Fortifying the Bioeconomy

Leveraging Shared Responsibility Security for Hardware and Software

bioisac
Creating Collaborative Threat Intelligence for the Bioeconomy

From electronic lab notebooks to industrial control systems and the artificially-intelligent design of experiment systems, digital technologies have become an integral part of innovation and scalable production in the bioeconomy. So much so, that the continued growth of the bioeconomy relies on the safe operation and development of these critical digital technologies. Recent digital attacks on bioeconomy infrastructure at Oxford University[1], Miltenyi Biotech[2], and Dr. Reddy's Laboratories[3] as well as similar attacks on other critical sectors such as the power grid, oil and gas, cold chain, and wastewater treatment, have shown how vulnerable operations are to such attack and manipulation.

To respond to this need, the Hardware & Software Security Working Group at BIO-ISAC was created to provide scalable security practices for assets in the bioeconomy.

Prior research quickly identified that the use of a shared responsibility model would enable Original Equipment Manufacturers (OEMs) and Asset Owners to securely work together with clear lines of accountability for developing and maintaining the security hygiene of Assets. Scaling security practices across OEMs and Assets Owners through a shared responsibility model is the critical step towards building a resilient and safe manufacturing ecosystem. This document focuses on defining two key models of Shared Responsibility Security: 1/ Secure Software Development Lifecycle (SSDLC) and 2/Secure Hardware Development Lifecycle. Asset Owners are also provided with implementation plans for strengthening the overall security posture of their digital technologies and the steps to actively practice defense-in-depth for manufacturing and research.

The Model for Shared Responsibility of Bioeconomy Systems

With the growth of cloud based services, the concept of shared responsibility formalized the **Shared Responsibility Model**[4]. This model clearly defines management responsibilities between a Cloud Service Provider (CSP) and Asset Owner but was built to be primarily focused on cloud based systems. When a non-cloud based, packaged system is purchased, the hardware and software generally reside within the Asset Owner's network. Unless the system requires custom code to support the Asset Owner needs, the Asset Owner generally is responsible for everything except the code within the Asset. The condition of the system's code is where the Asset Owner may have to fully rely on an OEM to deliver system updates and fixes for security vulnerabilities, data protection measures, and all other features that enable the use of the system in a secure manner.

1 https://www.forbes.com/sites/thomasbrewster/2021/02/25/exclusive-hackers-break-into-biochemical-systems-at-oxford-uni-lab-studying-covid-19/?sh=7bf1c0ab2a39

2 https://www.bleepingcomputer.com/news/security/biotech-research-firm-miltenyi-biotec-hit-by-ransomware-data-leaked/

3 https://www.thehindu.com/business/Industry/oct-22-data-breach-involved-a-ransomware-attack-dr-reddys/article32962438.ece

4 https://cloudsecurityalliance.org/blog/2020/08/26/shared-responsibility-model-explained/

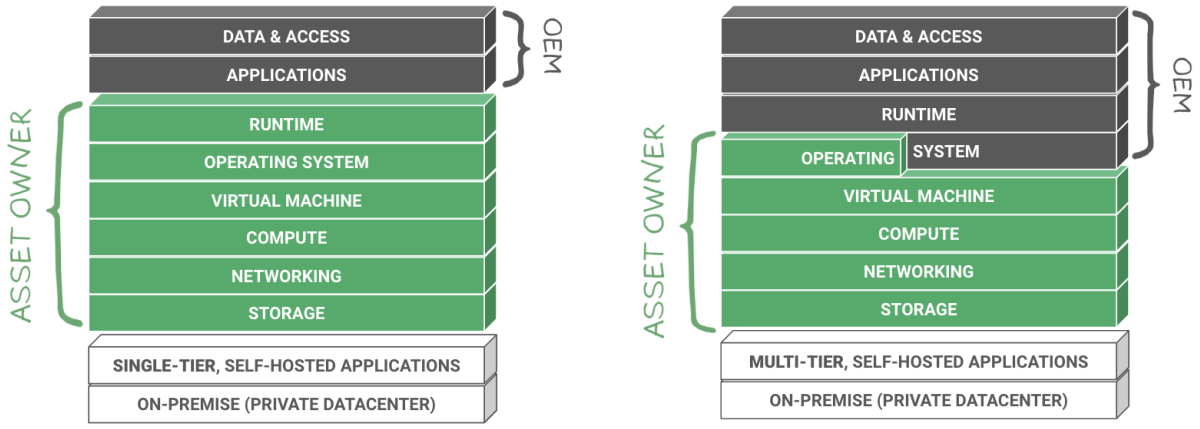
Shared Responsibility Models in Hardware and Software

Shared responsibility across all hardware and software decisions will strengthen and secure the bioeconomy by starting at the front line tools used in manufacturing, storage, and distribution.

