STATIC SCAN REPORT ON IRI COSORT-COMPATIBLE COMMAND-LINE MODULES AS OF COSORT 10.5.1

Background & Deliverable

IRI recognizes the importance of eliminating security vulnerabilities from its products, and is committed to doing so with consideration given to software performance and risks foreseeable under actual use conditions.

IRI conducts regular security testing of its software delivered to external sites which rely on the execution of CoSort and the CoSort SortCL data manipulation program. In most cases, IRI ships the IRI CoSort Command Line Interface (CLI) in the $COSORT_HOME/bin and /lib directories on Linux and Windows.

More specifically, the IRI CoSort Version 10.5: Sort Control Language (SortCL) program ("sortcl"), product setup utility, Unix sort program replacement, and metadata conversion utilities which generate SortCL script, are provided in the $COSORT_HOME/bin (%COSORT_HOME%\bin) directory. In addition, the CoSort engine "libcosort" -- which is used by sortcl and can be linked with customer programs -- is distributed with other CoSort API routines in the $COSORT_HOME/lib directory. Libraries used for custom field functions in sortcl jobs are in $COSORT_HOME/lib/modules.

Results

IRI recently performed static security scans using *cppcheck* on the sortcl program and CoSort library, and on the setup and utility programs delivered with the CoSort v10.5 installation.

The scans produced 10 errors and 295 warnings. All 10 errors were examined and determined not to be actual problems. Warnings were examined and those that might have potential problems were also eliminated.

The updates are in a separate source code branch and can be merged into the main source branch for delivery to ACI in subsequent patches or releases.

Errors reported in the static scan are as follows:

1) sortcl\joiner.c, Memory leak: script.S
   When performing not_sorted JOIN, a temporary sort job script is created but not released after it is passed to a new sortcl instance.

2) engine\lzoconf.h,"#error ""invalid CHAR_BIT""
   A compiler declaration in a 3rd party library used for file compression and decompression. The program will not compile if this error is encountered.

3) crypt\ptypes.h,#error Unknown processor target in ptypes.h.
   A compiler declaration in a 3rd party library used for encryption and decryption. The program will not compile if this error is encountered.

4) conch\conch.c,Memory leak: m
   A 128 byte allocation that actually is freed when the module is closed before exiting.

5) pcre\pcre_internal.h,#error
   Cannot determine a type for 16-bit unsigned integers. Another compiler declaration in a 3rd party library. The program will not compile if this error is encountered.

6) cla4db2\sqlusortchklvl.c,Boolean value assigned to pointer.
   This is in the (unused) cla4db2 Load Accelerator for DB2 utility.

7) sorti2scl\sorti2scl.c,Memory leak: scriptinfo.outputfilename
   This is a utility program that converts sorti syntax into a sortcl job script. The sorti program is obsolete and no longer delivered in CoSort.

8) yacc.c,Common realloc mistake: 'outfiles' nulled but not freed upon failure. This C code is auto generated by Bison, a Lexical Analysis and Parsing tool. It is used when building our cob2ddf utility that creates sortcl job scripts from COBOL copybook metadata.

9) yacc.c,Common realloc mistake: 'outfiles_dd' nulled but not freed upon failure. See #8 above.

10) yacc.c,Common realloc mistake: 'outfiles_used' nulled but not freed upon failure. See #8 above.

Considerations

It is important to recognize that the CoSort sortcl program and sortcl library modules, CoSort sort engine APIs, and associated utilities (mostly for metadata conversion) are application programs that run an individual instance of a specified data processing job. Each instance runs with the level of privileges of the user that started the job. Additional database user credentials are required for access to any database. Access to data is restricted by the Operating System to the access rights of the user that started the program.

When a program repeatedly allocates memory but does not release it when no longer needed, it has what is called a memory leak. These can accumulate causing severe performance problems and eventually the program or system to crash. IRI has eliminated most memory leaks except just a few that do not accumulate. We will continue looking for a solution to the few remaining. These would be released just before the program terminates. All the memory is released when the program terminates after each job script.

The sortcl program and CoSort engine run individual job scripts and are not web-based or interactive applications. The system reports errors when the program attempts to access any resources the current user does not have proper access to, resulting in termination of the program with an error report.

IRI Responses to OWASP Top 10 Security Vulnerabilities

1) Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

Sortcl cannot be run by an unprivileged user who can not already access the database in other ways. Credentials to access the database are contained in the sortcl job script and enforced by the DBMS. The database query is usually created from field or ODEF names written into the job script field statements. Sortcl also supports /QUERY to allow a SELECT statement to be specified inside the job script. Sortcl jobs run a prepared job script and do not use an interactive user interface.

2) Broken Authentication. Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

Authentication is performed by the OS and database and not our software. Sortcl runs with the current user privileges.

