HiFiPol:Browser - Securing The Web Browsing Ecosystem

A Thesis

Presented in Partial Fulfilment of the Requirements for the

Degree of Master of Science

with a

Major in Computer Science

in the

College of Graduate Studies

University of Idaho

by

Ananth Jillepalli

Major Professor: Daniel Conte de Leon, Ph.D.

Committee Members: Frederick T. Sheldon, Ph.D.; Michael Haney, Ph.D.

Department Administrator: Frederick T. Sheldon, Ph. D.

May 2017

Authorization to Submit Thesis

This thesis of Ananth A. Jillepalli, submitted for the degree of Master of Science with a major in Computer Science and titled "**HiFiPol:Browser - Securing The Web Browsing Ecosystem**," has been reviewed in final form. Permission, as indicated by the signatures and dates given below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor:	_____ Daniel Conte de Leon, Ph.D.	Date:_____
Committee Members:	_____ Frederick T. Sheldon, Ph.D.	Date:_____
	_____ Michael Haney, Ph.D.	Date:_____
Department Administrator:	_____ Frederick T. Sheldon, Ph.D.	Date:_____

Abstract

Web-browsers have been so successful today, that they are a necessity to both private and public sectors. Reasons behind such a success are: flexibility through Turing-complete execution and powerful graphic capabilities, which are accessible through network by both trusted and untrusted sites. These capabilities lead to multiple vulnerabilities. To prevent or mitigate the probability of vulnerabilities being exploited, a successful approach can be to configure all the web-browsers by specifying multi-level-granularity and tailored high-level secure policies. Where in, the policies are transformed into configuration files and deployed to all the applicable machines in a domain. In addition, the policies should accommodate the attributes: devices, users/roles, applications, and domains of an organization.

During our background study, we were not able to find any policy-oriented centralized system which had the tailored browser security settings approach, with the ability to accommodate devices, users/roles, applications, and domains. Therefore, we made it our mission to design and practically implement a centralized system called as HiFiPol:Browser, which is a policy-oriented, multi-platform, and high-fidelity security policy management system for web-browsers.

The contributions reported in this thesis are: (a) present a requirement for policy-oriented granularity-supporting web browser configuration tools, by demonstrating one of the current methods for remote web browser configuration, and leveraging upon the disadvantages of current method (b) architecture and component design of HiFiPol:Browser - a policy-oriented, granularity-supporting web browser configuration tool, and (c) designed and developed prototype of a High-level, Easy-to-use, Reconfigurable, Machine Environment Specification Language: HERMES.

Acknowledgments

I would like to extend my sincere gratitude to my major professor, Dr. Conte de Leon, for numerous hours of time he spent training me during these past 2 years. From him, I have not only learned how to perform research and publish results, but also learned how to be a true academician and a noble person in the general society. I would also like to thank Dr. Conte de Leon for financially supporting my research activity during the first year of my study. Thank you.

I would also like to extend my sincere gratitude to the members of my graduate committee. Dr. Sheldon has been like a second major professor to me, always welcoming and discussing new ideas with a prompt response. Dr. Haney has also had an instrumental role in the guidance of my research. His lucid, pinpointed suggestions, and time spent in reviewing, correcting manuscripts; were all invaluable. Thank you all.

I would like to thank Susan Branting, Arvilla Daffin, and Victor House for their administrative and technological support throughout the past two years. I would also like to thank all of the agencies that have financially supported my research: Department of Computer Science at University of Idaho, the State of Idaho: through IGEM grant for Cybersecurity Capacity Building, the Center for Secure and Dependable Systems at University of Idaho, and National Science Foundation.

During the past two years at the Computer Science Department and the Center for Secure and Dependable Systems, I have met and worked with several people, all of whom I cannot list here; including departmental faculty, staff, colleagues, and my students: with whom, I had very enriching discussions. All these people have contributed to the successful completion of this degree. Thank you all very much.

Table of Contents

Authorization to Submit Thesis ..... **ii**

Abstract ..... **iii**

Acknowledgements ..... **iv**

Table of Contents ..... **v**

List of Figures ..... **ix**

1 Introduction, Contributions, and Overview **1**

1.1 Introduction 1

1.2 State of Web Browser Usage 2

1.3 Impact of Web Browsers on Society 3

1.4 The Problem and a Plausible Solution 4

1.5 The Contributions of this Thesis 5

1.6 Author's Related Refereed Publications 6

1.7 Overview of this Thesis 7

2 Hardening the Client-side: A Guide to Enterprise-level Hardening of Web Browsers in Windows **8**

2.1 Introduction 8

2.1.1 Paradigm Shift 8

2.1.2 Chapter's Problem and Threat Model 9

2.1.3 Chapter's Contribution 10

2.1.4 Outline of this Chapter 10

2.2 Preliminaries 11

2.2.1 Active Directory Domain Services 11

2.2.2 Group Policy: Objects and Management 12
2.2.3 Environment Setup 12
2.3 Organizational Structure 12
2.3.1 Setting up Organizational Units 13
2.3.2 Attaching Users and Computers to OUs 14
2.3.3 Remote Installation of Google Chrome 16
2.3.4 Importing Administrative Templates for Google Chrome and Microsoft Edge 17
2.4 Hardening Web Browsers 20
2.4.1 Configuration Template Locations 20
2.4.2 Configuration Hardening Examples 22
2.5 Chapter Conclusion 24

3 An Architecture for a Policy-Oriented Web Browser Management System: HiFiPol:Browser 26
3.1 Introduction 26
3.2 Existing Tools 27
3.3 HiFiPol:Browser Architecture 28
3.4 Knowledge Entities 28
3.4.1 Remote Browser Deployment Platform 29
3.4.2 Centralized Browser Configuration File Database 29
3.4.3 File Transition Processes 30
3.4.4 Application/System Specific Configurations 31
3.4.5 Fully Instantiated Knowledge-Base 31
3.4.6 Policy Knowledge-Base 31
3.4.7 Policy Files 32
3.4.8 Graphical User Interface 32
3.5 Process Entities 33

