Software Transparency in SaaS Environments

Executive Summary

Producers of software as a service (SaaS) should maintain Software Bill of Materials (SBOM) data about their software, SaaS operators should request that data from the SaaS producers, and those considering subscribing to SaaS should request SBOMs from the operators. The benefits of transparency for software are well documented. Software transparency allows the choosers and operators of software to make informed decisions and supports software Operators in managing risks and responding to vulnerabilities. SBOM has emerged as a key tool for communicating data about the components in packaged software. Through an SBOM, a software Producer can communicate information about the components and dependencies of a software instance in a machine-readable procedure. Furthermore, by generating and maintaining an SBOM, Producers signal to Operators, Choosers, and Subscribers the maturity of their software security practices.

While there has been extensive work produced on the benefits of software transparency in packaged software, there remains ambiguity regarding how software transparency can be achieved for SaaS. This community-drafted white paper[1] argues for the value of SBOM-driven transparency, while identifying four key differences: (1) the frequency at which SaaS is updated; (2) the volume of software and services intertwined with SaaS; (3) the lack of definitive boundaries that determine the horizontal and vertical extent of software composition data; and (4) the opacity of SaaS systems. Using four roles (Producer, Chooser, Operator, and Subscriber), this paper discusses who could benefit from SBOM in SaaS environments and lays out a path to address current challenges.

The complexity of services offered through SaaS is one of the factors that limit established software transparency approaches like SBOM. In acknowledgment of the necessity for transparency in SaaS services, in addition to software components, this paper proposes preliminary data fields SaaS software Producers should communicate to software Choosers and Operators.

Introduction

In an era marked by growing cybersecurity concerns, conversations around software security have increasingly centered around transparency. Transparency is not a goal in and of itself but

[1] This document was drafted by the SBOM Cloud and Online ApplicationsWorking Group, a community-driven workstream. For more information see About this document. https://www.cisa.gov/resources-tools/resources/sbom-community-legal-explanation

rather a means to several ends. The benefits include identifying vulnerable components or libraries, risk factors, fortifying the software ecosystem against potential threats, and extending beyond security to improve efficiencies and better-aligning incentives. Much of these discussions are focused on customer-managed software, but many of these also apply to the Software as a Service (SaaS) ecosystem.

While many of the core concepts of Software Bill of Materials (SBOM) can be applied to SaaS, there are some characteristics that complicate the direct mapping. However, SaaS is software, and SBOMs generally provide insights into the software itself. This paper concludes that, while there are limiting factors in the efficacy of SBOMs for communicating software component data for SaaS, SBOM concepts provide a valuable jumping-off point toward transparency for SaaS.

Building on prior work[2] regarding SBOM and its role in software and supply chain transparency, this community-drafted white paper examines how to map the concepts and mechanics of SBOM to modern online applications and also how to address the opacity of SaaS instances.[3] In light of the complexity of SaaS software components and services, this paper discusses software component transparency[4] and software service[5] transparency,[6] two parts of the larger software transparency.

SBOM is one type of software component inventory. Though others exist, SBOM has emerged as a key tool for software component transparency and has become the leading method for documenting and communicating the components that comprise a software instance. This paper also proposes an inventory for an online service from the perspective of a stakeholder using the service's exposed endpoints, such as a service API, and the consumer data that the service will be ingesting, transporting, and/or storing.

Scope

This document provides background and a conceptual grounding in the significance of software component transparency and software service transparency in the context of SaaS. While acknowledging that SBOMs for SaaS are not yet mature or well-defined, discussions touch on both the current state of SBOM and SaaS and potential future data fields for SBOM that address some transparency concerns unique to SaaS. The recommendations proposed in this paper are intended to be a starting point for SaaS Choosers, Operators, and Subscribers to request transparency for SaaS components and services from Producers. Direct mapping to specific kinds of SaaS can be explored in later work.

[2] Resources from the NTIA and CISA SBOM working groups can be found at cisa.gov/sbom

[3] A software component participating in a service-oriented architecture that provides functionality or participates in realizing one or more capabilities. NIST. Glossary: Service. https://csrc.nist.gov/glossary/term/service

[4] Software component transparency supports understanding the composition of software.

[6]Service transparency supports understanding the general function of an online service.

This document should not be viewed as a technical specification of SBOMs for SaaS. Nor will this document provide normative guidance regarding legal, procurement, regulatory, or policy requirements for SBOM and software service transparency, or touch on infrastructure as a service (IaaS).

Definitions

To clarify discussions on SaaS transparency, the following definitions for frequently used terms are used. The definitions are grounded in existing work, and tailored for the SaaS context.

SaaS

Some definitions of SaaS center on the locus of deployment, contrasting "SaaS" or "Cloud" software with "on-premises" software. The National Institute of Standards and Technology's (NIST) 2011-era "The NIST Definition of Cloud Computing"[7] takes this approach, although notably in the cloud-native context of a narrow contrast between "SaaS," "PaaS" (platform as a service), and "IaaS."

