Securing Software Ecosystem Architectures

Challenges and Opportunities

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

// We identify the challenges and opportunities for improving the security of software ecosystems and supply chain processes. Every software ecosystem has one or more architectural models that can be visually mapped, communicated, and understood to improve supply chain process security. //

IS EVERYONE DESTINED to become a software cybersecurity administrator? There is growing evidence suggesting that whenever individuals interact with a software ecosystem, they may be exposed to security threats they do not recognize, understand, or know how to mitigate. People need to take action to mitigate the exposure of their computing platforms—smartphones, tablets, desktop computers, and so on—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 that we rely on routinely?

Digital Object Identifier 10.1109/MS.2018.2874574
Date of publication: 16 April 2019

Much software system development arises from composing existing open source, commercial, and internally developed software components into creative configurations for different platforms.[3] 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 the software is downloaded and installed onto a target platform, the components may be further tailored with permitted local customizations or extensions, and they can be configured through the 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.[3] These supply chains and intermediaries are sometimes known,

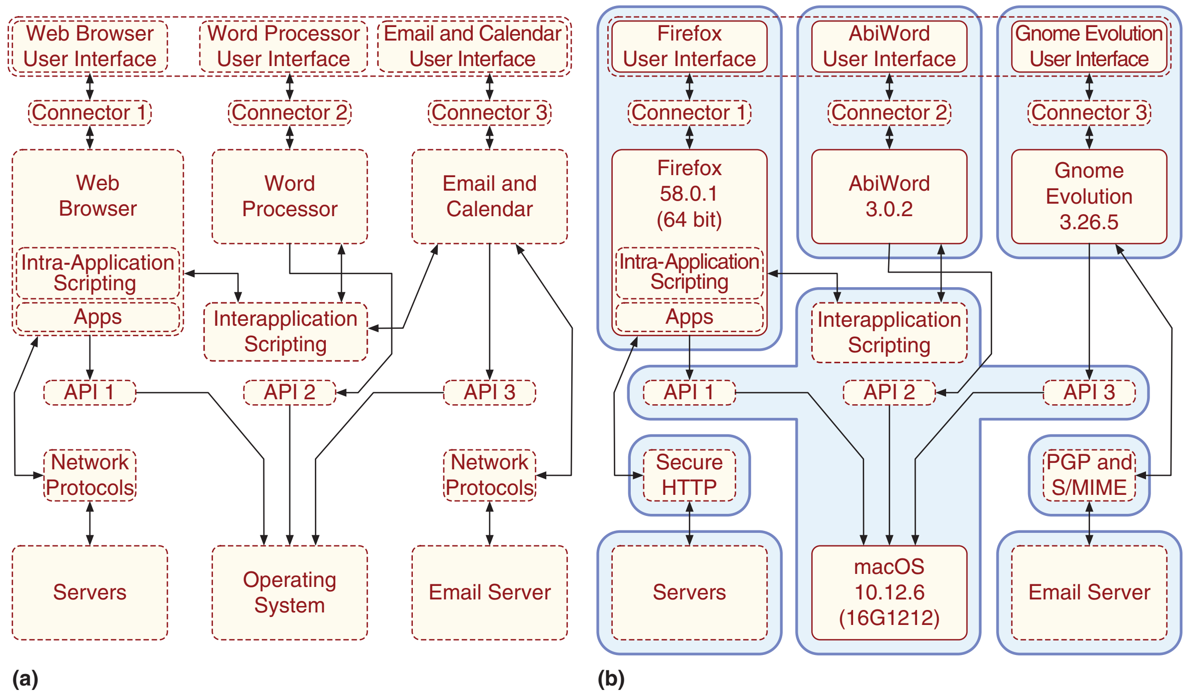

FIGURE 1. A visual model of (a) an OA ecosystem and (b) an example of a corresponding security-encapsulated, installed software configuration architecture. Different types of connectors are shown, including application program interfaces (API), network protocols like Pretty Good Privacy (PGP), and secure multipurpose Internet mail extensions (S/MIME).

but often they are unknown or not easily knowable. It may also be unclear what happens to software components or products as they progress through supply chain processes.[12] This lack of transparency, combined with uncertain provenance, enables software cybersecurity attacks and exploitations. How can vulnerabilities in software supply chain processes and ecosystems be mitigated, and how can local enterprise- or platform-specific cybersecurity be improved? We find that explicit models of open architecture (OA) software ecosystems[2,11] product lines, and product configurations offer promising insights.

