V3SPA: An IDE and Visualization Environment for SELinux Security Policy Abstractions

Robert Gove, Christopher Wacek, Matthew Oertle, Jeffrey Karrels
{robert.gove, christopher.wacek, matthew.oertle, jeff.karrels}@invincea.com
Invincea Labs

ABSTRACT

SELinux policies have enormous potential to enforce granular security requirements, but the size and complexity of SELinux security policies makes them challenging for security engineers and analysts to determine whether the implemented policy meets an organization's security requirements. To address the challenges in developing and maintaining SELinux security policies, we are developing V3SPA (Verification, Validation and Visualization of Security Policy Abstractions). V3SPA creates an abstraction of the underlying security policy using the Lobster [3] domain-specific language, and then tightly integrates visualizations of the policy abstraction with an IDE for editing the underlying policy and visually seeing the changes in real time. We describe the current state of the V3SPA system, and discuss our future research goals.

1 INTRODUCTION

As system complexity has increased, security goals have become incomprehensible at the implementation and configuration levels of an operational environment. To secure modern enterprise systems, the ability for security professionals to abstractly define and visualize the overall security policies of the system is essential to understanding and verification. While some tools exist to implement the system security policies themselves, the complexity of operational systems clutter the ability of security engineers to map security goals to proper system configurations, thus leaving a high probability of security errors. Furthermore, as system complexity increases, so does the difficulty in assessing differential system modifications without requiring a ground up re-verification.

Past research on security policies has focused on algorithmic and visual techniques for identifying information flow in security policies [1] [6], or techniques for grouping types in the security policy graph to simplify the analysis [4].

We propose a novel system to address the challenges faced by security engineers and analysts. In the same way that modern scooters are designed with a complete enclosing that covers the underlying complexities of the engine, V3SPA abstracts away the complexities of SELinux security policy development. V3SPA is an interactive visualization and security policy development environment built on top of the Lobster domain-specific language [3]. V3SPA is an on-going research project, and we report on its current state of development, as well as our research goals in the next phase of the project.

2 V3SPA DESIGN

V3SPA is a web-based IDE for exploring security polices, with three main user interface components: (1) the editor pane, which contains interface controls and a fully featured IDE; (2) the visualization workspace allows users to explore security policies interactively; and (3) the console displays logging and error messages. See Figure 1 for a screenshot of the interface.

From the editor pane, users can select a visualization to display in the visualization workspace, view the source files, view and edit the Lobster DSL abstraction of the security policy, and save or export modifications to the security policy. The visualization workspace shows a visual representation of the security policy abstraction. Users can interact with the visualization by hovering over visual elements to see tooltips, zooming and panning, or filtering out some of the data. The visualization updates in real time as users edit the security policy in the editor pane. Finally, the console at the bottom of the screen displays logging information and errors when there are problems parsing the security policy.

Currently, V3SPA includes three visualizations of the abstracted security policy:

1. The summary visualization displays SELinux types and object classes, and connections between then represent permissions (see Figure 1. Users can filter the permissions by clicking "Active Permissions" or"Permitted on", which also hides corresponding types that are no longer connected to anything.

2. The structured view shows a hierarchical interactive visualization to explore the policy (see Figure 2. Object classes are drawn within object types, and connections representing permissions are drawn from subject types to object classes in subject types.

3. The module view displays the module hierarchy in an icicle tree. Similar to the structured view, connections are drawn between object classes underneath types, indicating permissions.

In all visualizations, users can right-click on a type or connection and click "Show code", which highlights the security policy code in the editor pane that corresponds to the selected security policy element.

3 FUTURE WORK

As we move forward with the research and development of V3SPA, we have identified several research goals for the next phase:

Enhanced Visualizations We will enhance the current visualizations in V3SPA to support security engineers and analysts in their complex analysis tasks. As part of this, we will add more dynamic query mechanisms to improve V3SPA's interactivity, and we will design new visualizations to form a cohesive suite of analysis tools.

Raw SELinux Policies We will add a plug-in for V3SPA to also display raw SELinux security policies. Although there are advantages to using policy abstractions, such as the Lobster DSL, some analysis challenges are best met by analyzing the raw SELinux security policy.

Visual Policy Diffing We will build a plug-in to perform visual policy diffs between two security policies. Assessing differential system modifications is challenging due to the large size