3.5.1 Policy Implementation Engine 33
3.5.2 Conflict Identification, Resolution and Policy Instantiation Engine 33
3.5.3 Configuration Rewrite Engine 33
3.5.4 Configuration Pre-Deployment Engine 34
3.5.5 Configuration Conversion Engine 34
3.5.6 Configuration Association Engine 35
3.5.7 Policy Abstraction Engine 35
3.5.8 Policy Reflection Engine 35
3.6 Agent Entities 36
3.7 Chapter Conclusion 36

4 HERMES: A High-Level Policy Language for High-Granularity Enterprise-wide Secure Browser Configuration Management 38
4.1 Introduction to the Chapter 38
4.2 Chapter's Problem 39
4.3 Chapter's Approach 40
4.4 Characteristics of the HERMES Language 41
4.5 Formal Grammar of HERMES 43
4.6 Case Study 45
4.6.1 HERMES Policy Composition 45
4.6.2 Case Study Organizational Hierarchies 45
4.6.3 Usage of HERMES and Subsequent Results 47
4.6.4 Deployment of Final Configurations 55
4.7 Integration into the HiFiPol Framework 56
4.8 Tools Similar to HERMES 57
4.9 Extension of HERMES 59
4.10 Chapter Conclusion 60

5 Future Work **62**

5.1 Organizational Scenario 62

5.2 HiFiPol:Browser v0.1 63

5.3 Conflict Detection and Resolution 63

5.4 Semi-Automatic Policy Instantiation 64

5.5 HERMES v1.0 64

5.6 Configuration Repository 65

6 Thesis Conclusion **66**

References **68**

References **68**

Appendix A: Copyright and Credit Notice **72**

List of Figures

2.1 An sample organizational hierarchy for Devices and Roles/Users. 13
2.2 The corresponding AD DS hierarchy for the sample organizational structure in Figure 2.1. 13
2.3 An example of (`New Organizational Unit (OU)`) and "Group Policy Object" list (`New GPO List`). 16
2.4 The "`User Configuration`" group-policy tree. 21

3.1 HiFiPol:Browser architecture 28
3.2 An Illustration of Acme Organizational structure: (Left) Domain/Devices ; (Middle-Left) Applications ; (Middle-Right) Users/Roles ; (Right) Actions. 36

4.1 HiFiPol:Browser subsystems 40
4.2 Organizational hierarchy graph. 46
4.3 Application hierarchy graph. 46
4.4 User-Group hierarchy graph. 46
4.5 Possible-Actions hierarchy graph. 47
4.6 Proposed repository structure for configuration files deployment to agents through HiFiPol:Browser architecture design. 55
4.7 HiFiPol:Browser development subsystem 57

6.1 Permission from IEEE to reproduce an article as Chapter 2 of this thesis. 73
6.2 Permission from IEEE to reproduce an article as Chapter 3 of this thesis. 74

CHAPTER 1

Introduction, Contributions, and Overview

In this beginning chapter, I firstly introduce the terms *policy-oriented configurations*, *high-granularity*, *remote configuration*, and briefly introduce the need for policy-oriented, and high-granularity remote configuration tools. Secondly, I describe some of the current problems associated with development of granular and policy-oriented remote web browser configuration tools; here, I also discuss some possible solutions. Thirdly, I portray the contributions of this thesis. Fourthly, I present a list of coauthored refereed publications. Fifthly, I make notice of using previously published materials, which are copyrighted by the Institute of Electrical and Electronic Engineers, Inc. (IEEE). Lastly, in this introduction, I provide an overview of this thesis document.

1.1 Introduction

Every system or tool has some form of setting; which is variable and which can be changed by users and/or administrator of the system or tool. Such variable settings are called "configurations". In contrast, the set of rules, operational or technical, which are specified by the system administrator for a system or tool; are known as "policies". Thus, *policy-oriented configuration* is the process of automatically 'configuring' system or tools through 'policies' rather than manual configuration by system administrators and/or users.

Every system or tool must provide a mechanism which allows for configurations to be deployed at a per user-, per application-, per domain-, and per device-, level; for every action possible in that system or tool. Provision of such a mechanism is what we define as *high-granularity*.

In a typical domain-network environment, where one or more 'client(s)' are connected to a server, called as 'domain controller'; the usual practice of system administration is to administer the applications/tools of all 'client' machine(s) from the domain controller.

Thus, *remote configuration* is the process of 'configuring' any/all client systems and/or client applications through the domain controller; instead of the system administrator having to configure each client system/application individually.

In the contemporary world of technological sophistication, it is becoming increasingly harder to secure an organization's cyber-infrastructure; due to vulnerabilities arising from improper configuration of tools/applications. Out of several networking tools/application, our particular interest is on web browsers (more details in Chapter 2.1.2)[1]. To help mitigate vulnerabilities arising from improper configuration of web browsers, we argue that policy-oriented (not manual configuration-oriented), and high-granularity remote configuration tools are necessary (more details in Chapters 2.1.3, 1.2, 1.3).

1.2 State of Web Browser Usage

As of November 2015, more than 3.3 billion people, which is about 45% of the world's population were web browser users [2]. Such a tremendous success of web browsers can be credited to several factors. Three of these factors are: Turing-complete execution, World-wide digital network reach, and powerful graphics programmability.