Assets come with a wide range of designs (single-tier applications, multi-tier applications, cloud applications, etc.) and goals for security and use, which may require different shared responsibility models to fit their use case, security needs, and threat vectors.

Two examples are offered below:

Shared ownership between the Original Equipment Manufacturer (OEM) and Asset Owner



On the left, single-tier, self-hosted applications with an on-premise, private datacenter have opted to put the majority of risk management for security and maintenance on the Asset Owner. On the right, the multi-tier, self-hosted applications with a similar on-premise, private datacenter have opted to balance the risk management model nearly equally between partners. The key component of shared responsibility is to have defined roles and responsibilities for each part of the stack, including examples of what response, communication, coordination, and timelines are expected.

To robustly mobilize on behalf of the shared responsibility model of securing software and hardware throughout the bioeconomy, the following actions are required.

Securing Software and Hardware within the Shared Responsibility Model

Both software and hardware offer tremendous opportunities to define and refine the security standards for an organization, at the point of acquisition all the way through to the maintenance of each item.

Secure Software Development Lifecycle

The Secure Software Development Lifecycle (SSDLC) outlines the steps for security diligence throughout development, testing, and release to measure security effectiveness. A defined SSDLC is critical for the creation and maintenance of a system's security integrity. SSDLC will generally only apply to OEMs and for use in these key areas:

1. **Security Testing.** During this stage OEMs should be reviewing all proprietary code and open source code (software supply chain). This should include an evaluation against the Common Vulnerabilities and Exposures (CVEs) repository that has been published and is updated daily by The MITRE Corporation[5]. A threat analysis should drive the remediation of identified vulnerabilities based on the severity of a CVE and the likelihood of exploit based on the controls within the system, not anticipated controls within an Asset Owner's environment. The outcome of the OEM's threat analysis should result in remediation of identified vulnerabilities prior to release of software Asset Owners. In addition, OEMs should systematically review all code, daily, for newly published vulnerabilities that require patch releases to Asset Owners. The OEM should implement a secure coding framework such as BSIMM[6] or OpenSAMM[7] and follow secure coding best practices such as the PLC Top 20[8] or OWASP Top 10[9] while developing their Assets.

2. **New Feature Evaluation.** With the development of new features, a threat modeling[10] based evaluation of risk posed by such features should be performed. The risk evaluation should enable secure use by default and discourage Asset Owners from having the ability to implement known, unsecure, protocols or poor security practices. Preconfigured packaged systems should also abide by these practices directly from the factory. For example, when developing Single Sign-on (SSO) support the feature should not permit the transportation of username and password in clear text or the implementation of file sharing functionality such as SMB should not use a version of a protocol that has known exploits (SMBv1[11]).

---

5 https://cve.mitre.org/

6 https://www.bsimm.com/

7 https://www.opensamm.org/

8 https://plc-security.com/

9 https://owasp.org/www-project-top-ten/

10 https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling

11 https://www.tenable.com/plugins/nessus/100464

3. **Source Code Protection.** A source code repository, software utilized for managing source code, is the location where a malicious actor or insider threat would target to inject malicious code (viruses, worms, Trojan horses, logic bombs, spyware, adware, and backdoor programs, etc.) or to steal intellectual property. To prevent theft and malicious code injection, OEMs should focus on implementing: 1/ Single Sign-on (SSO) for GUI and CLI, 2/ Multi-Factor Authentication (MFA), 3/ Programmatic source code repository activity monitoring, 4/ Back-up code to separate host with service specific permission, and 5/ Require two or more approvals before code promotion*. These controls are critical for ensuring the integrity of an Asset.
   *Such approvals should be embedded in the code promotion process to prevent control circumnavigation.