Instead, this document highlights the key distinction that **SaaS is not customer-managed; it is provider-managed**. With this framing in mind, it is worth noting that SaaS software can be deployed on-premises (e.g., a managed appliance mounted in an office rack), and non-SaaS can be deployed in the cloud (e.g., a customer-managed VM or cloud project).

[7] NIST. Special Publication 800-145: The NIST Definition of Cloud Computing. September 2011. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

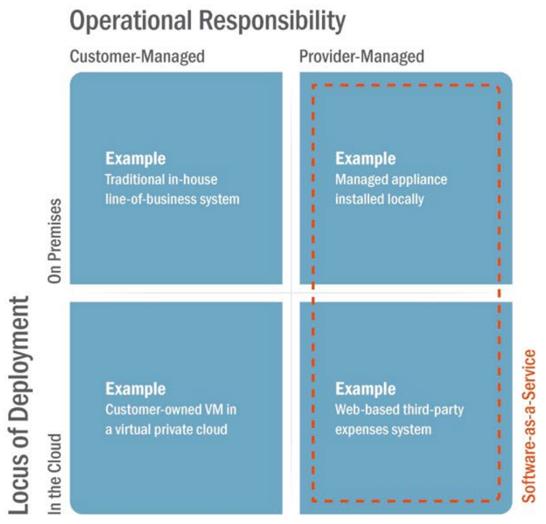
Operational Responsibility
Customer-Managed
Provider-Managed
Locus of Deployment
On Premises
In the Cloud
Example
Traditional in-house line-of-business system
Example
Managed appliance installed locally
Example
Customer-owned VM in a virtual private cloud
Example
Web-based third-party expenses system
Software-as-a-Service

Figure 1 —The defining characteristic of SaaS is that it is provider-managed.

For the purposes at hand, this paper uses SaaS to mean **software whose deployment, support, management, maintenance, and entire lifecycle are controlled by a supplier external to the consuming organization**; typically, such software is hosted in the cloud by the service- supplier. This stands in contrast with non-SaaS software which is delivered as executable bits and bytes to a customer for the customer to deploy and operate, either in the cloud or on-premises.

Roles

The following roles represent the blending of unique and complex stakeholder identities found throughout the software lifecycle. A stakeholder may take on one or more of these roles at any given time. For example, during the renewal period, a Subscriber may transition to a Chooser as they reevaluate continuing with the SaaS. In addition, an organization can be both a Producer and an Operator, as they develop their own software (Producer) and operate software they have purchased (Operator).

The roles of Producer, Chooser, and Operator are grounded in the National Telecommunications and Information Administration (NTIA) Document “Roles and Benefits for

SBOM Across the Supply Chain"[8] and adapted for a SaaS context. The Subscriber role is new to the software transparency conversation.

Producer: A Producer is the creator of the software. The Producer holds information regarding the software that Choosers, Operators, and Subscribers may be interested in.

Chooser: A Chooser is looking for software and/or services that meet their organization's needs (e.g., development, acquisition, procurement). Choosers are interested in information regarding the software and services to determine if it is the best choice for them. For some subscription models, organizations may revisit the Chooser role on an annual or other regular cycle.

Operator: Operators are currently managing a system using the SaaS in question. They are looking for up-to-date information—to which they will subsequently respond—regarding the software that would affect system performance or security. For SaaS, the Operator role often includes the application service provider that is maintaining the software, as well as the enterprise customer that is using the software. Operators are sometimes, but not always, the Producer of the software. Operators must communicate meaningful changes in the SaaS to Subscribers and mitigate their risk.

Subscriber: Having chosen the Producer-drafted software, provisioned by the Operator, the using organization or user understands the inherent risks of the SaaS and must monitor for any changes in risk. For packaged software, the Operator and the user are often synonymous. For SaaS, the user or organization that uses the SaaS has a different set of activities. The Subscriber receives information on the SaaS from the Operator.

Comparing SaaS and Non-SaaS

SaaS is often distinguished from non-SaaS software in discussions around software transparency. While there are some unique aspects to SaaS as defined above, both are software with dependencies.

Shared Responsibility Model

Discussions around the security of cloud applications may build upon the idea of a "shared responsibility model."[9] This model spreads responsibility for risk management among cloud

[8] NTIA Open Working Group on Use Cases and State of Practices. Roles and Benefits for SBOM Across the Supply Chain. November 8, 2019. https://www.ntia.gov/sites/default/files/publications/ntia_sbom_use_cases_roles_benefits-nov2019_0.pdf
[9]NIST. SP 500-291: NIST Cloud Computing Standards Roadmap. August 10, 2011. https://www.nist.gov/publications/nist-sp-500-291-nist-cloud-computing-standards-roadmap; NIST. SP 500-322: Evaluation of Cloud Computing Services Based on NIST SP 800-145. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-322.pdf; UK NCSC. Cloud Security Guidance. https://www.ncsc.gov.uk/collection/cloud/understanding-cloud-services/cloud-security-shared-responsibility-model&sa=D&source=docs&ust=1697652518825890&usg=AOvVaw3u4MitjxfGzicexAUdjp3U; CISA.

service providers and cloud service users. This document will apply a similar taxonomy for software Producers, Operators, Choosers, and Subscribers. While software writers focus on code quality, software Operators share operational and risk management responsibilities with SaaS Producers; the distribution of responsibility is dependent on the makeup of the service and its use. This includes understanding the specific risks that concern the Operator, such as third-party risk management. Choosers are responsible for ensuring compliance to standards and best practices, which can include the use of SBOMs as well as additional software and service transparency tools.

