Challenges and Opportunities for Securing Software Ecosystem Architectures

Walt Scacchi and Thomas A. Alspaugh, University of California, Irvine

We identify a set of challenges and opportunities for improving the security of software ecosystems and supply chain processes as software product components move through them. We focus attention to the role of explicit open architecture software ecosystem models spanning processes for software sourcing, integration, deployment, and evolution. These architectural models are relevant irrespective of the composition of organizational actors that delineate a software ecosystem. Consequently, every software ecosystem has one or more architectural models that can be visually mapped, communicated, and understood to improve supply chain process security.

Keywords: software ecosystems, software architecture, cybersecurity

Is everyone destined to become a software cybersecurity administrator? There is growing evidence suggesting that whenever someone interacts with a software ecosystem, they may be exposed to security threats they don't recognize, understand, or know how to mitigate. People will need to take action to mitigate the exposure of their computing platforms—smartphones, tablets, desktop computers, etc.—to cybersecurity challenges. How do different cybersecurity threats, vulnerabilities, and mitigations appear in software ecosystems? How can we model where these issues arise in the software ecosystems we rely on routinely?

Much of software system development arise from composing existing open-source, commercial, and internally developed software components into creative configurations for different platforms. The components originate from different software producers that are known, knowable, or unknown. Sometimes, component composition is fully transparent and open to analysis. More often, it is partially or fully closed from review, hiding which components are involved and how they are connected.

Software development and distribution are targeted to operate on specific kinds of platforms. System integrators may be situated between component producers and end-users/consumers to continuously integrate and release ready-to-install software packages. These packages appear in online app stores, open access repositories, or producer-managed download sites. Once downloaded and installed onto a target platform, the components may be further tailored with permitted local customizations or extensions, and configured through selection of appropriate, platform-specific (and sometimes user-specific) parameter settings.
The transfer of software components or app solutions from producers to end-users follows diverse software supply chain processes that progressively produce software packages. The overall composition of these supply chain processes forms a software ecosystem. These supply chains and intermediaries are sometimes known, but often are unknown or not be easily knowable. It may also be unclear what happens to software components or products as they progress through supply chain processes. This lack of transparency and uncertain provenance is an enabler of software cybersecurity attacks and exploitations. How can vulnerabilities in software supply chain processes and ecosystems be mitigated, and local enterprise or platform-specific cybersecurity be improved? We find that explicit models of open architecture software ecosystems (hereafter OA ecosystems), product lines, and product configurations offer promising insights.

Figure 1 presents a visual model of an OA ecosystem on the left, and one example of a security-encapsulated, installed software configuration realizing this architecture on the right. Other security encapsulations are possible. The architecture integrates components commonly found on desktop computing platforms—Web browser, word processor, email, calendar, and operating system—connected through data communication protocol handlers to network servers. When the OA ecosystem on the left is configured with software components from a single software vendor (e.g., Microsoft or Apple), then it designates a software product line architecture. But an OA ecosystem allows for many alternative installed configurations of components from different software producers, including that shown on the right in Figure 1. This OA ecosystem also accommodates alternative selections of the word processor and email/calendar apps, including ones hosted on remote servers and accessed and utilized from within a Web browser (e.g., Google Docs, Mail, and Calendar accessing shareable files via Apple iDrive using Firefox browser). Thus, visual models of OA ecosystems or software product line
architectures are informative in revealing what components are included, and how they are interconnected. It is also clear that no single software producer performs all actions within, or is responsible for, an OA ecosystem.

Software ecosystem researchers have sought to develop supply chain and network models to aid understanding, communication, and business strategy formation. Al Sabbagh and Kowalski, and the authors draw attention to socio-technical security threats that emerge during software supply chain processes. These processes include software sourcing, development and testing, packaging, software media manufacturing, and software delivery. Adapting their findings, one can foresee example software supply chain vulnerability scenarios such as: A software producer makes available a new component accompanied by a tamper-resistant software taggant, but the system integrator may not have a procedure for validating the software product origin using taggants. An integrator who is provided an unvalidated component for packaging and release may be unaware of this oversight, and releases the software with custom installation scripts using a different data encryption mechanism (e.g., an “installation wizard” or packer) for value-added product confidentiality. Last, the eventual consumer enterprise may have a weak mechanism for authenticating in-house end-users who are authorized to execute or update the resulting installed software configurations. They also may not provide recommended configuration settings to ensure secure and efficient software operation. None of these security vulnerabilities resides in software components or is specific to a software producer. Consequently, improving software development security does not address or resolve them. Instead, these vulnerabilities are propagated through socio-technical processes that articulate software ecosystem architectures.