3) Sensitive Data Exposure. Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

Sortcl is not a web application but can use SFTP for encrypting network traffic when sending or receiving data across the network. Sortcl supports AES256 encryption and decryption, format preserving encryption, and others. Sensitive input and output can be stored in encrypted form and decrypted while being processed, with results encrypted prior to output. Custom functions for field data or IO may also be applied. The sortcl user can not access data without permission from the OS or DBMS.

4) XML External Entities (XXE). Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and DoS attacks.

Sortcl does not evaluate external references in XML files.

5) Broken Access Control. Restrictions on what authenticated users can do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

Access control for sortcl is enforced by the OS. Sortcl currently runs as the current user and does no privilege escalation.

6) Security Misconfiguration. Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.

The privileges of users to access specific files is controlled by the system and not by sortcl. Sortcl runs only with the access rights of the current user.

7) Cross-Site Scripting XSS. XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim’s browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

Sortcl is not an interactive program or web application and does not use a browser interface. Sortcl may output HTML, but the user already must have full access.

8) Insecure Deserialization. Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

Sortcl can only send output to the destination specified in the script and accessible by the current user. Sortcl does not perform remote code execution.

9) Using Components with Known Vulnerabilities. Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

Sortcl is not using components with known vulnerabilities other than supporting old protocols such as FTP when specified in the job script. The network and external client/server/system file are not part of sortcl or CoSort. Their security is the responsibility of the system Administrator. Sortcl also supports the sftp protocol. See the separate OWASP report on IRI Workbench.

10) Insufficient Logging & Monitoring. Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

Sortcl includes several optional levels of logging, including the .cserrlog, statistics log, audit log, and monitor level.

IRI Responses to SANS Common Weakness Enumeration Top 25, CERT Secure Coding

[1] CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

The O/S will detect this and signal CoSort to report the error and terminate.

[2] CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Sortcl can process HTML data as text as I/O but does not run or interpret HTML or any language other than the proprietary sortcl job script language.

[3] CWE-20 Improper Input Validation

Improper input results in program termination and error report. Whenever the source code changes IRI runs a test suite including thousands of tests to verify functionality.

[4] CWE-200 Information Exposure

Data access is restricted to read, write and execute permissions specified by the operating system for the current user. No data can be exposed that the user does not already have permission to access.

[5] CWE-125 Out-of-bounds Read

Static scans reduce probability of this and the OS should detect the invalid access and signal CoSort, resulting in an error report and program termination.

[6] CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Injection is not possible because sortcl uses a script language that does not allow users to dynamically construct a database query. Sortcl can not be run by an unprivileged user that can not already access the database in other ways.

[7] CWE-416 Use After Free

The O/S will detect this and signal CoSort to report the error and terminate.

[8] CWE-190 Integer Overflow or Wraparound

The O/S will detect this and signal CoSort to report the error and terminate.

[9] CWE-352 Cross-Site Request Forgery (CSRF)

Cosort is not a web application so this is not possible. Credentials are supplied with the job script. Access control for sortcl is enforced by the OS. Sortcl currently runs as the current user and does no privilege escalation.

[10] CWE-22 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

CoSort recognizes relative or absolute paths which must be provided in the Sortcl job script. Access is restricted to the sortcl user privileges.

[11] CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

Sortcl allows an OS command to be run when specified in the sortcl script, but access rights are *restricted to the current user privileges, so the user must already have permission to run the command.*

[12] CWE-787 Out-of-bounds Write

The O/S will detect this and signal CoSort to report the error and terminate.

[13] CWE-287 Improper Authentication

Authentication is performed by the OS or DB and not sortcl. Access rights are *restricted to the current user privileges.*

[14] CWE-476 NULL Pointer Dereference

Sortcl source code uses methods to prevent this from occurring, but if it does, the O/S detects this and signals CoSort to report the error and terminate.

[15] CWE-732 Incorrect Permission Assignment for Critical Resource

Permissions are specified and verified by the OS and not controlled by sortcl. It is the system Administrator's responsibility to specify access restrictions.

[16] CWE-434 Unrestricted Upload of File with Dangerous Type

Data sources and destinations are specified in the sortcl job script. Sortcl will recognize and decompress gnu-zipped input but otherwise ignores the file type. Sortcl does not run program files specified as data sources or output, but can execute a program when specified in the sortcl job script. Sortcl runs with permissions of the current user. It is the system Administrator's responsibility to specify access restrictions.

[17] CWE-611 Improper Restriction of XML External Entity Reference

Sortcl does not evaluate external references in XML files.

[18] CWE-94 Improper Control of Generation of Code ('Code Injection')

Sortcl will not perform remote code execution.

[19] CWE-798 Use of Hard-coded Credentials

Sortcl does not evaluate credentials but supports multiple methods for providing credentials that are evaluated by the OS or DB. Credentials may be hard coded in the sortcl job script, an external file, or come from an encrypted key source.

[20] CWE-400 Uncontrolled Resource Consumption

IRI provides a resource control file to set system resource limitations.

[21] CWE-772 Missing Release of Resource after Effective Lifetime

Sortcl is an application program that runs one job script at a time. Resources are released when sortcl is finished with them or the instance terminates.

