritized. Missing The security control was found to be nonexistent or totally ineff ective for its intended purpose, despite being necessary for the system's security. The implementation of this control should be prioritized. Not Applicable The security control is not applicable to this review. Not Considered The security control was not considered in this review. Further Further investigation is required to reach a meaningful conclusion. HashEye 39 OSTIF Eclipse: Mosquitto Threat Model PUBLIC

Investigation Required Severity Levels Severity Description Informational The issue does not pose an immediate risk but is relevant to security best practices. Undetermined The extent of the risk was not determined during this engagement. Low The risk is small or is not one the client has indicated is important. Medium User information is at risk; exploitation could pose reputational, legal, or moderate financial risks. High The flaw could affect numerous users and have serious reputational, legal, or financial implications. Diffi culty Levels Difficulty Description Undetermined The difficulty of exploitation was not determined during this engagement. Low The threat is well known or common; an attacker can exploit it without significant eff ort or specialized knowledge. Medium An attacker must acquire in-depth knowledge of the system or expend a non-trivial amount of eff ort in order to exploit this issue. High An attacker must acquire complex

insider knowledge or privileged access to the system in order to exploit this issue. HashEye 40
OSTIF Eclipse: Mosquitto Threat Model PUBLIC