Security is seldom a single party's responsibility: it is more often a shared responsibility among multiple parties. The nature of most SaaS products is to hide the Producer's internal solution from the Chooser and Subscriber; Operators know and are able to leverage the functional capabilities of the SaaS but not its internal workings – which increases the security responsibilities of the Producer. This generally results in the Operator operating SaaS products and services that are well-defined but opaque to Choosers and Subscribers. Ideally, the Producer would rapidly remediate all known security issues and be able attest to their actions as part of their compliance responsibilities. The Operator can then choose to trust the Producer on the basis of their regulatory compliance attestations.

What the Shared Responsibility Model Means for SBOM

SBOMs inventory the software componentry for a software product. When the control of a non-SaaS software product is transferred from the Producer to the Operator, the Operator can store its corresponding SBOM in their SBOM management system. Whenever a new version of the software product is issued, the SBOM for the product must be updated in the SBOM management tooling.

Value of SBOMs in the SaaS Context

Both SaaS and non-SaaS software can be large, complicated, and likely to include third-party components, including open source software (OSS). Consequently, both suffer from risks associated with complex software supply chains, including but not limited to the inclusion of security vulnerabilities. For both SaaS and non-SaaS software, Choosers and Operators have a natural interest in understanding the composition of the software in use in order to better select and maintain the security of the environment.

Cloud Security Technical Reference Architecture. June 2022. https://www.cisa.gov/resources-tools/resources/cloud-security-technical-reference-architecture; Amazon. Shared Responsibility Model. https://aws.amazon.com/compliance/shared-responsibility-model/; Google. Shared Responsibilities and Shared Fate on Google Cloud. August 21, 2023. https://cloud.google.com/architecture/framework/security/shared-responsibility-shared-fate; Microsoft. Shared Responsibility in the Cloud. September 29, 2023. https://learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility; IEEE SA. Standard for Cloud Computing Shared Function Model. September 23, 2021. https://standards.ieee.org/ieee/2304/10690/.

The use cases in the Appendix make it clear that there are concrete benefits to understanding software composition that apply to much of the SaaS ecosystem as to that of packaged software. Indeed, for the Producer of the software and the Chooser, the benefits of SBOMs are largely the same for SaaS as for packaged software. In addition, in the case of SaaS Operators, SBOMs can help with defense-in-depth efforts by increasing the visibility into parts of, or the whole, system without introspecting the system itself (e.g., to determine the risk of an attack or the impact of a vulnerability).

Using SBOMs at this scale requires the coordination of disparate teams and a high level of automation. At this time, there are no defined specifications, processes, or tools that perform this task holistically; nor, as noted above, are there norms or standards governing the meaning of SBOM comprehensiveness in a SaaS context.

Limitations of SBOM in the SaaS Context

For packaged software, an SBOM provides Choosers and Operators with greater transparency into its composition. A common example of the use of an SBOM is in the case of a critical vulnerability,[14] where a software Operator can analyze the SBOM to determine whether a solution includes an affected version of a particular component. It should be noted that Producer SBOMs are likely to be large and complex, and a manual process does not scale.

In contrast, four key aspects necessitate additional considerations for the applicability of SBOM for SaaS. First, SaaS systems are frequently changing, and the change is outside the control of, and often not visible to, the Chooser and Subscriber. Operators may deploy continuously, potentially pushing code multiple times a day with small engineering changes made each time.[10] The frequency of change raises the question, for all actors, of how to keep track of the composition of the software.

Second, SaaS systems are often part of a larger ecosystem of associated services and tools, so they have more options available when applying security mitigations. For example, a SaaS system may resolve a remote execution issue in a component by stopping all inbound traffic including the malicious payload at the perimeter. This form of mitigation is likely to be much faster to deploy than component updates and, as a change to deployment configuration, does not affect the actual software composition.

Furthermore, more than one party may be involved in delivering functionality to the Subscriber. For example, an Operator may use a third-party identity provider to provide authentication to a service, or use a third-party storage provider to store customer data. The net result for the Subscriber is a service with little to no transparency as to the provenance or the maintenance of the network of services that are in use. There is no consistent machine-readable way to collate

[10] Not all updates will change the software composition, but the software composition, and therefore the SBOM, is still changing at a higher rate than non-SaaS. Furthermore, while the software composition may not change, the data (e.g. version) in the SBOM may need to be updated.

all the data required to either make decisions on choosing or troubleshooting systems that are in operation.