Figure 1(a) presents a visual model of an OA ecosystem, and one example of a security-encapsulated, installed software configuration realizing this architecture in (b). Other security encapsulations are possible.[13] This 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 in Figure 1(a) is configured with software components from a single software vendor (e.g., Microsoft or Apple), then it designates a software product line architecture.[3] However, an OA ecosystem allows for many alternative installed configurations of components from different software producers, including that shown in Figure 1(b). 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, email, and a calendar accessing shareable files via Apple iDrive using the Firefox browser). Thus, visual models of OA ecosystems or software product line architectures can reveal 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

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

Software supply chain processes	Common ecosystem security problems for each process	Example threats to the supply chain[1]	Software supply chain security defenses[1,12,13]	Further defenses enabled by explicit OA[1,2]	Challenges in progressing to the next software supply chain process[1,12,13]
Component sourcing	Untrusted or corrupt software producers	Counterfeit repositories for sourced components[7]	Independent validation of components and interconnections	Validation of provenance in component supply chains	Independent validation of components sourced from unknown providers
Continuous integration and release[1,2]	Infected or corrupt component producers	Counterfeit components in commercial products[7]	Component provenance tracking and analysis	Re-creation of multiversion builds to validate software product integrity	Collecting and passing on provenances, taggants, and other information for evaluation
Delivery and deployment[1,2]	False flag download sites, bait-and-switch downloads	Counterfeit software with false certificates[6]	Installation of components into security encapsulations	Maps of installed software configurations to guide defenses	Producing and delivering trustable, verifiable packages
Continuous evolution[11,12]	Outdated component versions with known vulnerabilities	Update mechanism hijacked to enable remote control and data exfiltration[5]	All of the defenses listed	Repositories listed, deployment of multiversion releases	Maintaining security, trust, provenance, and other requirements as ecosystem and configurations evolve

formation.[8] Al Sabbagh and Kowalski[1] and the authors[12,13] draw attention to sociotechnical security threats that emerge during software supply chain processes. These processes include software sourcing, development and testing, packaging, software media manufacturing, and software delivery. By adapting their findings, one can foresee example software supply chain vulnerability scenarios. For instance, a software producer makes available a new component accompanied by a tamper-resistant software taggant,[9] but the system integrator may not have a procedure for validating the software product origin using taggants. An integrator who is given an unvalidated component for packaging and release may be unaware of this oversight,[7] and might release the software with custom installation scripts using a different data encryption mechanism (e.g., an installation wizard or packer) for value-added product confidentiality. Finally, 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 reside in software components or are specific to a software producer. Consequently, improving software development security does not address or resolve them. Instead, these vulnerabilities are propagated through sociotechnical processes that articulate software ecosystem architectures.

> How do different cybersecurity threats, vulnerabilities, and mitigations appear in software ecosystems?

Common models of software ecosystems do not utilize explicit software architectural representations. Therefore, software ecosystem models can obscure and hide potential supply chain security vulnerabilities, as suggested in the scenarios described. These vulnerabilities can be recognized and exploited by software attackers. Accordingly, we sought a model that provides practical insights into security vulnerabilities within software supply chain processes as well as schemes for mitigating such vulnerabilities. Accordingly, we used software supply chain processes[12,13] and an OA ecosystem model[2,11] to identify process-centered relationships among participating software component producers, system integrators, and customers, along with preventive supply process security countermeasures. Samples of these relationships are identified in Table 1.

Using Explicit OA Models to Improve Software Ecosystem Security

The most common results of a web image search for "software ecosystem" are diagrams depicting partially ordered sets of related software producer brands or product names. The ecosystem is often depicted as a network whose nodes indicate software producers and whose links represent a relationship such as "provides" or "uses." Alternatively, the sets are represented by graphic boxes, organized hierarchically or tabularly. Such groupings suggest that 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.

Some software ecosystem researchers recognize the centrality of software architectures in understanding how ecosystem niches emerge around deployment platforms.[2,3,11] Unfortunately, it is most common that open, accessible, and visual models of software architectures are either nonexistent or not available. Therefore, end users or enterprises have little understanding of which software components may be connected to others and how (e.g., via the application programming interface, data communication protocol, or application scripts). Accordingly, on your smartphone or desktop computing platform with 20–50+ software applications installed, you cannot easily know which components interact with one another. We believe this can be remedied through more transparent and tractable architecture models of installed configurations and their software ecosystems.

> This lack of transparency, combined with uncertain provenance, enables software cybersecurity attacks and exploitations.

Figure 1 shows 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.[12,14] 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 it 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 before software product integration. If anticipated products do not fit, then they are not integrated—e.g., a payroll system is not functionally similar to a web browser, whereas browsers from Apple, Google, Microsoft, Mozilla, or Opera are. How well they fit, however, depends on other architectural component selections, interconnections, and configuration.

Second, architectural maps can be abstract or detailed, hiding or revealing data flow and control signal pathways. These pathways show 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 generally greater when compared with components that are unconnected or connected only through many intermediaries. Although nation-state software security threats such as 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 where 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(a), the map may inform the software sourcing selection process, and the configuration map in (b) 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] Architecture Description Language (ADL) development environments have been available in 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 OA configuration specifications.[2,11] Operational security constraints such as 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 OA specifications derived from an OA ecosystem model.[11]

Finally, 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 demands 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.