Like most computer applications today, common web browsers are known to have several security vulnerabilities at any given point of time [3]. These vulnerabilities, if exploited, can lead to data and device compromise. In a defense-in-depth approach multiple independent layers of security must be implemented. Designing and implementing security-oriented user/role-, device-, and domain-tailored browser configurations can significantly reduce the attack surface and prevent most active attacks on browsers. For example, configuring separate browsers, or instances, for Intranet and Internet access and selectively enabling JavaScript only for a group of users, on a selected set of devices, connecting to predefined application websites [4].

1.3 Impact of Web Browsers on Society

The web has become an essential tool for the economy, government, and civic society. A survey by RightScale Inc., found that 95% of U.S. organizations, public and private, offer or use cloud-based or web-based services [5]. It is estimated that 46% of the world population use the web today and this percentage is growing [2]. As such, web security has become extremely important. However, recent security breaches, which have resulted in more than a billion compromised records from organizations of all types and sizes, show that web and IT and OT cybersecurity is currently in a precarious state. To contribute towards the effort of securing the web ecosystem is the goal of the work presented in this chapter.

Web browsers are the essential client component of the web ecosystem. Modern web browsers expose powerful virtual machine environments, ubiquitous network access, and permanent storage. Many of the recent high-profile cybersecurity breaches started with spear-phishing of targeted users. Then the attacks progress with the use of URL spoofing, code injection and other attacks[6]. Extensions or plug-ins add another vulnerability vector[7]. Many, if not most, of these web browser-based attack vectors could have been mitigated or eliminated if client web browsers were configured with a high-granularity domain-, device-, user-, role-, application-, and action-based secure configuration. For example, using different web browsers to ensure domain separation, selectively enabling or disabling JavaScript, and restricting access depending upon user and role, device, and application are some ways to configure web browsers for secure browsing. Such a hi-fidelity configuration is not practically possible in today's diverse IT/OT infrastructure with current technology and tools.

In order to enable modern organizations to design and implement highly-specific and security tailored web browser configurations, we argue that an enterprise-wide and policy-oriented, rather than configuration-oriented, web browser configuration management system is needed. We have designed such a system, namely: HiFiPol:Browser[8]. HERMES is an essential and core component of HiFiPol:Browser. HERMES is a high-level security

policy description language that enables system administrators to write security policies that could then be implemented across the IT/OT ecosystem using HiFiPol. HERMES is the contribution described in this chapter.

1.4 The Problem and a Plausible Solution

During our literature review and related work evaluation, we were able to identify three main problems, which have potentially contributed towards lack of progress in development of policy-oriented, and high-granularity remote web browser configuration tools. The three main problems are elaborated in the following paragraphs.

Firstly, there is lack of a centralized database; which houses tutorials on how to remotely configure web browsers. There are several resources available - but they are either: not-easy to access or scattered in different places, without an easy to navigate index. Thus, it is difficult for a system administrator to access these resources (more details in Chapter 2.1.2). This problem can be potentially solved by composing easy-to-access tutorials and housing them in a centralized database/publication (more details in Chapter 2.1.3).

Secondly, there is lack of an architecture/model; which presents a design, specifying the components (and interactions between the components) required to achieve the functionality of a policy-oriented, and high-granularity remote web browser configuration tool (more details in Chapter 1.2). Thus, it is difficult for software developers to develop such a tool. This problem can be potentially solved by designing a modular architecture, which allows for evolutionary addition of modules from different parties (more details in Chapter 3.3).

Thirdly, there is lack of a high-level, easy-to-use, and a platform-/application-independent policy specification language for web browsers; which enables a system administrator to write web browser related policies and transforms the policies into configurations (more details in Chapter 4.2 and 4.3). Thus, it is difficult for system administrators to specify policies for all web-browsers and automatically have them transformed into configurations. This problem can be potentially solved by designing a modular language; which allows a system

administrator to write a high-level, easy-to-use, and reconfigurable machine environment specification (more details in Chapter 4.4).

1.5 The Contributions of this Thesis

The approach of this thesis work, is to use a theoretical (mathematical and software engineering theory) approach, so as to achieve its goal; which is: enabling the society to move towards a better secured web browsing operational environment. In the process of analyzing the problems and possible solutions, and in the process of applying our approach in achieving our goal; we have produced three contributions. We elaborately describe these three contributions, as follows:

1. We make an argument that policy-oriented, and high-granularity remote web browser configuration tools have the potential to protect against most of web browser based attacks. In addition, to help contemporary system administrators, we describe a step-by-step walk-through that can be used to remotely configure two web browsers (Internet Explorer and Google Chrome) for enhanced client security. We do this by: a) presenting a sample case study, and b) demonstrate the process of translating the policies of case study into actual configurations. Chapter 2 primarily deals with discussion of this contribution.

2. We present an elaborate architecture design of HiFiPol:Browser - a policy-oriented, and high-granularity remote web browser management system. We do this by: a) presenting an architectural design diagram, and b) explaining the intended functionality of each architectural component and interactions between the architectural components. Chapter 3 primarily deals with discussion of this contribution (HiFiPol:Browser).

3. We make an argument that a high-level, easy-to-use, and a platform-/application-independent policy specification language for web browsers is needed. In addition, to help fulfill the afore-argued necessity, we have created a prototype (v0.1) of aforementioned specification language; called HERMES. HERMES stands for High-level,

Easy-to-use, and Reconfigurable Machine Environment Specification language. Chapter 4 primarily deals with discussion of this contribution (HERMES).

1.6 Author's Related Refereed Publications

In the process of conducting graduate research and obtaining a 'Master of Science' degree; the dissemination of research knowledge - through utilization of well-recognized and peer-reviewed publications is a widely accepted need. As such, during my masters' study, in co-operation with colleagues; I have produced and developed information, that enhances our scientific knowledge and software engineering theory in two heavily-affiliated areas: safety and security.