Third, the technical boundaries of a SaaS product are not as intrinsically defined as that of packaged software. No widely accepted definition exists as to the appropriate horizontal or vertical extent of software composition data for SaaS. Because of this, individual component inventory implementations may vary, and understanding vendor-specific nuances becomes an important and unfortunate part of reasoning about SaaS composition. Machine-readability of a data format is little help without a shared understanding of semantics.

Lastly, SaaS systems are often, by their nature, opaque as to their implementation. A packaged solution may be directly analyzed by an Operator to determine its composition. A SaaS system does not typically offer this affordance. While these factors make the transparency challenging, they do not make it intractable.

Recommendations

Recognizing the limited transparency measures for SaaS software, the recommendations below are offered as initial steps towards Choosers, Operators, and Subscribers gaining visibility into the software components and services that comprise a Producer's SaaS product.

Software Component Transparency

Choosers: **Request an SBOM from the Operator**

Operators: **Request an SBOM from the Producer**

SBOMs offer SaaS Choosers and Operators some insight into the components within the SaaS product. Component transparency allows Choosers to evaluate the included components as part of the purchase decision for the SaaS product and would enable Operators to make better-educated decisions when managing vulnerabilities, risks, and cybersecurity incidents.

As SBOM consumers, Choosers and Operators should request details regarding how frequently the SBOM is updated or what actions trigger a new SBOM. For example, Producers may only be updating the SBOM when components change (e.g., new components or updated components) and not updating the SBOM with each production change.

In addition, Choosers and Operators should make sure that they have a clear understanding of the scope of any SBOM for SaaS, keeping in mind the blurred lines bounding the extent of SaaS software, which is currently an underspecified area. Differences in approach between SBOM suppliers will in general frustrate attempts to compare SBOMs across SaaS products, limit automated aggregation of SBOMs, and circumscribe conclusions which may be drawn from the presence or absence of a given component.

As transparency measures for SaaS software mature and best practices emerge, Choosers and Operators will be better equipped to request specific information in SBOMs that will provide them with the insights necessary to manage risk across their SaaS software.

Software Service Transparency

Organizations Responsible for Standards and Guidance: **Specify a software service transparency interchange format.**

Standards organizations across the software ecosystem should create a standard way for Producers, Operators, and Choosers to consolidate, exchange, and potentially automate processing metadata about a service's configuration and deployment.

This paper scopes SBOMs to provide rich machine-readable data inventorying the component parts of a software product. Security and compliance relevant macro-level properties of the service as deployed are not a part of SBOM specifications today, but they are nonetheless important from the perspective of those accountable for risk management when choosing, operating, and using SaaS software.

The following sections are intended to highlight the current gaps in software service transparency information. The proposed data fields would benefit from additional exploration, analysis, and detailed specification. The data fields are intended to be illustrative in nature, and are not intended to be exhaustive, normative, fully specified, or scoped in detail. Notably, however, in the absence of a formal specification, it may be useful to collect this data from actors upstream in the supply chain, even in an ad-hoc manual way.

Proposed Data Fields

The data fields[11] below are offered as usefully suggestive of information that Operators, Choosers, and Subscribers may find valuable regarding SaaS software Refer to the Appendix for detailed treatment of use cases and scenarios that may help with understanding the SaaS landscape.

Service Functions

Service functions are the types of functions the service provides. Examples include identity, authentication, certificate authority, CNA, load balancing, etc. This data field benefits the Chooser and Subscriber, and the data can be applied to infrastructure governance and regulatory compliance.

Service Location

Service location is the geographical location where the service is hosted. Cloud Providers list these as us-east, brazil-south, etc. Multiple locations may be listed here. This data field benefits the Chooser and Operator, and the data can be applied to data governance and regulatory compliance.

[11]See Table 1 Illustrative Software Service Transparency Data Fields in Appendix

Service Protocol

Service protocol is the communication protocol used by service endpoints (e.g., http, https, mqtt). This data field benefits the Chooser and Operator, and the data can be applied to infrastructure governance and regulatory compliance.

Service Agreement

Service agreement is the text from, or link to, the terms of service agreed to by the consumer of the service. Services use terms of service agreements to settle the conditions under which users can access the service, distribute and operate with the data received from the service, as well as the conditions under which the user can store data and the service can operate with that data. Additionally, it may include code copyright along with installation and distribution conditions that fall under the end-user license agreement and are not subject to this advisory. This data field benefits the Chooser, Subscriber, and Operator, and the data can be applied to service availability.

Service Status

Service status is a link to the status page showing service uptime information. This data field benefits the Subscriber, and the data can be applied to service availability.

Data Flow

Data flow is either selected as “unidirectional” or “bi-directional.” This data field benefits the Operator, and the data can be applied to data governance.