Common models of software ecosystems do not utilize explicit software architectural representations. So software ecosystem models can obscure and hide potential supply chain security vulnerabilities, as suggested in the scenarios above. These vulnerabilities are recognized and exploited by software attackers. Accordingly, we seek a model providing practical insights to security vulnerabilities within software supply chain processes, as well as schemes for mitigating such vulnerabilities. Accordingly, we build from software supply chain processes together with an OA ecosystem model to identify process-centered relationships among participating software component producers, system integrators, and customers, along with preventive supply process security countermeasures. A sample of these relationships are identified in Table 1.

Using Explicit, Open Architectures Models to Improve Software Ecosystem Security

If you do a Web image search for “software ecosystem” the most common results are diagrams depicting partially ordered sets of related software producer brands or product names. The ecosystem is depicted as a network whose nodes indicate software producers, and whose links represent a relationship like “provides” or “uses.” Alternatively, the sets are represented by graphic boxes, organized hierarchically or tabularly. Such groupings are intended to suggest adjacent software producers/products are alternatives that fit within some ecosystem architecture, and thus may represent competing products. But where and how they fit into the ecosystem architecture is unclear or unknown. Thus how
software components and security threats move through ecosystems is unclear.

<table>
<thead>
<tr>
<th>Software Supply Chain Processes</th>
<th>Common ecosystem security problems for each process</th>
<th>Example threats to supply chain</th>
<th>Software supply chain security defenses</th>
<th>Further defenses enabled by explicit open architectures</th>
<th>Challenges in progressing to next software supply chain process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component sourcing</td>
<td>Untrusted or corrupt software producers</td>
<td>Counterfeit repositories for sourced components</td>
<td>Independent validation of components and interconnections</td>
<td>Validation of provenance in component supply chains</td>
<td>Independent validation of components sourced from unknown providers</td>
</tr>
<tr>
<td>Continuous integration and release</td>
<td>Infected or corrupt component producers</td>
<td>Counterfeit components in commercial products</td>
<td>Component provenance tracking and analysis</td>
<td>Recreation of multi-version builds to validate software product integrity</td>
<td>Collecting and passing on provenances, taggants, and other info for evaluation</td>
</tr>
<tr>
<td>Delivery and deployment</td>
<td>False flag download sites, bait and switch downloads</td>
<td>Counterfeit software with false certificates</td>
<td>Installation of components into security encapsulations</td>
<td>Maps of installed software configurations to guide defenses</td>
<td>Producing and delivering trustable, verifiable packages</td>
</tr>
<tr>
<td>Continuous evolution</td>
<td>Outdated component versions with known vulnerabilities</td>
<td>Update mechanism hijacked to enable remote control and data exfiltration</td>
<td>All of above</td>
<td>Repositories of above; deployment of multi-version releases</td>
<td>Maintaining security, trust, provenance, and other requirements as ecosystem and configurations evolve</td>
</tr>
</tbody>
</table>

**Table 1: Software supply chain security threats and defenses, organized by supply chain process**

Some software ecosystem researchers recognize the centrality of software architectures in understanding how ecosystem niches emerge around deployment platforms. Unfortunately, it is most common that open, accessible, and visual models of software architectures are either non-existent or not available. So end-users or enterprises have little understanding of which software components may be connected to others and how (e.g., via API, data communication protocol, or application scripts). Accordingly, on your smartphone or desktop computing platform with 20-50+ software applications installed, you cannot know what components interact with what others. We believe this can be remedied through more transparent and tractable architecture models of installed configurations and their software ecosystems.

Figure 1 displays both an OA ecosystem, and an installed configuration of components that details its architecture. What do these representations make transparent, and how do they make potential security threats more tractable?

First, explicit OA ecosystem maps can serve as reference models that characterize a software system domain. A reference model is an abstract framework or domain-specific ontology consisting of an interlinked set of software element types. The framework situates participating software components, interconnections, and connector types, and provides an overall software configuration layout map. Different producers may provide components or connectors of a specific type that fit into the model. Such models can be visually inspected and analyzed prior to software product integration. If anticipated products do not fit, then they are not integrated—for example, a payroll system is not functionally similar to a Web browser, while browsers from Apple, Google,
Microsoft, Mozilla, or Opera are. But how well they fit depends on other architectural component selections, interconnections, and their configuration.

Second, architectural maps can be abstract or detailed, revealing/hiding data flow and control signal pathways. These pathways reveal where and how security threats may be detected or blocked. The potential for a threat introduced in one component to propagate across their interconnections to directly connected components is greater in general, compared to components that are unconnected, or connected only through many intermediaries. While nation-state software security threats like Stuxnet\(^6\) and the Equifax attack\(^10\) successfully propagated across multiple components and configurations, propagation is easier to detect and prevent if interconnection pathways are transparent rather than hidden. The pathways denote entry/exit points where threats can appear and defensive security mechanisms should be deployed.

