REFERENCE MONITOR FOR THE INTERNET OF THINGS

By Gunnar Peterson

The information security landscape is dominated by access control technologies. Identity and access management standards and products have carved out a niche in every Fortune 500 Information Security Department. Two questions on IAM tools challenge information security: Are they effective security mechanisms and are they cost effective? Or, in keeping with the theme, what futures are available to us and how should we choose?

Broadly speaking, these questions relate to how well access control aligns with what we can call "American-style" governance. On the one hand, identity and access management systems deploy maximal rulesets that define precisely what subject and object interactions are permitted, and that which is not permitted is forbidden. This requires that the subjects, objects, and interactions are fully enumerated and well understood (two things the software industry is not famous for). Strategies more aligned with American-style governance feature minimal rulesets and accountability. This style of governance leads security in the direction of accountability strategies that require visibility so as to enable accountability, rather than maximal rulesets that define all possible states in an access control matrix. Even weak security controls can be among the most costly parts of a system to develop and deploy. Events show that cost and complexity must be factored into every decision that technologists make. Accountability strategies offer concrete benefits in both areas and, in doing so, offer new possibilities. However, work is required to build this future. Those who do not record their history are doomed to uncertain futures.

Building Security In vs. Building Visibility In

"Normally, everything is split up and problems are solved separately. That makes individual problems easy to solve, but the connections between the problems become very complicated, and something simple ends up in a real mess. If you integrate it in the first place, that turns out to be the most simple solution. You have to think ahead and you must always expect the unexpected."
— Jan Benthem, chief architect, Schiphol airport

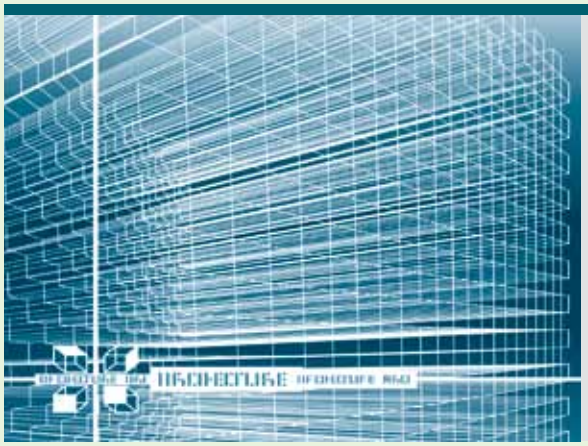
The idea of software security engineering is to build software that continues to perform even in the face of attack. This means that access control must be enforced, sensitive information must be kept confidential, and threats and failure modes must be accounted for. To enable the *Build Security In* approach, software development organizations engage in activities throughout the software lifecycle from development to deployment to operations. These activities typically include, but are not limited to: data classification, architectural risk analysis, design review, threat modeling, static analysis, identity and access management, code review, key management, system hardening, vulnerability scanning, and incident response. The industry has made tremendous progress in each of these areas over the last ten years; problems, however, remain.

Each *Build Security In* activity domain adds cost and complexity. We did not see security budgets evaporate in the Great Recession, but already cost-sensitive businesses are more focused on cost than ever. Whether or not businesses eventually gain from their security investments, the *Build Security In* activities add short term cost. This means decisions will be scrutinized not only as architectural issues, but decided by the stroke of a pen at budgeting time. However, cost is not likely to be the main issue facing *Build Security In*; since the industry has made progress in the area, companies do get the benefits of tools, patterns, and practices on a much wider scale. For example, static analysis is available from IBM and HP as a standard offering, not an expensive guru, but rather a "this is how you build software today" package from two leading vendors. Even more interestingly, companies increasingly make decisions on their static analysis tools not based on scanner findings, but whether the tools integrate better into the source code management and bug tracking systems.

When *Build Security In* tools are commoditized by major vendors, direct cost is reduced but the overarching challenge in *Build Security In* is, and will remain, complexity. There is no standard or set of standards that streamlines consistent communication across all the major activities required by *Build Security In* at design time, deployment time, and run time. More to the point, vastly different organizational skills (read: people) are required for different activities. The landscape is improving but will remain Balkanized. An organization could have an adept threat modeling team, a competent static analysis team, and substandard architectural and operational talents. Variability is to be expected, but what compounds the issue in *Building Security In* is the lack of coherent communication across the domain activities, each uses its own model and process, which is the last thing you want if your goal is engineering reliability.

