SHERLOCK

PUBLIC SUMMARY REPORT

Forensic Investigation of Golem Provider Setups

Assessment of a sandbox file-access vulnerability reported via the Golem Bug Bounty Programme.

PREPARED FOR	PROVIDER SETUPS INVESTIGATED	VERSION / DATE
Golem Factory	33	1.0, 4 September 2026

PREPARED FOR	Golem Factory
PREPARED BY	Defsec, Independent Security Assessment
SUBJECT	Provider-side exploitation evidence for a sandbox file-access vulnerability
PROVIDER SETUPS INVESTIGATED	33
INVESTIGATION PERIOD	July to August 2026
REPORTING STANDARD	NIST SP 800-86, augmented with the SANS DFIR six-phase model
VERSION / DATE	1.0, 4 September 2026
CLASSIFICATION	Public

1 Conclusion

> **No evidence of remote code execution, unauthorised remote access, or exploitation of the identified vulnerability was found across the collected samples.**

Across the 33 investigated provider setups, the analysis established the following, in decreasing order of evidentiary strength:

1. **The vulnerability was not exercised.** The complete vocabulary of commands ever issued to the investigated setups by remote requestors was reconstructed from surviving execution logs. It contains four command shapes. None is the command type through which the reported vulnerability is reached.
2. **The vulnerability was, in the observed configuration, structurally unreachable,** not merely unexploited. It requires a provider-side resource that was never instantiated on any investigated setup. This is a stronger statement than an absence-of-evidence finding, and is developed in §5.2.
3. **No indicator of exploitation, escape, or post-exploitation activity was recovered** by any independent evidence class examined: software integrity, authentication records, persistence mechanisms, outbound network activity, or deleted-object analysis.
4. **The provider software was unmodified.** 150 binary instances of six distinct provider components, from independent installations performed at different times, were byte-identical to one another and structurally consistent with the official build pipeline.

The investigation additionally identified provider-side configuration weaknesses which did not contribute to any observed compromise, but which would have materially increased the impact of the reported vulnerability had it been reachable (§7). Two residual items could not be resolved from the available evidence and are stated openly in §8 rather than resolved by inference.

2 Background and Scope

2.1 The reported vulnerability

A report submitted through the Golem Bug Bounty programme identified a previously unknown weakness in provider-side handling of requestor-supplied file paths. Under specific conditions, a crafted path could resolve outside the intended sandbox boundary, permitting a remote requestor to read files belonging to the provider's own environment rather than to the task workload. Depending on what such a primitive reached, and on whether it could be extended from read to write, escalation to remote code execution on the provider machine was considered plausible.

This report describes the vulnerability at class level only. It does not reproduce the exploitation path, the affected code, or the conditions required to trigger it. Remediation is handled separately, through the bug bounty process.

For forensic purposes the essential property is this: **the vulnerability is reached through a specific class of requestor-issued command, and requires a specific provider-side resource to be present in order to have anything to resolve against.** Both are directly observable in provider evidence, and both were tested.

2.2 The investigation question

The Golem Factory commissioned an examination of provider setups for which forensic data could be collected, to answer a single question:

> *Is there evidence that this vulnerability, or any equivalent path to code execution or unauthorised access, was exploited against provider machines before it was reported?*

The investigation was framed as a **falsification exercise**. Rather than searching only for signatures of a known exploit, which leaves silent variants undetected, the analysis enumerated every mechanism by which a provider machine could plausibly have been compromised and attempted to eliminate each against independent evidence. A hypothesis was considered addressed only when the evidence contradicted it, or when the attack surface it required was shown never to have existed.

2.3 Out of scope

The following are outside the scope of this report and are not analysed or reported here:

- **Marketplace and settlement behaviour**, including cases where invoices were not paid. Non-payment may result from defects in requestor software, or from a requestor determining that a provider did not deliver as expected, for example through overprovisioning or failure to meet declared performance thresholds. Under the current protocol design several such behaviours cannot be verified in a fully trustless manner and depend on trust assumptions. They are not, in themselves, protocol-level security vulnerabilities, and are addressable through blacklists, whitelists, and local or global reputation mechanisms. The protocol is expected to be redesigned in 2027, with parts of the future architecture based on the Arkiv Network.
- **Activity attributable to Vanity Market**, a partner-operated service that periodically benchmarks the network, maintains provider reputation rankings, and runs mining-like computations on the network.
- **Requestor conduct, provider performance, pricing, and reputation mechanisms** generally.
- **Any information capable of identifying an individual provider or its environment.** Operator identities, wallet and node addresses, node identifiers, network addresses, operating-system distributions and versions, hostnames, and detailed system configurations have been removed or generalised throughout. Where a finding applies to a subset of the setups, this is stated numerically without further breakdown.