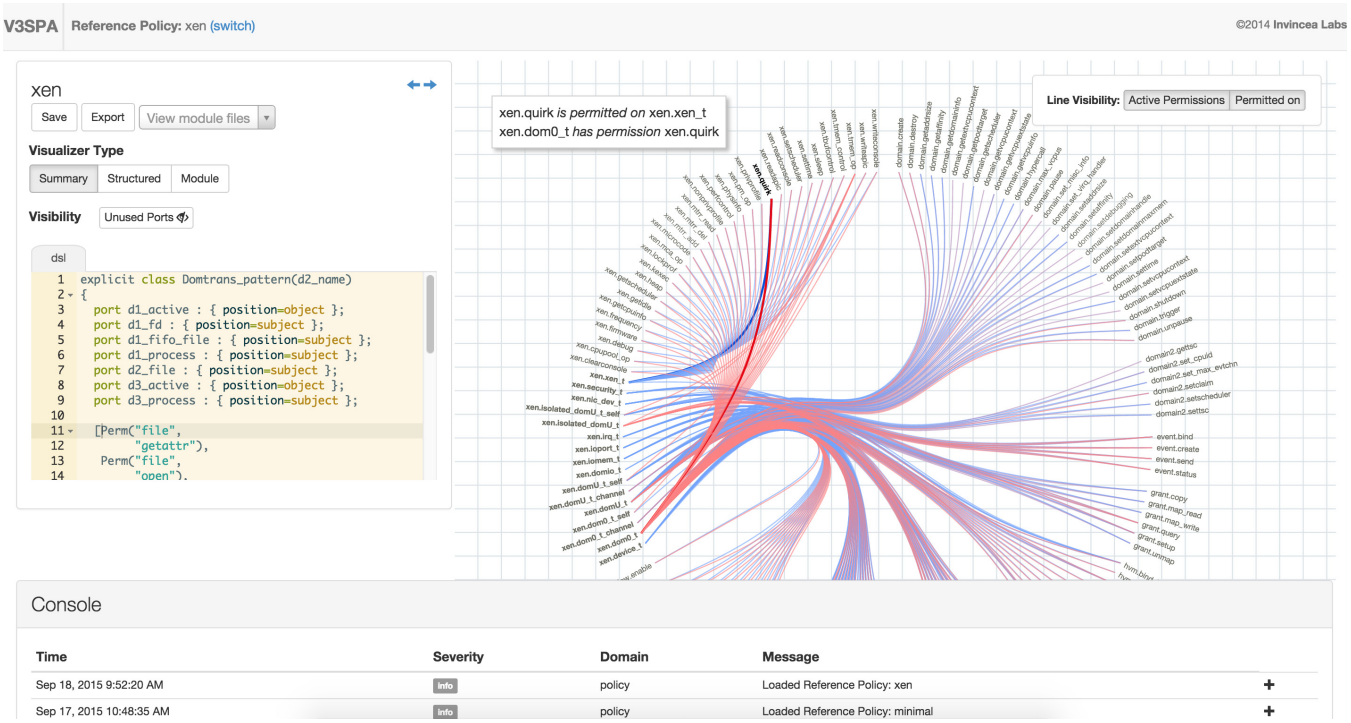

Figure 1: The V3SPA user interface. On the left, the editor pane shows the Lobster DSL representation of the security policy. On the bottom, the console displays logging and error messages. In the middle is the visualization workspace, which allows users to explore security policies interactively (the "Summary" visualization is shown here, which shows relationships between permissions, types, and object classes). Users

of many security policies and assessing modifications often requires a ground up re-verification of the entire policy. Other researchers have explored visualizing comparisons between trees (e.g. TreeVersity [2] and TreeJuxtoposer [5]), but there has been relatively little work on comparing large graphs, like those seen in security policies. We will explore various techniques for visually diffing security policies.

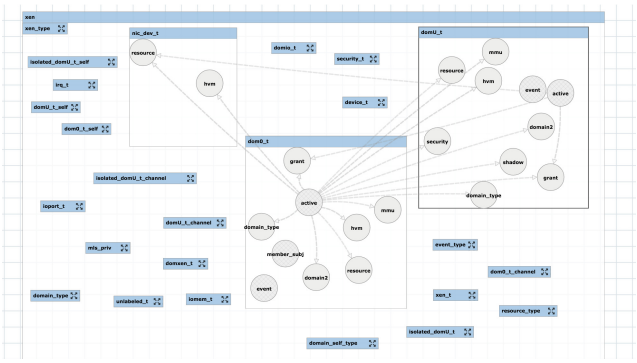

Figure 2: The Structured view in V3SPA shows object classes, and draws connections between object classes, indicating which types have permissions on which classes in other types. Users can expand or collapse a type (the blue rectangles) to see which types have permissions on object classes in that type.

References

[1] P. Clemente, B. Kaba, J. Rouzaud-Cornabas, M. Alexandre, and G. Aujay. SPTrack: Visual Analysis of Information Flows within SELinux Policies and Attack Logs. In *International Conference on Active Media Technology*, pages 596—-605, 2012.

[2] J. A. Guerra-Gomez, M. L. Pack, C. Plaisant, and B. Shneiderman. TreeVersity: Interactive Visualizations for Comparing Hierarchical Datasets. In *Transportation Research Board*, 2013.

[3] J. Hurd, M. Carlsson, S. Finne, B. Letner, J. Stanley, and P. White. Policy DSL: High-level Specifications of Information Flows for Security Policies. In *High Confidence Software and Systems: HCSS*, 2009.

[4] S. Marouf and M. Shehab. SEGrapher: Visualization-based SELinux policy analysis. In *Configuration Analytics and Automation*, pages 1–8, 2011.

[5] T. Munzner, F. Guimbretiere, S. Tasiran, L. Zhang, and Y. Zhou. TreeJuxtaposer: Scalable Tree Comparison using Focus+Context with Guaranteed Visibility. In *SIGGRAPH*, 2003.

[6] W. Xu, M. Shehab, and G. J. Ahn. Visualization-based policy analysis for SELinux: Framework and user study. *International Journal of Information Security*, 12(3):155–171, 2013.