Data Classifications

Data classification refers to the sensitivity of the data being processed by the service (as opposed to classification in machine learning). Examples of data classification include PII, PHI, confidential, and public. Each organization has a different way of classifying data based on government regulations and internal policies. This field is left generic to accommodate the different types of data classifications that may exist and future regulations that will arise. There is currently no canonical standard for classifying data,[12] but in general, levels of sensitivity are indicated by some string which can be used in this field. This data field benefits the Chooser and Operator, and the data can be applied to data governance.

Conclusion

The frequency with which SaaS products are updated, the multitude of services and software components, the blurred lines of responsibility for vulnerability management, and the general opacity of SaaS compound to give the impression that transparency solutions for non-SaaS software are not applicable to SaaS. However, the fact that SBOMs for SaaS are (a) currently underspecified; and (b) cannot offer complete transparency for SaaS does not preclude them

[12] There are national level definitions, for example HIPAA PII, PHI, GLBA, PCI DSS, SOX, GDPR

from being a promising direction for transparency of SaaS. Even for non-SaaS software, SBOMs are not a silver bullet, and they never claimed to be.

SBOMs can be used in conjunction with other transparency practices to enable Choosers and Operators to make informed decisions regarding the software that they incorporate into their operational domains. These transparency practices include but are not limited to requesting information regarding service identifier, provider, function, protocol, location, agreement, and status, as well as data flow and classifications. The additional data fields supplement the software component data provided through the SBOM. SBOMs can also serve as an important part of a Provider's documentation that is integral to obligations for regulatory compliance requirements.

Future Work

To address the gaps in transparency for SaaS, additional discussion and analysis from the perspectives of practitioners across the software ecosystem is needed, as well as future work exploring data governance indicators, service availability indicators, risk indicators, transitive dependencies of services, and other risk indicators.

Data Governance

The work of defining data governance indicators usually falls to compliance or regulatory experts in organizations based on existing policy. One area of discussion is defining "data handling" descriptors such as whether data is ingested, processed, or forwarded by the service, what other services have access to the data, and what encryption is being used in transit and at rest. Another area of discussion is "data discovery," which is concerned with identifying user data location and classification. Although the recommended data fields include service location and data classification, using the fields to facilitate data discovery is not addressed and requires further discussion.

Service Availability Indicators

The terms of service that are accepted by the consumer typically govern the availability of a service. It would be useful to convert service level agreements into a machine-readable format so they can be understood by Choosers, Operators, and Subscribers. An example application of this data would be identifying when there is a change in the terms of service so an alternative may be planned for and put in place. Another application is identifying when a certificate expires, and the service is no longer accessible.

Risk Indicators

It is well understood that hosted services change frequently due to the DevOps deployment workflow development teams adopt to add features, fix bugs, and make updates. Introducing change in a system introduces risk. Inventory, by design, takes a snapshot of the system at a point in time. It becomes extremely inefficient to keep track of every single change introduced to the system. Furthermore, not every change is risky. Discussion on how to identify what parts of transparency data is applicable for risk analysis and what parts can be safely ignored, as well as the scenarios governing these decisions, need further discussion.

Transitive Service Dependency Considerations

Transitive service dependencies are important for transparency into SaaS services, but encompassing transitive dependencies in a standardized format is not sufficiently mature to propose at this stage. In a “traditional” approach to SBOM, it is acknowledged that risks are often not top-level dependency, but in dependencies of dependencies. The NTIA has recommended using interlinking SBOMs.[13] The need for transparency through multiple levels in a supply chain is demonstrated by observing risks from low-level inclusions, such as the canonical 2021 Log4J example[14].

The same can be applied here, including the ability to signal known unknowns. However, tracking this data has several important limitations. Not all transitive service dependencies are visible. Some services are more stochastic or driven by user input or user properties. At a logical extreme, by using third-party services that in turn could rely on other large services, the scope could extend to the entire Internet.

Further work is required to clarify this transitive service dependency approach, with a focus on mapping to the risk management goals of transparency. Requesting information on direct dependencies in the meantime also offers value, especially for use cases dealing with sensitive data to be compliant with relevant standards and regulations. In the absence of standardized formats for communicating software component and service transparency data, Choosers and Operators should initiate discussions with Producers to ensure mutual understanding of SBOM in a SaaS context.

[13] NTIA Open Working Group on SBOM Framing. Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM). October 21, 2021. https://www.ntia.gov/files/ntia/publications/ntia_sbom_framing_2nd_edition_20211021.pdf
[14] Cyber Safety Review Board. Review of the December 2021 Log4j Event. July 11, 2022. https://www.cisa.gov/sites/default/files/publications/CSRB-Report-on-Log4-July-11-2022_508.pdf;