4. **Software Bill of Materials.** As more components are utilized it creates greater emphasis for OEMs to have an accurate inventory of what type and version of code, operating systems, firmware, etc. are within the Asset. This type of inventory is known as a Software Bill of Materials (SBOM) and is offered in multiple types of standardized such as SPD or CycloneDX. The components within the SBOM outline where security considerations must be taken during development, see New Feature Evaluation, and monitored on an ongoing basis for security updates by the OEM.

Secure Hardware Development Lifecycle

Lab or manufacturing equipment may include one or many hardware components (Desktop Computers, PLCs, HMIs, etc.) that are of proprietary design or provided through third-parties. A Secure Hardware Development Lifecycle (SHDLC) should enable Asset Owners to perform updates or repairs to maintain hardware security to respond to security threats in a timely manner and trust that steps have been taken to verify the trustworthiness of an OEM's supply chain. To achieve this goal, the following objectives have been developed:

1. **Security Testing.** Similar to SSDLC, hardware must undergo routine security testing. Testing may include following a secure coding practice for PLCs[12] and remediating desktop operating system, embedded operating system, and firmware vulnerabilities to maintain the integrity of the Asset Owner's security controls. Due to the degraded security posture of Assets, threat actors do not need to rely on complex zero-day vulnerabilities to perform malicious actions. Publicly known vulnerabilities with well-documented exploits remain unremediated and easy targets for threat actors to exploit. With the implementation of basic cyber hygiene practices within Assets, Asset Owners can begin to narrow down probably threat vectors for true defense-in-depth.

2. **New Equipment Evaluation.** As new tools, accessories, or equipment become available for the hardware, particularly in the area of security and defense, an equipment accessory risk assessment must be conducted to determine if the new accessory enhances or creates new security risks or compromises the security of the established hardware.

[12] https://plc-security.com/

3. **Secure by Design.** Should the Asset contain multiple hardware components, the Asset should be treated as its own independent collection of components that enforce multiple security practices such as independently authenticated and authorized services, network traffic segmentation with managed switches, and vulnerability management for embedded operating systems and firmware. Evaluating and maintaining these controls enables the Asset Owner to implement defense-in-depth for simple or complex systems by enforcing segmentation that minimizes traversal capabilities of threat actors and malware.

4. **Hardware Bill of Materials.** Similar to the SBOM for software, each piece of hardware should include a detailed inventory of components, design documentation outlining security intentions, threat model, utilized security verification methods, and repair pathways through a Hardware Bill of Materials (HBOM).

5. **Hardware Provenance.** In addition to the detailed description of what parts are included through the HBOM, a detailed inventory of the companies, contractors, and subcontractors contributing to the manufacturing of this software must be included. This allows companies to avoid engaging with vendors having relationships with known bad actors or foreign adversaries.

Secure Reset & Disposal

While the bulk of this narrative has been oriented towards securing Assets from the factory and throughout their useful life cycle, there is a commonly overlooked threat vector that must be addressed by OEMs. The ability to securely reset a device for disposal or for resale. Assets may outlive their Asset Owner or reach end of life after years of service and data generation. Both of these scenarios introduce the need for different workflows to protect an Asset Owner's information and/or usability of the Asset. OEMs should develop manual or systematic methods to enable recommissioning of an Asset to a new Asset Owner and Asset disposal in a manner that does not reveal network related information as well as purging all data from previous use.

Value of Engaging a Shared Responsibility

The attempts by Nation State backed attackers and organized crime organizations to steal intellectual property during the COVID-19 pandemic were a sobering reminder that the bioeconomy is not adequately prepared for the level of sophistication presented by Nation State attackers. Individuals linked to AstraZeneca disclosed observed attempts by North Korea to obtain access to their network and Microsoft separately stated their observation of North Korea hacking groups targeting vaccine developers in multiple countries.[13] The protection of manufacturing quality, stability of research environments, and intellectual property cannot be achieved by an OEM or Asset Owner alone, only together can it be achieved. Shared responsibility by both parties is the first step towards adapting our bioeconomy to protect itself from sophished, well funded, threat actors who introduce heightened risk to an already fragile

13 https://www.cnbc.com/2020/11/27/suspected-north-korean-hackers-targeted-covid-vaccine-maker-astrazeneca-sources-say.html

ecosystem. The ability to respond to cyber threats must also be met with direct guidance from our Regulators such as the Food & Drug Association (FDA) and European Food Safety Authority (EFSA) to safely advise Asset Owners on the compliant path to respond in a timely manner to these threats. The reduction of friction to address cyber threats between OEMs, Asset Owners, and Regulators is the responsible path forward to enable sustainability and the safe development and fortification of our bioeconomy.