Two of the contributions presented in this thesis have already been disseminated to scientific and technological community through conference article publications. One contribution has been prepared of publication and is currently under revision.

In this section, I list three bibliographic entries in the order of their research sensibility. Of these four entries, entry number 2 corresponds to 'Contribution 1' (see section 1.5), which is under review. Entries 3 and 4 correspond to 'Contribution 2' and 'Contribution 3' respectively (see section 1.5).

1. Ananth A. Jillepalli, and Daniel Conte de Leon, "Hardening the Client-side: A Guide to Enterprise-level Hardening of Web Browsers in Windows", ©Authors. (Chapter 2).

2. Ananth A. Jillepalli, and Daniel Conte de Leon, "An Architecture for a Policy-Oriented Web Browser Management System: HiFiPol: Browser", In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, vol. 2, pp. 382-387. IEEE, 2016. DOI: 10.1109/COMPSAC.2016.50. ©2016 IEEE [8]. (Chapter 3).

3. Ananth A. Jillepalli, Daniel Conte de Leon, Stuart Steiner, and Frederick T. Sheldon, "HERMES: A high-level policy language for high-granularity enterprise-wide secure browser configuration management", *Computational Intelligence (SSCI) 2016 IEEE*

Symposium Series on, pp. 1-9, 2016. DOI: 10.1109/SSCI.2016.7849914. ©2016 IEEE [9]. (Chapter 4).

1.7 Overview of this Thesis

The remainder of this thesis is organized as follows: Chapter 2 - *Hardening the Client-side: A Guide to Enterprise-level Hardening of Web Browsers in Windows*, Chapter 3 - *An Architecture for a Policy-Oriented Web Browser Management System: HiFiPol:Browser*, and Chapter 4 - *HERMES: A High-Level Policy Language for High-Granularity Enterprise-wide Secure Browser Configuration Management* correspond to contributions numbered 1, 2, and 3; respectively, as described in Section 1.5. Chapters 2, 3, and 4 are self-sufficient - with introductions and conclusions of their own.

Chapter 5 presents a list of future work to be accomplished in the direction of this thesis' research. Chapter 6 provides a conclusion to this thesis as a whole. This thesis also includes a bibliography, which is comprised of a complete list of references; organized in order of their citation's occurrence.

CHAPTER 2

Hardening the Client-side: A Guide to Enterprise-level Hardening of Web Browsers in Windows

2.1 Introduction

Today, web browsers are a major avenue for cyber-compromise and data breaches. Web browser hardening, through high-granularity and tailored configurations, can help prevent or mitigate many of these attack avenues. For example, on a classic client desktop infrastructure, an enforced configuration that enables users to use one browser to connect to critical and trusted websites and a different browser for untrusted sites, with the former restricted to trusted sites and the latter with JavaScript and Plugins disabled by default, may help prevent most JavaScript and Plugin-based attacks to critical enterprise sites. However, most organizations, today, still allow web browsers to run with their default configurations and allow users to use the same browser to connect to trusted and untrusted sites alike. In this tutorial chapter, we present detailed steps for remotely hardening multiple web browsers in a Windows-based enterprise, for Internet Explorer, Google Chrome, and Edge. We hope that system administrators use this guide to jump-start an enterprise-wide strategy for implementing high-granularity application-level hardening. This will help secure enterprise systems at the front-end, in addition to the network and back-end sides.

2.1.1 Paradigm Shift

Today, the web-based model, protocols, and applications, have penetrated all aspects of enterprise information technology (IT) systems. In addition, organizations are moving legacy and operational technology (OT) systems toward this model due to its flexibility and economic advantages. Furthermore, most cloud-based services use the same browser-based access model for client access and system administration. This migration has resulted in web browsers being used today for accessing critical and private enterprise data and sys-

tems, including Industrial Control Systems, while, at the same time, synchronously or asynchronously, the same web browser, being used to access untrusted sites and browse the Web at large.

2.1.2 Chapter's Problem and Threat Model

According to Verizon's 2016 Data Breach Investigations Report (DBIR) [1], the *Web App Attacks* pattern was used in 40% of the reported data breaches in the previous year; Though, the same pattern accounted for less than 10% of the reported incidents (5,334 incidents (n=64,199) and 908 breaches (n=2,260) for the pattern [1] pp. 22-23). This data indicates that the *Web App Attacks* pattern last year resulted in a high likeliness of breach. In addition, in a presentation given at the USENIX Enigma 2016 Conference, Mr. Rob Joyce, Chief of the Tailored Access Operations Office at the U.S. National Security Agency, pointed out that, today, most intrusions are carried out through one of the these three initial vectors: 1) Email (including Phishing), 2) Malicious website, and 3) Malicious removable media [10]. Mr. Joyce has been, very recently, nominated for the position of White House Cybersecurity Coordinator [11].

Under the, current and common, application usage and default client-side application configuration, attackers have direct and remote access to the same client application that is used to access critical enterprise sites, usually the user's, and even system administrator's, preferred web browser. This can be accomplished by malicious actors through simple web drive-by, phishing, or any other URL sharing method that can lead users to a malicious site. Even the highest levels of network segmentation and server-side hardening cannot adequately protect against data breaches when the same web browser (or other client-side application) is being used to access critical services and also browsing the Web at large, when the same set of permissive security configurations are used for all sites, trusted and untrusted.

The web browser is, inarguably, the weakest link on the enterprise today. However, most well-known and commonly used desktop web browser applications are designed and developed

by teams with cybersecurity expertise and the best secure development processes in industry. These applications are also fixed and updated on a continuous basis. The problem resides on organization's one-size-fits-all approach to security configurations for web browsers and the resulting use of default and permissive security configurations, especially for JavaScript and Add-Ons. To solve this problem, high-granularity and organization-tailored configuration of web browsers is much needed.