[22] CWE-426 Untrusted Search Path

Sortcl access control is enforced by the OS using current user access rights.

[23] CWE-502 Deserialization of Untrusted Data

Sortcl does not perform remote code execution. Sortcl can only send output to the destination specified in the sortcl job script and accessible by the current user.

[24] CWE-269 Improper Privilege Management

SortCL does not control privilege management; that is the responsibility of the system administrator. Sortcl is restricted to the rights of the current user.

[25] CWE-295 Improper Certificate Validation

Sortcl is not a web interface or web application. sortcl does no Certificate Validation. Any certificate validation would be performed by software on the system external to sortcl.

Final/Summary Points

1. All scanning errors reported reflected non-compliant code with the scanner itself rather than a salient security vulnerability.
2. Any memory-related errors are already handled by O/S interrupts (seg faults) which SortCL traps and automatically handles with clean exits and removal of open temporary files.
3. SortCL is a command line program run by authorized users on specific, tested job scripts which you create for known data. It is not run as root and offers no route to privilege escalation.
4. SortCL and Workbench are not web applications, so do not open any network ports.
5. IRI would be happy to discuss this report, and the applicability of additional scans given the facts above.

IRI Workbench GUI and IRI DarkShield Software Composition Analyses

Preface

Software composition analysis scans were performed on both the IRI Workbench front-end job design IDE for Voracity platform software (built on Eclipse) and back-end IRI DarkShield API routines performing data discovery and masking using the OWASP Dependency-Check software composition analysis (SCA) tool. This type of scan analyzes software dependencies for Common Vulnerability and Exposure (CVE) entries, based on the identified Common Platform Enumeration (CPE), and chosen because it works on Java software like Workbench and DarkShield.

Synopsis

There were several hundred dependencies from the scans of over 20K dependencies which came back listed as having **potential** vulnerabilities. Many of them are false positives or duplicates.

All vulnerabilities in the SCA report relating to Eclipse (IRI Workbench) are false positives. They all mention older versions of Eclipse than the actual version of Eclipse which IRI Workbench uses is 4.28/06 -2023. Several other vulnerabilities in the report are also false positives based on the descriptions of the vulnerabilities (the version of the dependency actually used is greater than the version of the dependency the vulnerability exists in).

Several of the vulnerabilities in the report are also not even pertinent to the way the dependency is used. For example:

> **CVE-2023-33201:** Bouncy Castle For Java before 1.74 is affected by an LDAP injection vulnerability. The vulnerability **only** affects applications that use an LDAP CertStore from Bouncy Castle to validate X.509 certificates. During the certificate validation process, Bouncy Castle inserts the certificate's Subject Name into an LDAP search filter without any escaping, which leads to an LDAP injection vulnerability.CWE-295 Improper Certificate Validation

DarkShield is **not** an application that uses an LDAP CertStore from Bouncy Castle to validate X.509 certificates.

For the types of vulnerabilities that are not false positives in the DarkShield Software Composition Analysis (SCA) report, the type of CWE (Common Weakness Enumeration) relates to excess use of system resources (CWE-770 Allocation of Resources Without Limits or Throttling) or similar, and the vulnerability can only be triggered in very particular cases.

An example of one of these types of vulnerabilities follows:

```
In Apache PDFBox, a carefully crafted PDF file can trigger an OutOfMemory-Exception
while loading the file. This issue affects Apache PDFBox version 2.0.23 and prior
2.0.x versions.
CWE-770 Allocation of Resources Without Limits or Throttling
```

While such vulnerabilities may technically be exploitable, they could only occur if someone were aware of, and able to intentionally exploit, the vulnerability. As importantly, the effect of the vulnerability is simply excessive usage of system hardware resources like CPU or RAM, which is a different category of CWE than unintended code execution, corruption of data, or leakage of sensitive data.

Furthermore, since both the DarkShield front-end and back-end are Java applications, only the allotted maximum amount of memory for the JVM will be used before an Out of Memory Exception occurs. This is typically much less than the total amount of memory on the system.

CWEs come in different categories, of which an organization may deem to have different levels of impact and severity. Some types of CWEs include allowing adversaries to completely take over a system, steal data, or prevent applications from working. The weaknesses found in this SCA scan are all of that type which could prevent an application out of memory from working.

These vulnerabilities could be remediated by updating to the latest version of the dependency. However, because several of the dependencies have breaking changes in the current version, updating them would require changes in the DarkShield application. Several weeks of headway would be necessary to address them; this is something IRI does not believe to be worthwhile given the remote chances of limited resource overuse.

Finally, it should also be noted that:

1) DarkShield is a self-managed application. Therefore, managing access to it is in the hands of the user.
2) Data processed by it remains in the user's own managed environment. The O/S user given access to DarkShield should be given the least amount of permission necessary.
3) Any credentials needed to access data silos are encrypted, and not stored in plain text at rest in DarkShield job configuration files.