Third, OA ecosystem maps reveal where and how system configurations are potentially modified during each supply chain process. In Figure 1, the map on the left may inform the software sourcing selection process, while the configuration map on the right denotes a number of details added as a result of integration and deployment processes.\(^12,13\) These maps thus enable rapid analysis of software evolutionary changes, through visual means at least at an abstract architectural level. Even with such an abstract map, it is possible to identify potential systemic threats that can be mitigated through architecture-level defenses before evolutionary software updates are made.

Fourth, software architectures can be specified, visualized, automatically analyzed, and updated if a processable *architecture description language* is utilized.\(^2,11,14\) ADL development environments have been available within the software engineering research community for several decades.\(^14\) ADLs have been extended to associate intellectual property obligations and rights with software components configured into open architecture configuration specifications.\(^2,11\) Operational security constraints like capability lists, access control lists, and virtual machine encapsulations can be similarly specified as access obligations and usage rights on component interfaces.\(^13\) The key benefit here is that formalization of ADL-based software ecosystem and configuration architectures offers the potential to automate analysis of the consistency, completeness, traceability, and internal correctness of a configuration. This analysis can be reconciled against the evolving provenance of the open architecture specifications derived from an OA ecosystem model.\(^11\)

Last, Figure 1 provides an abstract view of a configuration map for common desktop computing environments entailing millions of lines of code. Connectors are grouped into types, and interconnections are abstracted to hide many possible interconnections between components. Such simplified maps may be understood by managers or end-users who do not need the challenge of understanding deep, fully articulated software configuration representations. Deep architectural analysis requires software elements and their configurational dependencies to be automatically extracted, inventoried, and versioned. Their respective supply chain provenances must be serialized, tracked, and automatically validated across repositories controlled respectively by software producers, integrators, and customers. Subsequently, when deep security analysis is required, it requires open access to all software code utilized during integration, build, release,
delivery, and deployment, and to the automated mechanisms used to update or evolve software configurations. Otherwise, without extraordinary effort, supply chain process vulnerabilities will persist and go undetected.

**Conclusions**

There are many challenges for improving the cybersecurity of software ecosystems and supply chains. The multiplicity of producers, system integrators, and consumers implies no one actor has overall responsibility, nor the ability to readily utilize the complete range of cybersecurity countermeasures now available. Cybersecurity best practices are still partial, and often apply only to a particular software process. We identified how visual models of OA ecosystems can reveal different ecosystem architecture risks and defenses spanning processes for software sourcing, integration, deployment, and evolution. The challenges still need innovative interventions for automated modeling and analysis capabilities.

Many opportunities to improve software ecosystem security can be identified and explored to address the problems, defenses, and challenges outlined in Table 1. These opportunities are intended to seed discussions of software ecosystem cybersecurity, and expect more can be articulated and pursued. Ultimately, these approaches can produce innovations to mitigate the threats and vulnerabilities that emerge as software components move through software supply chain processes.

Mapping architectures of software ecosystems and configurations that articulate supply chain processes offers immediate benefits. Visual models may be more acceptable and more easily understood and analyzed by managers and end-users. But for complex architectures, maintaining detailed architectural maps manually can be difficult. Architecture models should ideally be specified and derivable via architecture description languages. Consequently, we call for effort giving rise to automated production and update of open architecture representations that visually map ecosystems and configuration components, their interconnections, and their connector types based on an interpretable open specifications.

**Acknowledgments**

This research was supported by grants #N00244-16-1-0004 and #N00244-16-1-0053 from the Acquisition Research Program at the Naval Postgraduate School, Monterey, CA.

**References**


Walt Scacchi is senior research scientist at the Institute for Software Research, University of California, Irvine. Contact him at wscacchi@ics.uci.edu

Thomas A. Alspaugh is a project scientist at the Institute for Software Research, University of California, Irvine. Contact him at thomas.alspaugh@acm.org

3-5 Tweets:

- New paper examines how cybersecurity threats are propagated across supply chains supporting software ecosystems. (113 chars)

- Scacchi and Alspaugh examine challenges and opportunities for improving the security of software ecosystems and supply chain processes as software product components move through them. (185 chars)

- Scacchi and Alspaugh examine role of explicit open architecture software ecosystem models spanning processes for software sourcing, integration, deployment, and evolution. (172 chars)

- Scacchi and Alspaugh describe how software ecosystems can be visually mapped, communicated, and understood using explicit architectural models to improve supply chain process security. (185 chars)