A companion source-code review of the provider software was produced as a separate deliverable. Its findings are not enumerated here; none describes a path that the evidence shows to have been exercised against the investigated setups.

3 Methodology

3.1 Phases and governing standards

The investigation followed the four-phase model of NIST SP 800-86 [1], augmented with the Identification and Preservation stages of the SANS DFIR curriculum [3], six phases in total: **Identification** (scope, threat model, evidence classes to acquire), **Preservation** (image acquisition, hash sealing, chain of custody), **Collection** (structured extraction using non-mounting tools), **Examination** (enumeration of persistence, authentication, scheduling, service, package, journal, temporary-directory and home-directory artefacts), **Analysis** (cross-setup correlation, root-cause reconstruction, hypothesis ranking) and **Reporting**. Examination practice followed Linux-oriented incident-response methodology [4].

Each setup moved through the pipeline independently; cross-setup correlation occurred only in the Analysis phase, so that a conclusion reached on one setup could not condition the examination of another.

3.2 Evidence handling

- All disk artefacts were treated as read-only and **no filesystem was mounted at any stage**. Extraction was performed at inode level using tooling that reads bytes directly from the raw image

without traversing or modifying the source.

- Provider databases were **copied out before being opened**. This matters: opening a write-ahead-logging database read-write triggers recovery and checkpointing, which rewrites the source. Where a database was read in place it was opened through a read-only URI. No source database was modified.
- Timestamps cited in the analysis are taken from *within-artefact* content (log lines, database rows, filesystem journal entries), not from the modification times of extracted files, which the extraction workflow does not preserve.
- Cryptographic hashes of all primary artefacts were recorded at acquisition and retained. Analysis was performed on a dedicated offline examination workstation.

All tooling was open-source and the examination is reproducible on any Unix-like forensic workstation: inode-level forensic extraction tooling [5], a read-only database client, scripted batch parsing across the 33 datasets, standard binary-analysis utilities, and standard search utilities including decompression of every rotated log archive prior to searching.

3.3 An investigative principle: reachability before signature

A recurring weakness in exploitation assessments is that they search for the signature of an attack and, finding none, report absence of evidence. Absence of evidence is a weak conclusion: it is equally consistent with a competent attacker who left none.

Where possible this investigation therefore sought to establish **reachability** instead. If the resource an attack requires never existed in the observed configuration, the attack could not have occurred regardless of the attacker's sophistication, and the conclusion no longer depends on the completeness of the log record. §5.2 sets out where this stronger form of argument was available; §8 states plainly where it was not.

4 Evidence Base

4.1 Scale

Forensic data was collected across **33 provider setups**. In addition, one clean single-session reference installation of the Golem software, not part of any operator's production environment and not itself a provider deployment, was examined as a control: it establishes what a stock, unmodified installation looks like at rest, and supplies an independently-sourced set of daemon binaries for the integrity comparison in §5.4.

4.2 Artefact classes collected and analysed

The following artefact classes were collected from and analysed across the investigated provider setups. Not all classes were available from every setup; where a class was absent this is accounted for in §4.3 and §8.

System and configuration state: complete system configuration trees, including account, group, credential, scheduling, privilege-escalation, service-manager and remote-access configuration; host firewall and packet-filter configuration; installed-package logs and package-manager transaction history.

Authentication and access records: authorised-key material for all accounts permitting remote login; authentication logs across the full retention window available on each setup (up to four weeks), including all rotated and compressed archives; login, failed-login and last-login accounting databases; known-hosts records and interactive shell histories for all accounts.

System and service logs: full system log trees, including kernel, boot, service-manager and package logs; binary service-manager journals, in aggregate exceeding 8 GB; container-runtime standard-output logs, in aggregate exceeding 320 MB; provider daemon and provider runtime logs, current and rotated, over 2,000 individual runtime log files.