2.1.3 Chapter's Contribution

We argue that configuring client Web Browsers using a white-listing approach and strictly enforcing the principle of least privilege can protect against most of these attacks. In order to implement such tailored and high-granularity configurations at an enterprise scale and in a sustainable way, policy-based and high-granularity, remote policy enforcement tools are needed. In previous publications [8, 9], we describe how policy-level and high-granularity application-level hardening would be possible. Most current approaches are configuration-level and not policy-level.

In this chapter we present a different contribution intended to help system administrators today. We describe a step-by-step walk-through that can be used by system administrators to remotely configure two web browsers for enhanced client security. One web browser, Internet Explorer, will be configured to enable all browser functionality but for trusted internal sites only. A second browser, Google Chrome, will be configured with high security configurations to enable safer browsing of the Web at large. The latter will be used as default browser but not to access internal trusted sites. In this tutorial, all steps needed to implement these configurations are detailed using Microsoft's ® Active Directory Domain Services, Group Policy environment and tools.

2.1.4 Outline of this Chapter

The rest of this chapter is organized as follows: Section 2.2 introduces the terminology, tools, and technologies involved. Section 2.3 discusses the process of translation between

an organization's hierarchical structure and the Active Directory Domain Services (AD DS) hierarchy. This section describes the needed steps for creating groups that will enable remote configuration of a set of clients and the remote installation of Google Chrome. Section 2.4 describes all the steps needed to implement the proposed enhanced web browser configurations. Conclusions, Acknowledgments, and References follow.

2.2 Preliminaries

In this section we introduce the services, concepts, and nomenclature needed to understand Microsoft®'s Active Directory Domain Services (AD DS) and be able to remotely manage Windows users and computers in an enterprise. This knowledge is necessary to design and implement remote configurations on Web Browsers in managed Windows clients.

2.2.1 Active Directory Domain Services

Active Directory (AD) is Microsoft®'s brand name for a suite of remote administration and configuration tools. AD includes a wide range of directory, identity, and remote management services. A Windows Server that is used to remotely control and configure a group of clients is called a Domain Controller, when AD services are added as a server role the term Active Directory Domain Services is used (AD DS)[12]. AD DS enables administrators to organize enterprise assets and objects in a hierarchical structure. Managed objects are users and computers. These devices are organized in a tree-like fashion. The root of the tree is called and AD Forest. At the second level are Domain Controllers (Servers). Multiple Domain Controllers can manage and replicate the organization's structure and group configurations in whole or in part and in a distributed and fault-tolerant fashion. At the third level are Organizational Units (OUs). At the fourth-level are Users and Computers which are attached to OUs. Then configurations (called Group Policy Objects) are applied to OUs. This way a give set of configurations can be remotely enforced in all Users and Computers in the respective OU.

2.2.2 Group Policy: Objects and Management

Group Policy Objects (GPOs) are sets of per-system-, per-user-, or per-application-specific configurations and their instance value. *Group Policy Management Console* (GPMC) is the tool used to create and edit GPOs based on available templates (ADMX/ADML). GPM uses the information in the GPOs to populate Windows Registry keys and values on remote clients and make the configurations effective. Only systems and applications that support configuration through Microsoft's Group Policy approach, and that use the Windows Registry for application configuration, can be configured this way. However, most popular desktop Web Browsers can be configured using AD DS and GPO.

Our focus in this chapter are Web Browsers. Specifically, Internet Explorer, Google Chrome, and Microsoft Edge. Firefox does not use the Windows Registry to store configurations. The authors are developing a policy-oriented and cross-platform solution to address this issue [8, 9, 13].

2.2.3 Environment Setup

The activities described in this tutorial were performed on a single workstation, using three virtual machines (VM). One Windows Server 2016 with AD DS role VM. Two Windows 10 Education edition VM, referred as *Clients* or *Client 1* and *Client 2*). All three virtual machines were attached to the AD DS Server.

2.3 Organizational Structure

In AD DS an organization's departmental layout must be translated into groups called `Organizational Units` (OUs). An OU is an AD DS entity that groups users and/or computers (machines) [14]. Let us assume that an organization has two departments. These departments can be respectively modeled in AD DS as OUs and relevant users and machines can be attached to a particular OU.

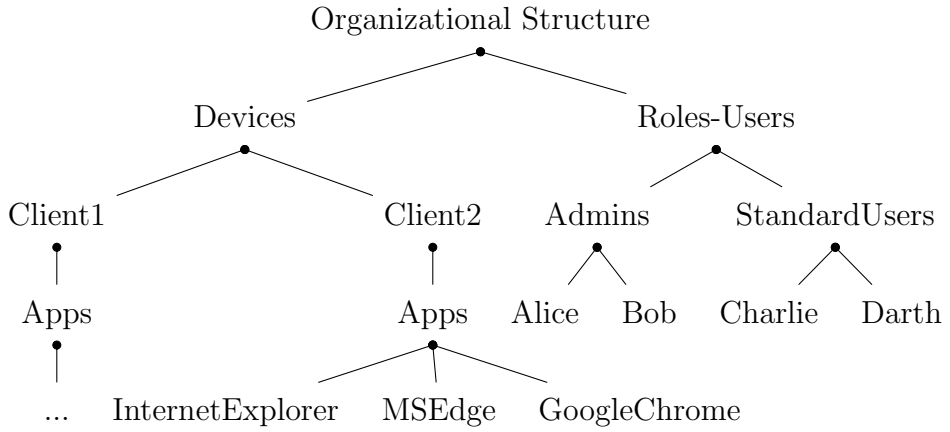

Figure 2.1: An sample organizational hierarchy for Devices and Roles/Users.