Implementation

To further support the implementation of the necessary security controls for OEMs and Asset Owners, BIO-ISAC has developed the **Biosecurity Evaluation Questionnaire** (BSEQ) to assist OEMs with improving their Asset security practice and for Asset Owners to evaluate and grade the practices within their OEMs. All materials are available at **isac.bio/device**.

While the BSEQ will help you develop your program and some supporting Asset features, a resource with greater security granularity is available to help your Asset teams understand the technical configurations they should be prioritizing. The MITRE Corporation (MITRE ATT&CK) has published techniques utilized by malicious actors when specifically attacking Industrial Control Systems (ICS)/ Operational Technologies (OT)[14]. The OT techniques described by MITRE should be referenced for impact during feature development and incorporated into user stories during development and control effectiveness during testing.

Conclusion: The Case for Fortifying the Bioeconomy

Vulnerability management across Asset lines, pre- and post- sale, is mandatory in order to secure digitally advancing environments in the bioeconomy and requires active advocacy from the teams making these decisions. Taking action to deploy and maintain well designed SSDLC and SHDLC protocols will reduce overhead expenses in security while also enabling development teams to move with great velocity and job satisfaction. Lastly, Using technical controls to protect source code will protect both your intellectual property and the end user clients, from finding themselves the victim of an attack.

By implementing the software and hardware security best practices into OEM Asset development, those in the bioeconomy will promote widespread remediation of its vulnerable Assets, protecting the drugs, foods, vaccines, crops and animals, and countless components critical to everyday life and wellbeing.

14 https://attack.mitre.org/techniques/ics/

APPENDIX I - About BIO-ISAC

The Bioeconomy Information Sharing and Analysis Center (BIO-ISAC), is the critical resource for addressing threats unique to the bioeconomy and enabling coordination among stakeholders to facilitate a robust and secure global industry. The BIO-ISAC is a 501(c)3 nonprofit member organization (pending), formally chartered in 2021.

BIO-ISAC provides a central resource for gathering information on threats to infrastructure impacting and forming the bioeconomy including the two-way sharing of information between and among public and private sectors in order to identify, protect, detect, respond, recover, and build resilience from attacks on public and private bioeconomy infrastructure. BIO-ISAC helps spur the development and evaluation of defensive tools to address these incidents and includes vulnerability identification and mitigation, as well as education, training, and outreach, aimed at reducing risk to the nation's biosecurity infrastructure.

APPENDIX II - Definitions

Asset(s). Network capable equipment or software that supports the manufacturing of biological products for human consumption, agricultural, and/or livestock, or affects other critical infrastructure as defined by the Department of Homeland Security, and any of the following:

a. Processes or stores PII, PHI, genomic information; or
b. Reasonably carries intellectual property; or
c. Connectivity to the public internet; or
d. Is connected to industrial control systems required for manufacturing processes; or
e. Carries, modified, transports, or stores the material that is being manufactured; or Generates the offline record, batch record information, or other regulatorily-required record

Asset Owner. The organization, person, or other responsible party managing things of value to an organization.

Biomanufacturing. As defined in the *Executive Order on Advancing Biotechnology and Biomanufacturing Innovation for a Sustainable, Safe, and Secure American Bioeconomy,* Biomanufacturing means the use of biological systems to develop assets, tools, and processes at commercial scale.

Client-Server Model. The client-server model, or client-server architecture, is a distributed application framework dividing tasks between servers and clients, which either reside in the same system or communicate through a computer network or the Internet.

Cloud Applications. A cloud application simply refers to any software application that is deployed in a cloud environment rather than being hosted on a local server or machine.

Cloud Service Provider (CSP). A cloud service provider is a third-party company offering a cloud-based platform, infrastructure, application, or storage services. Much like a homeowner would pay for a utility such as electricity or gas, companies typically have to pay only for the amount of cloud services they use, as business demands require.