Provider application state: complete provider data directories, including agent state, configuration, hardware declarations, pricing presets, access-control rule files and certificate stores; provider databases covering identity, activity, market, payment and settlement state; surviving per-agreement working directories, each retaining a complete execution log, **67 such directories survived the provider software's own cleanup routine and were recovered in full**; cached task-package images, examined as filesystem images in their own right.

Binaries and filesystem images: provider filesystem images for 26 of the setups, examined at raw-image level, with a clean unmount state recorded for every image on which that check was run; all provider-stack binaries recovered from those images, being six distinct components and 150 binary instances in total; deleted-object (orphaned inode) enumeration across all recovered images.

4.3 Coverage limitations

For a subset of the setups the acquisition captured the variable, configuration, home and temporary directory trees but not the system binary directories. The consequences are stated rather than minimised:

- A small number of operator-installed helper utilities residing outside the captured tree **could not be hashed**. A binary-replacement rootkit located exclusively in those directories cannot be affirmatively excluded for those setups.
- Container image layers had been pruned before capture on some setups. This is largely compensated: each provider's actual working state resided on a separate filesystem image, and all such images were present and complete.
- No live process, socket, or loaded-module state was available, as acquisition was performed against powered-down systems.

Everything asserted in §5 about the provider software stack and the requestor-facing attack surface is fully evidenced. Assertions about the underlying host operating system are evidenced to the limit of the captured tree, and flagged accordingly.

5 Analysis and Results

5.1 Reconstruction of the complete remote command vocabulary

The reported vulnerability is reached through a specific class of requestor-issued command. Whether that command class was ever issued is therefore directly decidable, provided the record of issued commands is complete for some portion of the evidence.

67 per-agreement working directories survived the provider software's cleanup routine, each retaining the complete execution log for its agreement. Aggregating **every** command-execution line across all of them yields the entire vocabulary of commands that remote requestors ever issued to those setups: an environment-deployment command, a start command, a single-purpose informational command, and a single computational workload command.

That is the complete set. **There is no fifth command shape anywhere in the corpus, and in particular no instance of the transfer-class command through which the reported vulnerability is reached.**

This is a positive enumeration, not a keyword search. It does not depend on knowing what an exploit would look like: had a requestor issued any command outside these four shapes, it would have appeared in the aggregate as an unaccounted-for shape.

5.2 Structural unreachability of the vulnerability class

The reported vulnerability resolves a crafted path against a path namespace that the provider maps into the workload environment. If no such namespace is mapped, there is nothing for a crafted path to escape from, and the vulnerability has no attack surface irrespective of what the requestor sends.

Four independent observations establish that this was the case:

1. **No mapped volume existed on any deployment.** Every recovered deployment record declares an empty volume set, corroborated by the corresponding transfer-subsystem log line on each deployment. The path namespace the traversal requires was never instantiated.
2. **Workload virtual machines received no host filesystem share.** The launch arguments of the workload virtual machines contain no host-directory passthrough of any kind. A passthrough does appear in the provider runtime's own start-up self-test, which executes against a fixed local self-test image and never against a requestor workload. This is an important distinction, because a search that did not separate the two would produce a false positive here.
3. **The only transfer activity in the entire corpus was provider-initiated and integrity-checked.** Every transfer observed is the provider's own retrieval of the task package from the official registry, verified against the digest declared in the requestor's demand. Every such verification succeeded. No requestor-initiated transfer was recorded anywhere.
4. **No network path out of the workload existed.** The provider outbound access-control rule file was byte-identical across all recovered filesystem images and identical to the shipped default; it was never modified on any setup. Requestor manifests declared virtual-network capability, but every deployment record carried an empty network set, no network-creation or node-addition events

appear in any provider log, and both workload network interfaces were bound to loopback addresses.

Taken together these establish that the vulnerability class was **unreachable in the observed configuration**, and correspondingly that the related class of transfer-layer server-side request forgery was ruled out rather than merely unobserved.