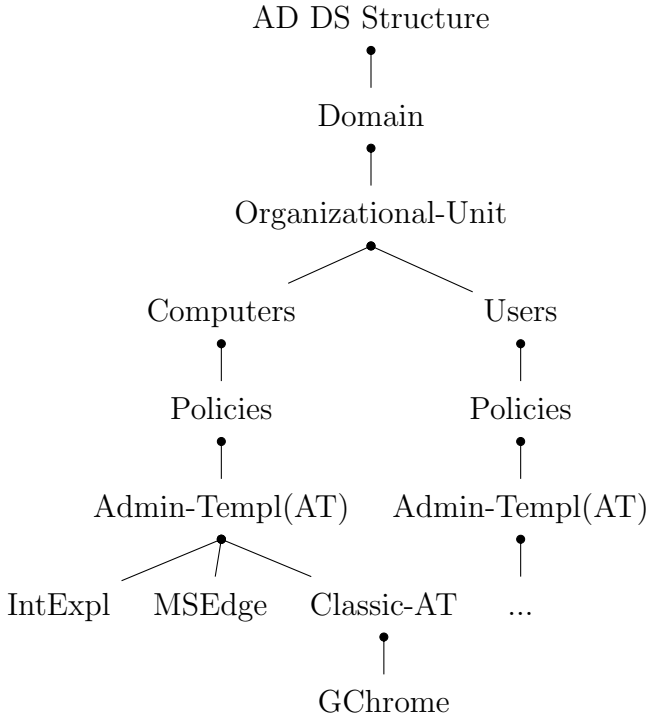

Figure 2.2: The corresponding AD DS hierarchy for the sample organizational structure in Figure 2.1.

2.3.1 Setting up Organizational Units

Here we details the steps to create an organizational unit in AD DS.

1. On the Server, open `Server Manager` suite, if it is not automatically opened by default on startup.

2. Click on the `Tools` menu, in the top-right part of `Server Manager` suite's `Dashboard` screen.

3. In the `Tools` drop-down menu, click on the `Group Policy Management` selection. This action should produce `Group Policy Management` interface box.

4. In left pane of the `Group Policy Management` interface box, please identify the target forest and subsequently, the target domain.

5. Right-click on the target domain name and in the pop-up menu, select `New Organizational Unit`. In the subsequent dialogue-box, type-in the intended name of the OU to be created.

6. As a result, the OU is created. To verify this creation, one can find the new OU within the domain tree (Figure 2.3).

2.3.2 Attaching Users and Computers to OUs

Once an OU is created, objects (groups of users and/or computers) can be attached to the organizational unit. Here, we detail the steps to attach objects (users or computers) to an OU.

1. On the Server, open `Server Manager`, if it is not automatically opened by default on startup.

2. Click on the `Tools` menu, in the top-right part of `Server Manager` suite's `Dashboard` screen.

3. In the `Tools` drop-down menu, click on the `Active Directory Users and Computers` selection. This action should produce `Active Directory Users and Computers` interface box.

4. In this interface box, search for the target domain name tree, on the left pane of the interface box. Once identified, expand the target domain name tree.

5. Under the target domain name tree, identify the target object, which would be stored in its' respective folder. For example, if the target object is a user named `Joe`, it can be found in the `Users` folder.

6. Once the target object has been identified, right-click on the object to activate a menu pop-up. In this menu, select the item `Move...`. In the subsequent `Move` pop-up box, select the desired OU as the destination to move the object to, then click `OK`.

7. The target object has now been successfully attached to the desired OU. To verify, in the `Active Directory Users and Computers` interface box, search for the target domain name tree, on the left pane of the interface box. Once identified, expand the target domain name tree.

8. Under the target domain name tree, search for the desired OU folder. Click on it. Press the `F5` key on the keyboard or click on the `Refresh` icon in the top tool-bar. If the target object is visible in the center pane of the desired OU, we can consider the attachment to be verified.

Creating a GPO and linking it to a desired OU

Once an Organizational Unit (OU) is created, and once all the desired individual users or groups of users and computers have been attached to the OU; for implementing group policies, we have to create a Group Policy Object (GPO) list and link it to the target OU. Here, we detail the steps to create a GPO and link it to the desired OU.

1. In the `Group Policy Management` interface box, identify the target OU. Once found, right-click on it.

2. In the resultant pop-up menu, click on the `Create a GPO in this domain and Link it here...` selection.

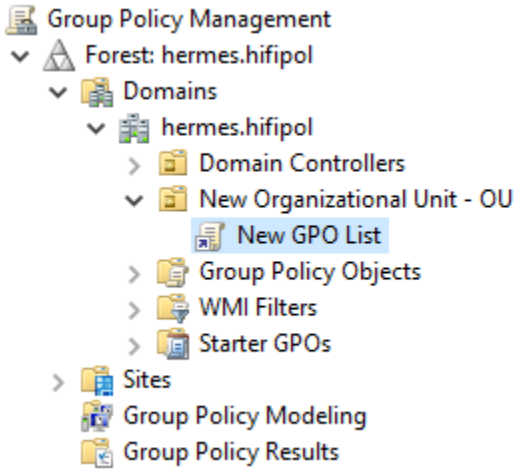

Figure 2.3: An example of (`New Organizational Unit (OU)`) and "Group Policy Object" list (`New GPO List`).

3. In the `NEW GPO` pop-up box, enter the desired name for the GPO list to be created and click on the `OK` button. Before clicking the `OK` button, one may want to select a `Source Starter GPO`, if one already has a GPO list to be inherited.

2.3.3 Remote Installation of Google Chrome

Since Google Chrome is not installed natively on Windows OS, the following steps specify how to remotely install Google Chrome.

1. Download Google Chrome installer, depending on client processor architecture [15].

2. Place the installer in a network-drive or in a location, which can be read by all the target domain clients.

3. In the `Group Policy Management` interface box, identify the target GPO list. Once found, right-click on it. In the resultant pop-up menu, click on `Edit...` option.

