Pre-Deployment Security Assessment for Cloud Services through Semantic Reasoning (Extended Abstract)***

Claudia Cauli[1], Meng Li[2], Nir Piterman[1], and Oksana Tkachuk[2]

[1] University of Gothenburg
[2] Amazon Web Services

Over the past ten years, the adoption of cloud services has grown rapidly, leading to the introduction of automated deployment tools to address the scale and complexity of the infrastructure that companies and users deploy. The practice of configuring, deploying, and updating systems resources from source code files is known as Infrastructure as Code (IaC) [17]. In addition to instructions relevant for resource creation, dependencies, and updates, IaC configuration files contain information about settings, dataflow, and access control. Without the aid of automation, ensuring the security of such deployments becomes more and more challenging.

In this study, we focus on the first IaC tool ever introduced: Amazon Web Services' CloudFormation. In particular, we investigate the application of description logics to the formalization and reasoning over IaC deployments. We are interested in three aspects: *(i)* whether proposed cloud IaC configurations comply with security best practices, *(ii)* how to aid users in building more secure infrastructure *before* deploying it, and *(iii)* to what extent formal automated techniques can support manual pre-deployment security reviews.

We provide a framework to encode IaC into description logic, and investigate its effectiveness in answering configuration queries and reasoning about dataflow, trust boundaries, and potential issues within the system. Specifically, we test DLs reasoning capabilities to infer new facts about underspecified resources (such as those not included in a given deployment but used by it) and leverage DLs *open-world assumption* to perform verification and refutation, depending on the property being checked. We formalize additional security knowledge that allows for checking system-level semantic properties; i.e., properties that consider the nature of the cloud environment and more complex reachability over an *inferred* graph representation of the infrastructure.

Formalizing and Encoding Infrastructure as Code According to the IaC paradigm, resources are deployed on the cloud by writing one or more JSON-formatted *configuration files*, by which users configure settings and communication of the desired instances. Configuration files must validate against the *resource specifications* [10], which are files prescribing resource properties and their allowed values.

* Full paper published in CAV 2021.

In this study, we formalized both configuration files and resource specifications, and encoded these into assertional and terminological knowledge, respectively.

Checking Compliance to Security Requirements We query the resulting models with properties related to the security posture of the infrastructure being deployed. Properties are grouped into three categories: *security issues*, *mitigations*, and *global protections*, which we view in analogy to *must* and *may* specifications, which one would use to express that an issue may be present (vs. must be absent) or that a protection must be in place (vs. may be missing). Different query structure and pass/fail logic characterize each property type. We implement the encoding and the security properties verification procedures in a publicly-available tool, CloudFORMAL [9], and report on our findings. When applying the developed toolchain to publicly available deployment files, we find numerous violations of widely-recognized security best practices, which suggests that streamlining the methodologies developed for this case study would be beneficial.

Semantic Reasoning for Dataflow Analysis By extending the models with dataflow-specific knowledge, we use more comprehensive semantic reasoning to further support security analyses. We manually craft two proof-of-concept ontologies of terms related to cloud security. These formalize several common cloud terms, such as account, deployment, authenticated and unauthenticated users; generic dataflow terms, such as storage, process, nodes, and flows of different kind; and service-specific dataflow terms. By adding these on top of the underlying IaC formal specification, we can reason about the higher-level business logic and reachability of the infrastructure, and we can abstract it and visualize it in a more convenient way. This is where the full inference power of description logics comes into play.

To the best of our knowledge, no formal automated technique currently exists to verify cloud deployments during the design phase. Formal reasoning techniques have been successfully applied to different aspects of the cloud, e.g. networks and access policies [11,3,1,2]. Non-formal tools exist that recommend and run checks against *already deployed* resources [20,8], or scan IaC templates [7,6,21] for syntactical patterns violating security best practices. Large-scale configuration problems have been tackled with description logic before [15,16]. Semantic-based approaches, even DL-based, are being used to do conceptual modeling of security engineers' expertise with the provable and explainable inference capabilities of logics (see, e.g., the OWASP *"Ontology-driven Threat Modeling"* project [18]). We took advantage of DL's open-world assumption to implement, in our properties encoding, verification and falsification. An alternative approach would be to use 3-valued models with labels on states and transitions and apply model checking [4,5]. However, expressive branching-time logics [14,19] have not been studied in the context of 3-valued models and we are also not aware of tool support at the level available for DLs (cf. [12,13]).