5.3 Content sweep for exploitation and post-exploitation indicators

Independently of the structural argument, a content sweep was run across all extracted artefacts for the indicator classes associated with path traversal, local file inclusion, request forgery and credential theft: encoded and unencoded traversal sequences, local-file URI schemes, cloud instance-metadata addresses, loopback references to the provider's own management interface, and references to credential, key and process-information paths.

The sweep covered every extracted artefact on every setup for which the relevant trees were captured, **including every rotated and compressed log archive, decompressed before searching**.

It returned exactly one class of hit: the provider agent's own loopback calls to its local provider daemon, that is, normal operation. Two compressed archives produced apparent matches on compressed bytes; both yielded zero matches once decompressed, an artefact of compression entropy rather than content. No other match of any kind was recovered.

5.4 Software integrity

A compromised or backdoored provider binary would explain observed anomalies without requiring any exploitation of the reported vulnerability, and was therefore treated as a first-class hypothesis.

- **Independent replication.** The setups were provisioned at different times through independent installation runs, which produced **byte-identical binaries** for all six provider-stack components (150 binary instances, **zero variance**), matching the binaries recovered from the separate reference installation described in §4.1. Targeted per-setup or per-installation tampering is excluded by this result alone.
- **Build provenance.** Every embedded source-path string carries the path prefix characteristic of the official continuous-integration environment used to produce Golem's released builds. A binary rebuilt elsewhere, or patched after the fact, would not carry it.
- **Structural comparison.** The deployed release was compared against the officially published successor release across ELF class, section layout and count, compiler and toolchain identifiers, dependency commit identifiers and build-path prefix. All matched; the only divergences were those expected between two builds of adjacent versions.
- **Binary composition.** No anomalous or non-standard sections were present. The binaries are statically linked position-independent executables with no dynamic-linker dependency, and so cannot be subverted through library preloading. A URL and address sweep resolved every external reference to official project, block-explorer, public RPC or attestation endpoints; no routable public address and no embedded payload was found.

- **Task-package images.** Both cached workload images were examined as filesystem images. A strings sweep returned ordinary distribution packaging strings only: no command-and-control endpoint, no exfiltration target, no non-standard address. Both execute exclusively inside a hardware-virtualised guest, never on the host.

Verdict: no evidence of supply-chain compromise or binary tampering. High confidence. The single residual gap, a direct hash comparison against an official archive of the exact deployed release, no longer retrievable from the distribution point, is recorded in §8 and is of marginal weight against the replication result above.

5.5 Unauthorised access, persistence and outbound activity

Remote access. Across the setups retaining authentication logs, the retained window covers up to four weeks per setup and in excess of **1.4 million accepted sessions** in aggregate. Every one originated from a single management source on the operator's own local network, authenticating with a single key. Across the entire aggregate: **zero failed authentication attempts**, and **zero sessions originating from outside the local network**. Remote-login configuration, authorised-key material, privilege-escalation policy and account state were enumerated on every setup for which the relevant trees were captured; one anomaly was identified and is reported openly in §8, and no other deviation was found. The provider management interface was bound to loopback on every instance, confirmed from the start-up log line of each individual provider process, not from configuration alone, and was therefore not remotely reachable.

Persistence. Scheduling tables, service-manager unit definitions, dynamic-linker preload configuration, authentication-module configuration and package transaction logs were enumerated across the setups. All non-default service units were traceable to a corresponding command in the operator's own interactive shell history, in chronological order, and are consistent with the operator's declared use of the machines for unrelated workloads. Preload configuration was absent; authentication-module and scheduling configuration were stock; package logs showed distribution updates only, with no manual package installation. **No persistence mechanism attributable to a third party was identified on any setup.**

Outbound activity. The only outbound activity attributable to the provider stack is task-package retrieval from the official registry. Operator-run workloads unrelated to Golem reach their own public endpoints and nothing else. No beacon pattern, unexplained endpoint, or command-and-control indicator was recovered from logs, journals, binaries or task-package images.

5.6 Anti-forensic indicators

Deleted-object enumeration across all recovered provider filesystem images identified 67 orphaned inodes in total, all unnamed and zero-length. **Zero** deleted objects matching shell-history, key material, credential, database or configuration patterns were recovered from any image.

There is no evidence of log truncation, log manipulation, timestamp tampering, or evidence destruction anywhere in the collected samples. This matters for the interpretation of §5.3 and §5.5: the negative results in those sections rest on a log record that shows no sign of having been curated.