Command Line Interface (CLI). A text-based interface that is used to operate software and operating systems while allowing the user to respond to visual prompts by typing single commands into the interface and receiving a reply in the same way.

Common Vulnerabilities and Exposures (CVEs). A dictionary of common names for publicly known information system vulnerabilities.

Defense-in-Depth. Information security strategy integrating people, technology, and operations capabilities to establish variable barriers across multiple layers and missions of the organization.

Graphical User Interface (GUI). An interface through which a user interacts with electronic devices such as computers and smartphones through the use of icons, menus and other visual indicators or representations (graphics). GUIs graphically display information and related user controls, unlike text-based interfaces, where data and commands are strictly in text. GUI representations are manipulated by a pointing device such as a mouse, trackball, stylus, or by a finger on a touch screen.

Hardware. Hardware refers to the external and internal devices and equipment that enable you to perform major functions such as input, output, storage, communication, processing, and more. There are two types of computer hardware: external and internal. External hardware devices include monitors, keyboards, printers, and scanners, whereas internal hardware devices include motherboards, hard drives, and RAM.

Insider Threat. The threat that an insider will use her/his authorized access, wittingly or unwittingly, to do harm to the security of organizational operations and assets, individuals, other organizations, and the Nation. This threat can include damage through espionage, terrorism, unauthorized disclosure of national security information, or through the loss or degradation of organizational resources or capabilities.

Malicious Actor. An entity that is partially or wholly responsible for a security incident that impacts – or has the potential to impact – an organization's security.

Multi-Factor Authentication (MFA). Authentication using two or more factors to achieve authentication. Factors include: (i) something you know (e.g. password/personal identification number (PIN)); (ii) something you have (e.g., cryptographic identification device, token); or (iii) something you are (e.g., biometric). See authenticator.

Multi-Tier Application. A Multi-tier application is an application which is split into multiple pieces. This commonly includes a database server that is separate from the application server and so on.

Original Equipment Manufacturer (OEM). Original equipment manufacturer, an organization that makes devices from component parts bought from other organizations.

Threat Analysis. Process of formally evaluating the degree of threat to an information system or enterprise and describing the nature of the threat.

Secret. Secrets are defined as any form of sensitive credentials that need to be tightly controlled and monitored and can be used to unlock sensitive information. Secrets could be in the form of passwords, API keys, SSH keys, RSA tokens, or OTP.

Secure Software Development Lifecycle (SSDLC). Secure SDLC (SSDLC) integrates security into the process, resulting in the security requirements being gathered alongside functional requirements, risk analysis being undertaken during the design phase, and security testing happening in parallel with development

Self-Hosted Applications. Self Hosting is a form of running your own website or application by setting up a server and network yourself. Instead of using a Platform as a Service or a Public Cloud Provider, those who choose to self-host will run their own networks and be responsible for the maintenance and uptime in addition to building their website or application.

Shared Responsibility Model. The Shared Responsibility Model is a security and compliance framework that outlines the responsibilities for securing every aspect of an environment, including hardware, infrastructure, endpoints, data, configurations, settings, operating system (OS), network controls and access rights.

Single Sign-On (SSO). The capability to authenticate once, and be subsequently and automatically authenticated when accessing various target systems. It eliminates the need to separately authenticate and sign on to individual applications and systems, essentially serving as a user surrogate between client workstations and target systems. Target applications and systems still maintain their own credential stores and present sign-on prompts to client devices. Behind the scenes, SSO responds to those prompts and maps the credentials to a single login/password pair. SSO is commonly deployed in enterprise, Web and federated models.

Single-Tier Application. A single-tier application is an application where the backend logic, database, and user interface lies in the same machine.

Software Bill of Materials (BOM). A formal record containing the details and supply chain relationships of various components used in building software. Software developers and vendors often create assets by assembling existing open source and commercial software components. The SBOM enumerates these components in an asset.

Software Supply Chain. The software supply chain is anything and everything that touches an application or plays a role, in any way, in its development throughout the entire software development life cycle (SDLC). Software supply chain security is the act of securing the components, activities, and practices involved in the creation and deployment of software. That includes third-party and proprietary code, deployment methods and infrastructure, interfaces and protocols, and developer practices and development tools.