All the results in this study can be generalized to other existing IaC frameworks. We plan to continue researching for an even better-fitting description logic

formalism, query language, three-valued semantics, and decision procedures for verification and falsification of properties relevant to security analyses, such as dataflows, trust boundaries, and threat modeling.

References

1. Backes, J., Bayless, S., Cook, B., Dodge, C., Gacek, A., Hu, A.J., Kahsai, T., Kocik, B., Kotelnikov, E., Kukovec, J., McLaughlin, S., Reed, J., Rungta, N., Sizemore, J., Stalzer, M.A., Srinivasan, P., Subotic, P., Varming, C., Whaley, B.: Reachability analysis for aws-based networks. In: CAV (2). Lecture Notes in Computer Science, vol. 11562, pp. 231–241. Springer (2019)
2. Backes, J., Bolignano, P., Cook, B., Dodge, C., Gacek, A., Luckow, K.S., Rungta, N., Tkachuk, O., Varming, C.: Semantic-based automated reasoning for AWS access policies using SMT. In: FMCAD. pp. 1–9. IEEE (2018)
3. Bouchenak, S., Chockler, G.V., Chockler, H., Gheorghe, G., Santos, N., Shraer, A.: Verifying cloud services: present and future. Operating Systems Review **47**(2), 6–19 (2013)
4. Bruns, G., Godefroid, P.: Model checking partial state spaces with 3-valued temporal logics. In: CAV. Lecture Notes in Computer Science, vol. 1633, pp. 274–287. Springer (1999)
5. Bruns, G., Godefroid, P.: Model checking with multi-valued logics. In: ICALP. Lecture Notes in Computer Science, vol. 3142, pp. 281–293. Springer (2004)
6. The AWS CloudFormation Linter (2020), `https://github.com/aws-cloudformation/cfn-python-lint`, Last accessed on 2020-10-15
7. The CFnNag Linting Tool (2020), `https://github.com/stelligent/cfn_nag`, Last accessed on 2020-10-15
8. Infrastructure Security, Compliance, and Governance (2020), `http://www.cloudconformity.com/`, Last accessed on 2020-08-04
9. CloudFORMAL: Prototype Implementation, `http://github.com/claudiacauli/CloudFORMAL`, Last accessed on 2020-10-15
10. Resource Specification (2020), `https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-resource-specification.html`, Last accessed on 2020-08-13
11. Cook, B.: Formal reasoning about the security of amazon web services. In: CAV (1). Lecture Notes in Computer Science, vol. 10981, pp. 38–47. Springer (2018)
12. D'Ippolito, N., Fischbein, D., Chechik, M., Uchitel, S.: MTSA: the modal transition system analyser. In: ASE. pp. 475–476. IEEE Computer Society (2008)
13. Gurfinkel, A., Wei, O., Chechik, M.: Yasm: A software model-checker for verification and refutation. In: CAV. Lecture Notes in Computer Science, vol. 4144, pp. 170–174. Springer (2006)
14. Kupferman, O., Grumberg, O.: Buy one, get one free!!! J. Log. Comput. **6**(4), 523–539 (1996)
15. McGuinness, D.L., Resnick, L.A., Jr., C.L.I.: Description logic in practice: A CLASSIC application. In: IJCAI. pp. 2045–2046. Morgan Kaufmann (1995)
16. McGuinness, D.L., Wright, J.R.: Conceptual modelling for configuration: A description logic-based approach. AI EDAM **12**(4), 333–344 (1998)
17. Morris, K.: Infrastructure as code: managing servers in the cloud. " O'Reilly Media, Inc." (2016)

18. OWASP Ontology-driven Threat Modeling, `https://github.com/OWASP/OdTM`, Last accessed on 2021-05-17
19. Sattler, U., Vardi, M.Y.: The hybrid μ-calculus. In: IJCAR. Lecture Notes in Computer Science, vol. 2083, pp. 76–91. Springer (2001)
20. Multi-Cloud Security Auditing Tool (2020), `http://github.com/nccgroup/ScoutSuite`, Last accessed on 2020-08-04
21. Static Analysis Security Scanner for Terraform (2020), `https://tfsec.dev/`, Last accessed on 2021-05-10