Acknowledgments

Adrian Diglio, Microsoft
Allan Friedman, CISA
Bhargav Vivekanandan, Blue Shield of California
Cassie Crossley, Schneider Electric
Charles Kelly, SAP
Christine O’Leary, Intel
Craig Rubin, HPE
Daniel Bardenstein, Manifest
Deanna Medina, Honeywell
Doug Cavit, Cavit and Hohman
Duncan Sparrell, sFractal Consulting
Emily Fesnak, Deloitte
Isaac Hepworth, Google
Ivana Atanasova, VMware
Ixchel Ruiz
Jeremiah Stoddard, INL
Jim Routh, Saviynt
Joerg Eschweiler
Joyabrata Ghosh, CARIAD SE
Lynn Westfall
Michael Greco, ServiceNow
Nisha Kumar, Oracle
Rene Pluis, Philips
Ricardo Reyes, Tidelift
Scott Armstrong, Interos
Shafia Zubair, JCI
Steve Springett, ServiceNow/OWASP
Tom Alrich
Trevi Housholder, Prudential
Victoria Ontiveros, CISA

Appendix

Use Cases

Organizations along the software lifecycle face different information gaps, risks, and responsibilities regarding the security of SaaS. Using the framing of the four roles (Producer, Chooser, Operator, Subscriber), the introduction of this document noted that the benefits of SBOM apply to multiple roles and points in the software lifecycle. The use cases below do not apply to all SaaS contexts, but they should help the reader understand the landscape of how transparency can make the overall market for security more efficient.

Producer

The motivations for a SaaS Producer to generate an SBOM are the same as those of a Producer of non-SaaS. In fact, given the rapid pace of modern application development and the increased number of dependencies used to build and run these applications, tracking components with automation-friendly data is even more important. Providing an SBOM, either as a document or as a managed service, to internal teams within the Producer’s organization improves interoperability among internal teams and provides a better picture of the state of the overall service at a given point in time. Sharing SBOM data with a customer communicates transparency and may therefore, increase vendor trust. Furthermore, providing an SBOM may be required by relevant standards or regulatory bodies or may be used to comply with standards of transparency or vulnerability management. Lastly, a Producer may generate an SBOM to fulfill a customer’s request.

Chooser

The phase of choosing software applies to the process through which an organization selects and opts to use a SaaS product. During this process, the software component and service transparency data also supports the Chooser in addressing concerns and performing due diligence to ensure they are meeting their commitments to protect company and customer data that the SaaS Producer and Operator might be responsible for. SBOM data can support this decision in a number of ways beyond checking for known vulnerabilities. Automated analysis can surface questions about risks from end-of-life or end-of-support versions, or risks from OSS projects with insufficient support. SBOM data can also be used in analysis around technical debt or licensing concerns, which would help a potential customer get a more realistic picture of total cost-of-ownership or potential risk of data and functional lock-in.

The Chooser should be aware that they are receiving SBOM data that represents a static snapshot; this is particularly important to note given the dynamic nature of SaaS products. Since the SaaS Producer and Operator will be accepting a greater share of the risk (compared to the distribution of risk management responsibility in non-SaaS) to protect the potential customer, understanding the potential risks in the SaaS product is an important part of the decision-making process. There is value in knowing that a supplier can produce an SBOM, as this demonstrates some level of maturity.

Operator

The Operator is responsible for the delivery and maintenance of the SaaS, including maintaining service level agreements, service up-time, and mitigating risk, e.g., patching vulnerabilities. Operators use a combination of acquired software and subscriptions to other SaaS products to create value for their users. SBOMs give Operators insight into critical components of the SaaS. As such, and in part due to the complexity of SaaS systems, Operators benefit significantly from increased transparency into SaaS. Software component and service transparency is useful to SaaS Operators as it allows them to understand how their acquired software works in the context of other service providers.

Subscriber

The primary mediation method between the Subscriber and the SaaS is through legal contracts (e.g., Terms of Service), a news feed, or status feed provided by the Operator. From this perspective, an SBOM itself does not provide any relevant information to the Subscriber. However, Subscribers can ask for other information described in this document's recommendations. During the lifetime of the Subscriber's use of the SaaS, the Subscriber may transition to the Chooser persona in response to new information about the security or performance of the SaaS.

Illustrative Use Case: Recent Vulnerability Example

In September 2023, a critical vulnerability, CVE-2023-4863[15] (heap buffer overflow in libwebp), was found in the webp format widely used by software producers as part of their products and services. This vulnerability created confusion and extra work to determine if a particular piece of software was vulnerable due to the lack of software component transparency for the affected products.

This scenario offers an opportunity to discuss how the four actors respond to new vulnerability information. The owner of libwebp provided a fix for the vulnerability and published the issue. The library component producer did what was required of them. Now, everyone who utilizes this software will need to recognize their responsibility to patch using the provided software. The library component producer does not have visibility into where their vulnerable code is being used to be more proactive in helping to mitigate this vulnerability.

The following use cases are examples that illustrate an organization's concerns and considerations in light of the vulnerability scenario described above. These examples offer one possible path forward for responding to a CVE in the SaaS context and are not intended to be restrictive or prescriptive.