6 Results Summary and Confidence

HIGH: multiple independent artefact classes agree, no contradicting evidence. **MEDIUM:** one or two classes agree, alternatives not fully excluded. **LOW:** single-source evidence, alternatives remain plausible.

#	HYPOTHESIS TESTED	RESULT	CONFIDENCE
1	The reported sandbox file-access vulnerability was exploited against the investigated setups	**NOT SUPPORTED**, surface structurally absent (§5.1, §5.2, §5.3)	HIGH
2	Transfer-layer server-side request forgery was exploited	**RULED OUT** (§5.2)	HIGH
3	Remote code execution was obtained on provider machines by any means	**NOT SUPPORTED** (§5.4 to §5.6)	HIGH
4	The provider software was compromised through the supply chain or tampered with after installation	**NOT SUPPORTED** (§5.4)	HIGH
5	Unauthorised remote access occurred via the remote-login service	**NOT SUPPORTED** (§5.5)	HIGH
6	Third-party persistence was established on provider machines	**NOT SUPPORTED** (§5.5)	HIGH
7	A binary-replacement rootkit exists in system directories outside the captured tree	**NOT EXCLUDED** (§8)	LOW
8	A single unused key represents dormant remote-access persistence	**NOT RESOLVED** (§8)	LOW

7 Provider-Side Configuration Observations

The following did **not** contribute to any observed compromise. They are reported because they bear directly on the impact the reported vulnerability would have had if it had been reachable, and because they are within operators' control today. They are stated generically and apply to some but not all of the investigated setups.

Credential material stored in cleartext in operator scripts. On one setup a provider API key was stored in cleartext inside root-owned helper scripts. The management interface it authorises is loopback-bound, so using the key requires code already executing on the machine, but that is

precisely the position an escaped workload or a file-read primitive occupies. Such a key is not read-only: it authorises the full management surface, including payment operations. A file-read primitive reaching this file would have been converted from “read one file” into “control the provider instance and its wallet operations”. **This is the observation that most directly amplifies the reported vulnerability class, and should be the primary remediation item for operators.** Keys should be rotated, stored in restricted-permission files sourced at runtime, and scoped through the software’s origin-restriction setting.

Provider processes running as root without a container boundary. Some setups initially ran the provider directly as root on the host before migrating to a containerised model, serving remote workloads with only the hardware-virtualisation boundary between requestor-supplied code and full host privilege. No breach indicator was found on any of them; the later containerised model is the correct one.

Containers running with all isolation disabled, on a flat management network. Where containerisation was used, containers ran fully privileged, effectively disabling capability, device and mandatory-access-control confinement, and were attached directly to the operator’s production local network as first-class hosts rather than isolated tenants. A hypothetical guest escape therefore lands not in a constrained container but on the same segment as the operator’s other machines and management infrastructure. Equivalent functionality is achievable with a narrowly scoped device grant and a single added capability, on an isolated segment.

No host-side network filtering. Host firewalls were installed but disabled, and packet-filter rule sets were empty.

Detection capability degraded by monitoring volume. A high-frequency management polling loop generated hundreds of thousands of accepted sessions per week per setup. The sessions are legitimate, but at that volume they saturate the authentication log and would effectively mask a genuine intrusion. A persistent connection or a metrics agent would preserve the monitoring function without destroying the log’s evidentiary value.

Two positive controls are worth recording. Workload virtual machines were launched with no host filesystem share and with network interfaces bound to loopback, one of the reasons the reported vulnerability had no surface here. And the provider access-control rule file was byte-identical to the shipped default across every recovered image, confirming the shipped defaults were never weakened.

8 Limitations and Residual Uncertainty

This report states its limits rather than inferring past them.

Scope of the sample. The conclusion in §1 applies to the collected samples: 33 provider setups. It is not a statement about the network as a whole. Setups outside this sample were not examined and no claim is made about them.

System binaries outside the captured tree. For a subset of setups the acquisition did not include system binary directories, so a binary-replacement rootkit residing exclusively there cannot be affirmatively excluded for those setups. No indicator suggesting one was found in any class that *was* captured (package transaction logs, service definitions, scheduling configuration, deleted-object enumeration), but this is corroborating absence, not exclusion. Closing the gap requires a full root-filesystem capture or package-verification output from the running systems.