4. In the subsequent `Group Policy Management Edit` interface, locate the `Computer Configuration` tree, which can be found in the left pane of the interface. Once located, expand the `Policies` tree, located under `Computer Configuration`.

5. In the subsequent expanded tree view, a folder with name of `Software Settings` should be visible. Double-click on the folder. Subsequently, a folder named `Software installation` should be visible. Right-click on it.

6. In the resultant pop-up menu, hover mouse pointer on the `New` selection. A sub-menu item named `Package...` should pop up. Click on it. A directory browser will pop-up, as a consequence. Navigate/browse to the location of the Google Chrome installer. Select the installer and click on the `Open` button.

7. Subsequently, the `Deploy Software` dialogue box will pop-up. The default selection in this box should be `Assigned` radio button. If not, change it to `Assigned` and click on `OK`.

8. All systems connected to the domain will run the installer upon restart or log-on of domain users. Alternatively, a system administrator can also right-click on a target OU and in the resultant pop-up menu, select `Group Policy Update...` option to execute the installation right away.

2.3.4 Importing Administrative Templates for Google Chrome and Microsoft Edge

Group Policies for Google Chrome and Microsoft Edge are also not available on vanilla installations of Windows Server 2016. We have to download administrative template packages and import them to incorporate Google Chrome and Microsoft Edge Group Policies.

Importing a Configuration Template: Google Chrome:

1. Download the ADM package for Google Chrome [16]. Extract the template packages to a location of preference.

2. In the `Group Policy Management` interface box, identify the target GPO list. Once found, right-click on it. In the resultant pop-up menu, click on `Edit...` option.

3. In the subsequent `Group Policy Management Edit` interface, locate the `Computer Configuration` tree, which can be found in the left pane of the interface. Once located, expand the `Policies` tree, located under `Computer Configuration`.

4. In the subsequent expanded tree view, a folder with name of `Administrative Templates` should be visible. Right-click on the folder. In the resultant pop-up menu, click on `Add/Remove Templates...` option.

5. In the subsequent `Add/Remove Templates` pop-up, click on `Add...` button. A directory browser will pop-up, as a consequence. Navigate/browse to the location of extraction for Google Chrome administrative templates package. Within the extracted package folder, navigate to the preferred locale folder, example: `en-US`.

6. A file named `chrome.adm` (filename verified as of 02/28/2017) will be found within any locale folder. Select the file and click on `Open` button, which can be found at the bottom of navigation window.

7. Back in `Add/Remove Templates` pop-up, click on `Close` button. As a result, a new folder with the name `Classic Administrative Templates (ADM)` will be added, within `Administrative Templates` folder. Double-click on this newly added folder.

8. Double-click on the resultant `Google` folder and the two folders containing configuration policies for Google Chrome web browser can be found in this location. There are two folders, one folder's policies can be overridden by standard users and the other folder's cannot be overridden by standard users.

Importing a Configuration Template: Microsoft Edge:

1. Download the administrative template installer for Microsoft Edge [17]. Install the template package to a location of preference.

2. By default, the package is installed at *C:\Program Files\ Microsoft Group Policy\Windows 10 and Windows Server 2016 (version 2.0)\ PolicyDefinitions*, for a 64-bit OS (default path verified as of 02/28/2017). Let us call this path as *Default Install* pathname.

3. Navigate to the `Default Install` pathname's `Policy Definitions` folder and within this folder, identify the file with name `MicrosoftEdge.admx` [18]. Once found, copy the file to *C:\Windows\PolicyDefinitions*.

4. Navigate back to the `Default Install` pathname's `Policy Definitions` folder and within this folder, open the folder matching the OS's locale. In our case, folder name will be `en-US`. Within the `en-US` folder, identify the file with name `MicrosoftEdge.adml`. Once found, copy the file to *C:\Windows\PolicyDefinitions\en-US*, or as corresponds to the OS locale.

5. In the `Group Policy Management` dialog box, identify the target GPO list. Once found, right-click on it. In the resultant pop-up menu, click on `Edit...` option.

6. In the subsequent `Group Policy Management Editor` interface, locate the `Computer Configuration` tree, which can be found in the left pane of the interface. Once located, expand the `Policies` tree, located under `Computer Configuration`.

7. In the subsequent expanded tree view, a tree with name of`Administrative Templates` should be visible. Expand it. In the subsequent expanded tree view, another tree with the name of `Windows Components` should be visible. Expand it. In the subsequent expanded tree view, a folder with the name of `Microsoft Edge` should be visible. Open it (by double-clicking on the folder name).

8. If the left side of bottom most bar of the `Group Policy Management Editor` shows `20 setting(s)` (as of 03/03/2017), and the policy settings are visible in the central

pane of `Group Policy Management Editor` interface, then the import process can be considered as successful.

Therefore, this section details the process of successfully "translating" an organizational hierarchy into AD DS. To summarize the process, the following four steps are required for the "translation": 1) Create OUs as necessary, 2) Attach users/computers to respective OU, 3) Create GPO lists, and 4) Link respective GPO list to the relevant OU, as per organizational policy requirement.

2.4 Hardening Web Browsers

Once the organizational hierarchy is translated into AD DS, as discussed in Section 2.3, we can edit the GPO list settings for the respective OUs, to change configuration of domain system applications. In this section, we will go through the process of remotely configuring web browsers (more information on web browser selection in Sub-Section 2.2.2), by changing group policy settings.

2.4.1 Configuration Template Locations

Within group management tree structure, the default location of web browser's policy settings; for domain users, are as follows:

Internet Explorer:

1. In the `Group Policy Management` dialog box, identify the target GPO list. Once found, right-click on it. In the resultant pop-up menu, click `Edit...`.