There are many challenges for improving the cybersecurity of software ecosystems and supply chains. The multiplicity of producers, system integrators, and consumers implies that no one actor has overall responsibility or the ability to readily utilize the complete range of cybersecurity countermeasures now available. Cybersecurity best practices are still partial, and they 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

ABOUT THE AUTHORS

WALT SCACCHI is a senior research scientist and research faculty emeritus at the University of California, Irvine, Institute for Software Research. Scacchi received a Ph.D. in information and computer science from the University of California, Irvine. He has received more than 60 research grants and given more than 200 invited lectures, 17 keynote addresses, and seven invited tutorials at international conferences. He serves on the advisory board of *Journal of Software: Evolution and Process*. He was a Member of the IEEE and ACM for 35 years. Contact him at wscacchi@ics.uci.edu.

THOMAS A. ALSPAUGH is a former faculty member and research scientist at the University of California, Irvine, Institute for Software Research. Alspaugh received a Ph.D. in computer science from North Carolina State University. He was a long-time Member of the IEEE, ACM, and SIGSOFT. Contact him at alspaugh@ics.uci.edu.

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 we expect that more opportunities 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. For complex architectures, however, maintaining detailed architectural maps manually can be difficult. Architecture models should ideally be specified and derivable via architecture description languages. Consequently, we call for efforts giving rise to automated production and update of OA representations that visually map ecosystems and configuration components, their interconnections, and their connector types based on 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, California.

References

1. B. Al Sabbagh and S. Kowalski, "A socio-technical framework for threat modeling a software supply chain," *IEEE Security Privacy*, vol. 13, no. 4, pp. 30–39, 2015. doi: 10.1109/MSP.2015.72.
2. T. A. Alspaugh, W. Scacchi, and H. A. Asuncion, "The challenge of heterogeneously licensed systems in open architecture software ecosystems," in *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*, S. Jansen, S. Brinkkemper, and M. Cusumano, Eds. Northampton, MA: Edward Elgar Publishing, 2013, pp. 103–120.
3. J. Bosch, "From software product lines to software ecosystems," in *Proc. 13th Int. Software Product Line Conf. (SPLC 2009)*, San Francisco, CA, 2009, pp. 111–119.
4. E. Brumaghin, R. Gibb, W. Mercer, M. Molyett, and C. Williams, "CCleanup: a vast number of machines at risk," Cisco TALOS Blog, Sept. 18, 2017. Accessed on: Jan. 2018. [Online.] Available: https://blogs.cisco.com/security/talos/ccleanup-a-vast-number-of-machines-at-risk
5. A. Cherepanov, "Analysis of Telebots' cunning backdoor," July 4, 2017. Accessed on: Jan. 2018. [Online.] Available: https://www.welivesecurity.com/2017/07/04/analysis-of-telebots-cunning-backdoor/ *WeLiveSecurity.com.*
6. N. Falliere, L. Murchu, and E. Chien, "W32.Stuxnet dossier," Symantec Inc., Mountain View, CA, Feb. 2011. Accessed on: Jan. 2018. [Online.] Available: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf
7. A. Greenberg, "Software has a serious supply-chain security problem," *Wired*, Sept. 18, 2017. Accessed on: Jan. 2018. [Online]. Available: https://www.wired.com/story/ccleaner-malware-supply-chain-software-security/
8. S. Jansen, S. Brinkkemper, and M. Cusumano (Eds.), *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*, Northampton, MA: Edward Elgar Publishing, 2013.
9. M. Kennedy and I. Muttik, "IEEE Taggant System," IEEE Standards Association, 2011. Accessed on: Jan. 2018. [Online]. Available: https://media.blackhat.com/bh-us-11/Kennedy/BH_US_11_KennedyMuttik_IEEE_Slides.pdf
10. M. Riley, J. Robertson, and A. Sharpe, "The Equifax hack has the hallmarks of state-sponsored pros," *Bloomberg Businessweek*, Sept. 29, 2017. [Online]. Available: https://www.bloomberg.com/news/features/2017-09-29/the-equifax-hack-has-all-the-hallmarks-of-state-sponsored-pros
11. W. Scacchi and T. A. Alspaugh, "Understanding the role of licenses and evolution in open architecture software ecosystems," *J. Syst. Softw.*, vol. 85, no. 7, pp. 1479–1494, 2012. doi: 10.1016/j.jss.2012.03.033.
12. W. Scacchi and T. A. Alspaugh, "Processes in securing open architecture software systems," *Proc. 2013 Intern. Conf. Software and System Processes*, San Francisco, CA, 2013, pp. 126–135.
13. W. Scacchi and T. A. Alspaugh, "Advances in the acquisition of secure systems based on open architectures," *J. Cybersecurity Inform. Syst.*, vol. 1, no. 2, pp. 2–16, 2013.
14. R. N. Taylor, N. Medvidovic, and E. M. Dashofy, *Software Architecture: Foundations, Theory, and Practice*, Hoboken, NJ: Wiley, 2009.