Producer

SaaS Producers using this library as a first-level dependency may choose whether to update the library. The SaaS Producer may have transitive dependencies that use the library, and those third-parties may also choose whether to update the library. An SBOM tree provides a checklist that helps ensure that no vulnerable code is left in the SaaS. The SBOM may also serve as proof of mitigation if the SBOM is detailed enough (includes transitive dependencies).

Chooser

An organization is looking for a SaaS solution for managing its customer data. In considering cloud-based options from SaaS vendors, the organization requests an SBOM from vendors and uses the SBOM in its decision-making process. An SBOM allows the Chooser to identify vulnerabilities, in this case libwebp. This is important considering the libwebp vulnerability, as the library's presence in an SBOM is now an indicator of a risk that a Chooser has to be concerned about when making risk management decisions. They may also be considering vendor transparency, response times, and the quality of the vendor's response to the vulnerability. The risk, especially in a SaaS environment, can be best evaluated in detail based on Producer attestations in the form of an SBOM.

Operator

In response to the CVE, the Operator reviews available resources to gather information about the vulnerability to find the relevant services, libraries, and transitive dependencies. This research may include reviewing relevant SBOMs to assess the vulnerability's impact on the Operator's software. Based on this information, the Operator may take actions to mitigate immediate risk and may also evaluate and consider more secure software development practices.

Subscriber

The Subscriber, in this scenario, uses a SaaS to host their data. In light of the CVE, the Subscriber may be concerned about the security of their data and look for information from their SaaS Operator. The Subscriber does not have direct visibility into the vulnerability's impact on the SaaS. As a result, the Subscriber will be working with the SaaS Operator to obtain information and/or mitigations for the impact of the vulnerability on their service.

When the subscription cycle allows, the Subscriber may revisit the Chooser role and re-evaluate continuing the service relationship with the Operator. See above for the Chooser's considerations.

Architecture Example

The examples below describe scenarios where additional software service transparency would provide substantive value to each of the stakeholders interacting with the SaaS.

Application using Third-Party Services

The term application can mean a desktop application, mobile application, or an edge device (IoT or OT) that may look self-contained when distributed or installed but calls to one or more third-party services when running. A Chooser would want to know about these third-party services, and an application Operator may need to monitor the network traffic from the application to the third-party services.

Service using Third-Party Services

A service is any software that is accessed over a network. A service may use one or more third-party services to deliver some business value. The Producer of the service may want to inventory their third-party services to design the service for redundancy and security. A Chooser would want to know about these third-party services in order to understand and manage risk. An Operator of the service may need to monitor the availability of the third-party services.

Services used by Thin Clients

A thin client is an application that has just enough functionality to access the service's Application Programming Interfaces (APIs).

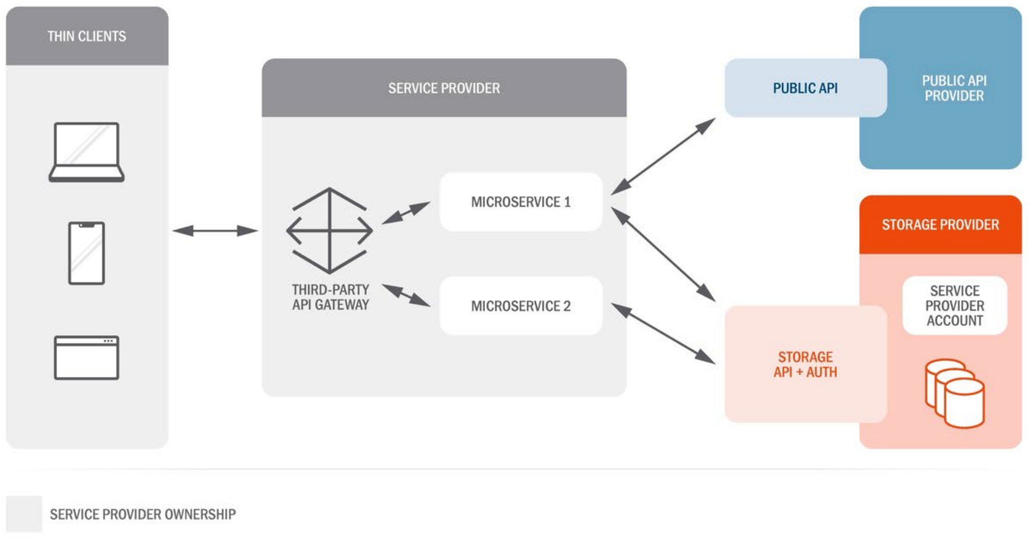

Figure 2 – Services Used by Thin Clients Architecture.

In this example, the Producer and Operator belong to the same organization. The Producer creates both the hosted service and the thin client. The Operator operates the hosted service. The Producer and Operator's organization provides instructions to Subscribers on how to operate the clients. The thin client's functionality is very minimal. Therefore, there may not be much difference in SBOM among the distributions. However, the service itself is composed of a combination of services operated by the Operator and third-party services. A Chooser would want to know details described in the Data Fields section along with a traditional SBOM for the thin clients. If the Producer and the Operator belong to the same organization, the Operator may already have access to the service transparency data.