As an alternative to *Build Security In,* the idea of *Building Visibility In*[1] enables accountability-based governance rather than control-based governance, and does so at far lower cost and complexity. The credit card industry is the canonical private sector example of the effectiveness of visibility strategies. Credit issuing banks regularly mail small plastic cards worth $3,000-$10,000 and sort out the challenges of fraud and payment on the back end. The credit card issuers network is comprised of three main areas of visibility: first, selecting to whom the cards are mailed; next, the merchant and merchant terminal; and finally, the runtime transaction. This has enabled global deployment, and, crucially, with very few controls built in. There is a registration process for consumers and merchants to be sure, but beyond that it is visibility that governs the system events.

How is this accomplished in information systems? Again, the credit card industry has led the way. The PCI DSS standard (itself an example of *Building Accountability In*) mandates that companies must "track and monitor all access to network resources



Whether or not businesses eventually gain from their security investments, the *Build Security In* activities add short term cost. This means decisions will be scrutinized not only as architectural issues, but decided by the stroke of a pen at budgeting time.

and cardholder data." Unlike other standards, PCI DSS goes to prescribe very specific audit logging architecture. Given the number of companies impacted by PCI DSS, the knock on effect of this has been to create a marketplace for log management vendors whose systems offer ways to tamperproof, provide integrity and reporting, and other secure log management capabilities. The log management vendors provide real value to the industry because organizations lack the ability to build these tools on their own, however, log management is in some sense the easiest, or at least most concrete, problem to solve.

The core issues that drive the overall effectiveness of the *Build Visibility In* strategy are not simply the sanctity of the console and reporting engine the analyst consumes on the back end, but rather what sources and targets are registered by the audit log observer, what events the audit logger has visibility into, and what event data is useful to the incident responder. PCI DSS mandates some portion of each of these, but these are focused on credit card data. Leaving audit log management aside, effective audit logging requires the following technical architecture elements:

A) The audit loggers place(s) in the stack have visibility into authoritative sources of the origin (for example, the user principal and the system they authenticated to) and initiator of the event (for example, the browser client application making request on behalf of user principal and the system it's bound to), the target of the event (for example, the web service being called), and the source of the event (for example, the messages passed over the network). This sounds trivial, but issues such as artifact resolution make it problematic.

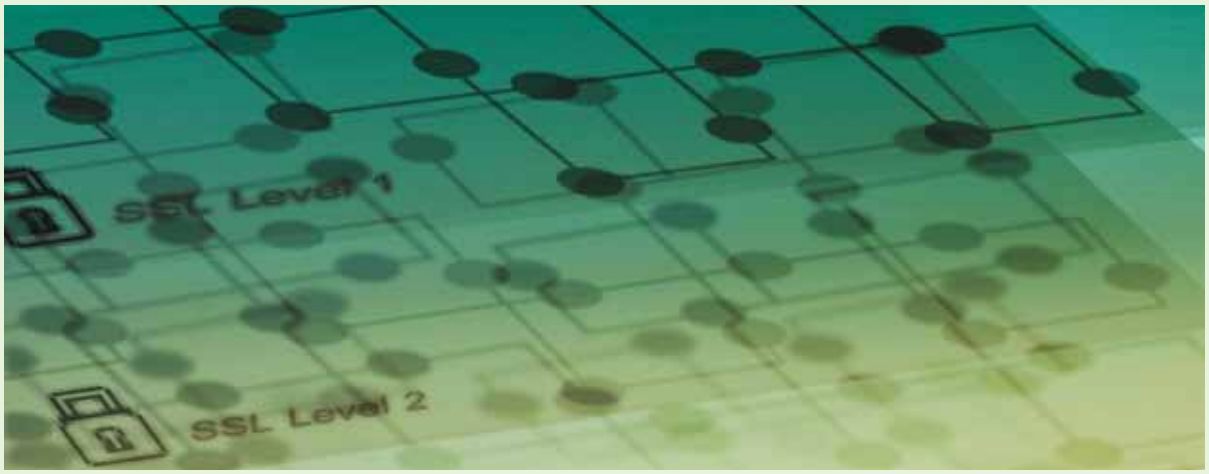
B) The audit loggers' event model and what events it is aware of must be mapped to the application origins, initiators, targets, and event sources. The audit loggers' event model implements observers on the subjects, objects, and event streams that it monitors. For client side events, this might include authentication or privilege change events, and for server-side objects this might include session and resource access events.

