# Security Measures for Critical Software

*Submitter:* *Eric Brewer, on behalf of Google, LLC*
*Topic:* *(3) Guidelines outlining security measures for critical software*
*Speakers:* *Jon McCune, Principal Software Engineer, jonmccune@google.com*

Even software with a sound provenance record can lead to an insecure system if that system is not designed and configured appropriately. Because security is never perfect, mechanisms must be in place to mitigate and recover from vulnerabilities or attacks. This paper offers guidance for mitigation: managing configuration, reducing privilege, and network segmentation, that have withstood the test of time at Google.

## Proper Configuration

**Patching and Updates**. Patches and updates should be supported as part of day-to-day operations with frequently-used procedures. This practice ensures that vulnerabilities are addressed as soon as possible, and reduces availability risk because the procedures are part of normal operations.

**Configuration Management**. Software systems should use revision control systems to store configuration data, similar to those used to manage source code and release artifacts, and should separate configuration data from executable code such as scripts. These techniques reduce the complexity of the deployed software system and make it easier to reason about during an incident.

**Rollback Support**. Ensure that tooling supports return to a known-good state. This allows rapid and confident response to a recent problematic change, such as a feature that is discovered to contain a security vulnerability. This is an area where configuration management is useful, since revision control systems have rich support for tracking and managing multiple releases.

**Know Your Hardware**. Do not forget to manage hardware-specific artifacts, such as firmware for motherboards, peripherals, and management controllers. These highly privileged components may otherwise be left with outdated and vulnerable firmware. These management requirements should be part of the hardware selection criteria, as vendors' security practices vary widely.

## Least Privilege

**Compromise Containment**. Identify the boundaries between system components of varying levels of trust, and take action to contain the less trusted components. For example, use sandboxing, system call filtering, separate virtual machines, or distinct identities to introduce barriers to moving laterally from one compromised component into another. Good software configuration and system design anticipates that arbitrary system components may become compromised, and seeks to contain the damage.

**Disable Unused Features.** Enable only those features that are necessary and have an explicit use case. This limits the attack surface, since many software packages enable large numbers of features by default. Other features can always be enabled in the future, as needs or use cases change.

**Credentials**. Avoid use of bearer tokens, avoid transmitting cleartext credentials, and enable multi-factor authentication. This protects against adversaries who harvest credentials committed to revision control repositories, cookies or tokens via compromised client devices, and credentials accidentally captured in debug logs.

**Multi-party controls**. Use multi-party controls so that a single person cannot unilaterally destroy or leak data, or maliciously modify the state of critical systems. Pay particular attention to highly privileged operations, accidental and malicious actions, and compromised or stolen devices.

**Audit and Notification**. Privileged operations should generate an audit record and notification to other administrators. This can alert responders when a compromised account takes an unexpected action, or when a risky action may be unavoidable, such as responding to an unanticipated incident. Ensure that logs are write-once to the systems generating the log data, so that a compromise cannot lead to destruction or tampering with the log data. The audit trail can aid in post-mortem forensics.

**Encrypt**. Encrypt data at-rest and in-transit. This both protects confidentiality and ensures that adversaries cannot inject malicious traffic such as administrative commands. Confidential Computing is a new technology that also offers a layer of protection for data in use.

**Root of Trust**. Incorporate root-of-trust hardware into operating systems and management software to protect sensitive keys and represent ground-truth for system identity. This is the strongest available signal that "this machine is one of ours". Attestation of software and configuration can confirm that systems are in the intended state while only having to trust a small fraction of the system as a whole.

## Network Segmentation

**Availability-only**. Trust the network only for availability, and avoid systems that grant access to resources simply by being connected to the network. In practice, there may be factors that still lead to some network trust, such as legacy devices, third-party software or hardware that cannot be modified, but the overall attack surface is still reduced.

**Limit reachability**. Partition networks so that the only permitted connectivity corresponds to an explicitly named need. Do not allow internet access for systems that do not require it, and build explicit processes around internet reachability to inhibit malicious data exfiltration.

**Proxy**. Use a proxy between network segments whenever performance permits. This enables logging and auditing, and can serve as a first line of defense for preventing traffic matching known malicious patterns when a new attack is discovered. The proxy can also enforce best practices, such as ensuring that clients use https even if some legacy servers internally might serve both http and https.

**Log and Monitor**. Inspect traffic to ensure that all in-use protocols are known, and that all encrypted protocol types are actually encrypted. At high bandwidth, it will only be possible to do statistical sampling, but it is still worthwhile. Where deep packet inspection is necessary, consider proxy designs that do not have the keys necessary to violate packet integrity (as compared to confidentiality).