Web Applications using Third-Party Functionaries

Producers and Operators such as websites or web applications use third-party services to integrate certain functions such as authentication, payment, or federated access to other services. In the following example, a hosting service uses a content delivery network (CDN) provider. The CDN, in turn, has an identity provider integration. A hosting service may provide this identity provider integration as a service option for a Subscriber.

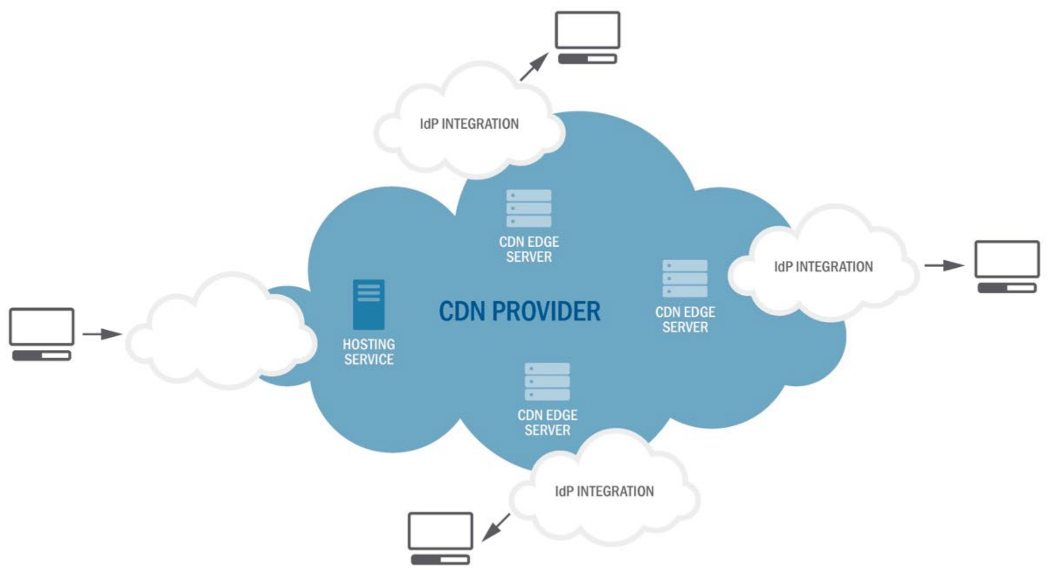

Figure 3 – Web Applications using Third-Party Functionaries Architecture.

The Chooser would want to know the list of third-party functionaries used by the service, with details concerning data governance. The Producer may use information about the third-party services to design their web application for security and availability.

Cloud Services

Cloud Service Providers (CSP) take on the role of Producer and offer a variety of services depending on how much a consumer would like to offload to them. Every service offering is backed by a few internal services which are in turn backed by hundreds of microservices, all operated by the CSP within their own infrastructure. Depending on the type of service, a Chooser and Operator have, to some degree, access to information regarding the services, as it is their responsibility to set up and run the service.

For example, a storage service is a typical cloud provider service offering. It is up to the Operator of the service to control access to the service and take care of the data classification within the service. The CSP, as Producers and Operators, in turn ensures the service is always functional, and the data and service is recoverable in the event of an outage. A CSP would benefit from maintaining SBOM and service transparency information for their own infrastructure and operation.

Data Fields Table

The data fields proposed here for software service transparency in SaaS software are intended to be a starting point, the first step in a longer journey towards establishing best practices for SaaS software transparency.

Table 1: Proposed Service Transparency Data Fields

Field	Description	Persona Who Benefits	Data Application
Service Functions	The types of functions the service provides. (e.g., identity, authentication, certificate authority, CNA, load balancing, etc).	Chooser, Subscriber	Infrastructure Governance, Regulatory Compliance
Service Location	The geographical location where the service is hosted. Cloud Providers list these as us-east, brazil-south, etc. Multiple locations may be listed here.	Chooser, Operator	Data Governance, Regulatory Compliance
Service Protocol	Communication protocol used by service endpoints (e.g., http, https, mqtt).	Chooser, Operator	Regulatory Compliance, Infrastructure Governance

Service Agreement	The text from or link to the Terms of Service agreed to by the consumer of the service.	Chooser, Subscriber, Operator	Service Availability
Service Status	A link to the status page showing service uptime information.	Subscriber	Service Availability
Data Flow	Unidirectional or bi-directional.	Operator	Data Governance
Data Classifications	The classification of the data being ingested by the service (e.g., PII, PHI, confidential, and public).	Chooser, Operator	Data Governance

Glossary

Software-as-a-Service (SaaS)	Software whose deployment, support, management, maintenance, and entire lifecycle are controlled by a supplier external to the consuming organization.
Chooser	A Chooser is looking for software and/or services that meet their organization’s needs.
Operator	Operators are currently managing a system using the SaaS in question.
Producer	A Producer is the creator of the software.
Subscriber	The Subscriber receives information on the SaaS from the Operator.