C) Audit Record Format: The event model must structure its observations in a consistent way to enable effective incident response.

The closer the audit log observer is to that which is being monitoreds the more relevant and contextual the audit records are likely to be. Network monitoring is not an answer; networks are too dumb; they lack context about application logic, rules, policies, identity, and data. Further, network monitors are in no position to verify information, meaning they are limited to reporting unverified partial point in time data streams.

Characteristics of the Thingfrastructure

"The Internet of Things — An Action Plan for Europe"[2] report discusses implications of current trends in Internet of Things (IOT) including mobile, RFID, Near Field Communication (NFC), 2-D bar codes, wireless sensor/actuators, Internet Protocol Version 6 (IPv6),



ultra-wide-band, or 3/4GOT. The report identifies three major trends:

- Scale: The number of connected devices is increasing, while their size is reduced below the threshold of visibility to the human eye.
- Mobility: Objects are ever more wirelessly connected, carried permanently by individuals and geo-localisable.
- Heterogeneity and Complexity: IOT will be deployed in an environment already crowded with applications that generate a growing number of challenges in terms of interoperability.

The issue of scale is that with devices getting smaller all the time and more widely deployed, there is a concomitant trend towards lower power chips like RFID that offer limited or no storage capacity. Lower power systems will not likely offer robust messaging due to lack of queueing, local storage, and processing; however, they can be widely deployed due to the relative cheapness of the devices. So individual failures may be resolved on the server side, through voting or other reputational algorithms.

Mobility and geolocation services offer some possible advantages to audit logging by enriching the audit record data with a more precise location for a given audible event. However, for this combination to be useful, the audit log observer must be tamperproof and always on.

For most of the past decade, technologists worried about monoculture and cascade failure, but going forward the future looks increasingly heterogeneous. iPhone, Kindle, Blackberry, and Droids are all examples of products selling in the millions based on proprietary operating systems, hardwares, and even networks. Heterogeneity has the advantage of an excellent hedge against cascade failure, but has the side effect of overwhelming complexity. Deploying audit loggers to the four aforementioned devices would be four families of codebases to simply support the operating system; that is before the audit logger is integrated into the rest of the proprietary infrastructure.

Applying Reference Monitors in IOT Considerations, Issues, and Barriers

"The real trouble with this world of ours is not that it is an unreasonable world, nor even that it is a reasonable one. The commonest kind of trouble is that it is nearly reasonable, but not quite. Life is not an illogicality; yet it is a trap for logicians. It looks just a little more mathematical and regular than it is; its exactitude is obvious, but its inexactitude is hidden; its wildness lies in wait." — G.K. Chesterton

Chesterton benefits from careful reading; pay attention to the last sentence. Authentication, authorization, and cryptography attempt an exact partitioning of the system into secure and insecure states. The sad fact is that since security is not achieved, the system remains insecure. Attackers know and exploit these

gaps, however, the answer is not "add more precise partitioning between secure and insecure states," but rather in applying visibility into how the system is used in the real world — regardless of state.

Applying audit logging in the IOT, means tackling the following issues:

A) Event Ownership: Even in a simple example, there are likely to be multiple participants in a federated relationship for a mobile use case. The event stream owner will vary between the user side, the network side, and the server side, depending on what part of the sequence of events the message is traversing. An application is developed by one or more firms, then signed for distribution into the proprietary application distribution center, sent over a proprietary network, verified and installed on a proprietary OS on the handset and possibly interacts with secure local storage, and then it is finally consumed by the user. Assigning ownership to each part of the event sequence is necessary to generate the audible event stream and to be able to rehydrate events on the back end.