An unused key of undetermined purpose. One setup carries an additional authorised key present on no other setup and never used in the full retained authentication window. Its embedded comment names the operator's own management infrastructure and there is no other anomaly on that setup, so the probable reading is benign. It is nonetheless the only key material in the sample confined to a single machine, and an unused key is also what dormant persistence looks like. The evidence cannot distinguish the two, so it is reported as unresolved rather than assumed benign. The operator should confirm its purpose and remove it if not required.

Direct hash verification of the deployed release. An archive of the exact deployed release was no longer retrievable from the distribution point at the time of examination, so a direct comparison against an official archive was not possible. The integrity conclusion rests instead on the independent-replication and provenance arguments in §5.4, which are strong. Publication of historical release checksums would close this gap permanently and cheaply.

Incomplete execution records for early operating windows. A portion of the workload activity predates the working directories that survived cleanup, and no per-activity execution log exists for those windows; this material is not recoverable. The assessment for those windows rests on the identical requestor population, identical cached task-package digests, and the absence of any host-side anomaly across the same period, a weaker basis than the direct enumeration in §5.1, and identified as such.

9 Recommendations

For the platform

1. **Publish release checksums, including for historical versions.** A low-cost change that converts the strongest remaining integrity gap in this investigation into a one-command check for any future one.
2. **Preserve per-activity execution records for longer, or make retention configurable.** The single most decisive evidence class here, the complete command vocabulary in §5.1, survived by chance, in a minority of working directories not yet cleaned. A short, bounded retention would make that evidence available by design rather than by accident.
3. **Ship provider defaults that assume the workload is hostile.** The two positive controls in §7 are the reason the reported vulnerability had no surface on these setups. They should remain non-optional defaults, and any future feature that maps host paths into the workload environment should be treated as re-opening this vulnerability class.

4. **Provide first-class guidance for containerised provider deployment,** so operators are not left to derive privilege and network configuration themselves. The configurations in §7 are the predictable result of an absent recommended baseline, not of operator carelessness.

For provider operators

1. **Rotate any credential stored in a script, and stop storing provider API keys in scripts.** Use restricted-permission files sourced at runtime, and set the software's origin restriction.
2. **Do not run the provider as root on the host.** Use the containerised deployment model.
3. **Do not run provider containers fully privileged.** Grant the specific device access required and the single network capability needed, nothing further.
4. **Place provider containers on an isolated network segment,** separate from any management or production network.
5. **Enable host-side network filtering.**
6. **Review authorised-key material on every machine** and remove keys not in active use.
7. **Replace high-frequency polling loops with a persistent connection or a metrics agent,** so the authentication log remains usable as evidence.

10 Closing Statement

The investigation examined 33 provider setups against a specific, credible, independently reported vulnerability, using a methodology designed to falsify rather than to confirm. **No evidence of remote code execution, unauthorised remote access, or exploitation of the identified vulnerability was found across the collected samples.** In the observed configuration the vulnerability class was not merely unexploited but structurally unreachable, and the provider software was verified as unmodified across every setup examined.

The residual uncertainties in §8 are stated rather than resolved by inference, and are individually closable. The configuration observations in §7 are the actionable output of this work: they did not cause any compromise, but they define how much a comparable vulnerability would have cost had it been reachable, and that is the margin worth reducing.

References

1. Kent, K.; Chevalier, S.; Grance, T.; Dang, H. *Guide to Integrating Forensic Techniques into Incident Response.* NIST Special Publication 800-86, National Institute of Standards and Technology, August 2006. DOI: 10.6028/NIST.SP.800-86
2. ISO/IEC 27037:2012. *Information technology. Security techniques. Guidelines for identification, collection, acquisition and preservation of digital evidence.* International Organization for Standardization, 2012.
3. SANS Institute. *FOR508: Advanced Incident Response, Threat Hunting, and Digital Forensics.*
4. SANS Institute. *FOR577: Linux Incident Response and Threat Hunting.*
5. Carrier, B. *The Sleuth Kit (TSK).* https://www.sleuthkit.org

END OF PUBLIC SUMMARY REPORT.