B) Assurance: Due to the complex relationships and responsibility, it is difficult to create the end to end view necessary for hardware, software, and process assurance activities

C) Occasionally Connected: Mobile devices and lower power devices can be relied on to do one thing — go silent from time to time. This can be the result of normal events like network or power failure, or it can be a malicious way to hide activity beacons. For this reason, local storage and a way to protect and verify this storage is essential. Message and transaction counters that can be reconciled later with the server side message count are one partial way to do this.

D) Lack of Logging Standards: Given the amount of different technologies, lack of standards in the logging area adds to the challenge of consistent audit record generation. There are two promising candidates — CEE and XDAS[3,4] — but neither has been adapted to mobile and IOT use cases.

E) Quality of Visibility: Due to lower power in mobile and IOT, the information that is passed to these system is almost always partial. In the case of SAML, a normal SAML token is sent by value and is likely to have information about the assertion issuer, the certificate authority, the authentication authority, authorization information, and relevant attributes and values. This is stark contrast to what data is generally sent as SAML token by reference instead of by value. A SAML token by value is often dozens of lines long filled with information, all quite obvious and tagged, that is helpful to audit log observers. On the other hand, a SAML token by reference is:
<samlp:Artifact>AAQAAMh48/1oXIM+sDo7Dh2qMp1HM 4IF5DaRNmDj6RdUmllwn9jJHyEgIi8=</samlp:Artifact>

This is opaque from an observer standpoint and must be resolved on the server side. This means that audit loggers require an asynchronous message system that can correlate the events after the fact. Complicating this scenario further is that the client side and server side audit log observers are quite likely to be running in different domains operated by different groups. So the audit logger's view of the system will be subjective based on where it's located, what events it can observe, where it can write those events, and what actions are necessary to take after the fact to rehydrate the event messages.

Conclusion

"Elements stored in a mind do not have names and are not organized into folders; are retrieved not by name or folder but by contents. (Hear a voice, think of a face: you've retrieved a memory that contains the voice as one component.) You can see everything in your memory from the standpoint of past, present and future. Using a file cabinet, you classify information when you put it in; minds classify information when it is taken out. (Yesterday afternoon at four you stood with Natasha on Fifth Avenue in the rain – as you might recall when you are thinking about "Fifth Avenue," "rain," "Natasha," or many other things. But you attached no such labels to the memory when you acquired it. The classification happened retrospectively.)"
— David Gelernter

Audit logging architecture faces technical challenges, such as decentralized architecture, occasionally connected mobile devices, and low powered "things" like RFID. But the more substantial problem is a priori knowledge on development, deployment, and usage. For access control-based architecture, this combination is a crippling blow because those architectures rely primarily on being able to partition the system into secure and insecure states at design time, and then implement that version of the future into some digital runtime.

For accountability-based architectures, the lack of a priori predictive efficacy is a challenge because event models and audit records require some up front modeling and assumptions, however, accountability strategies can still deliver value amidst uncertainty. Just as in real estate, it's "Location, Location, Location." The security architect may not be able to partition the system into all possible states, but it's quite likely that she will be able to identify the primary subject and object assets. Then the job ahead becomes integrating audit loggers to enable visibility into events and searching those events. Q

Gunnar Peterson *is a Managing Principal at Arctec Group. He is focused on distributed systems security for large mission critical financial, financial exchanges, healthcare, manufacturer, and insurance systems, as well as emerging startups. Mr. Peterson is an internationally recognized software security expert, frequently published, an Associate Editor for IEEE Security & Privacy Journal on Building Security In, a contributor to the SEI and DHS Build Security In portal on software security, leads OWASP Web Services Top Ten project, a Visiting Scientist at Carnegie Mellon Software Engineering Institute, and an in-demand speaker at security conferences. He maintains a popular information security blog at http://1raindrop.typepad.com.*

REFERENCES

[1] Build Visibility In," Richard Bejtlich
http://taosecurity.blogspot.com/2009/08/build-visibility-in.html

[2] "The Internet of Things — An Action Plan for Europe"
http://ec.europa.eu/information_society/policy/rfid/documents/commiot2009.pdf

[3] "A Standardized Common Event Expression for Event Interoperability"
http://cee.mitre.org/

[4] "Distributed Audit Services Project"
http://www.opengroup.org/projects/security/xdas/