2. In the `Group Policy Management Edit` dialog, locate the `User Configuration` tree (figure 2.4), which can be found in the left pane of the interface. Once located, expand the `Policies` tree, located under `User Configuration`.

3. In the expanded `Policies` sub-tree, a folder named `Administrative Templates` should be visible; expand it. In the resulting tree view, a sub-tree with the name of `Windows Components` should be visible; expand it.

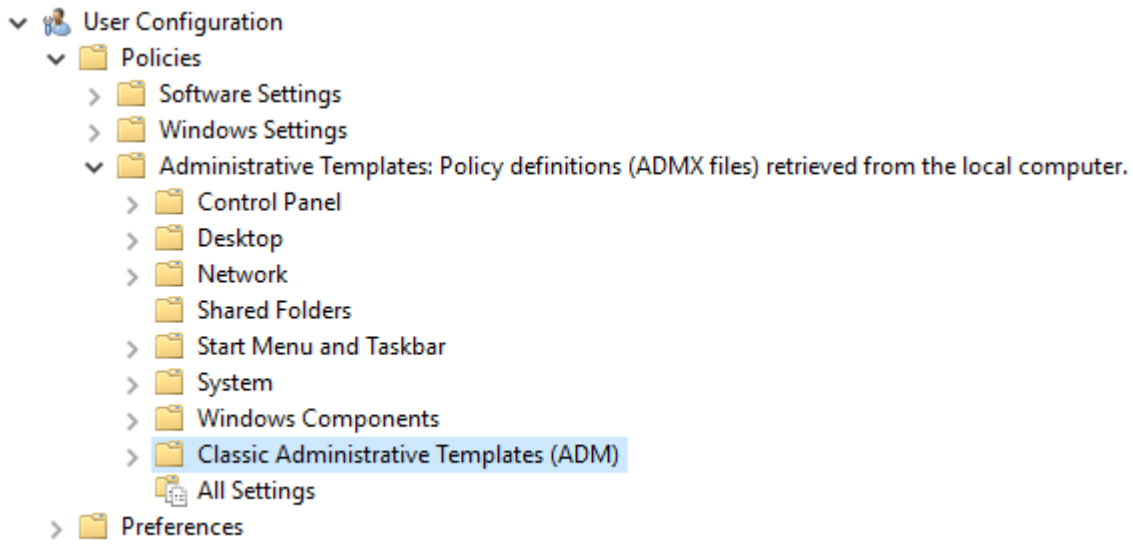

Figure 2.4: The "`User Configuration`" group-policy tree.

4. In the expanded `Windows Components` sub-tree, a folder named `Internet Explorer` should be visible; Open it by double-clicking on the folder name.

Microsoft Edge:

1-3. Same as stated for Internet Explorer in (2.4.1).

4. In the subsequent expanded tree view, a folder with the name of `Microsoft Edge` should be visible. Open it by double-clicking on the folder name.

Google Chrome:

1-2. Same as stated for Internet Explorer (2.4.1).

3. In the subsequent expanded tree view, a tree with name of `Administrative Templates` should be visible. Expand it. In the subsequent expanded tree view, another tree with the name of `Classic Administrative Templates (ADM)` should be visible. Expand it.

4. In the subsequent tree view, a folder named `Google` should be visible; Expand it. Resultantly, there should be two folders visible, with the names `Google Chrome` and `Google Chrome - Default Settings (users can override)`.

5. For this tutorial, we will be focusing on `Google Chrome` folder settings, so that users cannot override organizational configurations. As such, open the `Google Chrome` folder (by double-clicking on the folder name).

2.4.2 Configuration Hardening Examples

Let us assume that chief security officer of an organization called Acme, would like to enforce the following two web browser related policies: 1) Internet Explorer shall be used only for the purpose of connecting to Acme's web domains and not any other web domains. 2) Google Chrome shall be used only for the purpose of connecting to any web domain apart from Acme's web domain. Also, JavaScript functionality in Google Chrome should only be allowed for the web domain `example.org`. Third-party cookie functionality should be completely disabled in Google Chrome. We did not include Microsoft Edge (ME) in this example because group policy settings for ME are very volatile (as of 03/21/2017) and are being changed every couple of months by Microsoft.

There are no group policies, both for Internet Explorer and Google Chrome, to block access to a set of websites (as of 03/21/2017). As such, in the following walk-through, we will be blocking all functionality for websites which are intended to be accessed with a particular web browser.

Internet Explorer::

1. Navigate to the location of Internet Explorer policies (Sub-Section 2.4.1). In the `Internet Explorer` folder, search for a folder `Internet Control Panel`. Once found, double-click on it. Search for a folder `Security Page`.

2. In the `Security Page` folder, search for a policy setting `Site to Zone Assignment List`. Once found, double-click on it. In the resultant box, select the radio button

`Enabled`. Doing so will activate the `Show...`-labeled button under `Options` pane, beside the string `Enter the zone assignments here`. Click on the `Show...` button.

3. A `Show Contents` box will pop-up. In this box, under the `Value name` column, type-in Acme's web domain - for example: `www.acme.org` and under the `Value` column, type-in the number `1`. Then click `OK`. This assigns the web domain `www.acme.org` and all its' sub-domains to the `Intranet` zone of Internet Explorer.

4. Back in `Security Page` folder, search for a policy setting `Internet Zone Template`. Once found, double-click on it. In the resultant box, select the radio button `Enabled`. Doing so will activate a drop-down menu under `Options` pane, beside the string `Internet`. Select the option `High` from the drop-down menu. Then click on the `OK` button.

5. Back in `Security Page` folder, search for a folder `Internet Zone`. Once found, double-click on it. Inside `Internet Zone` folder, click on `Setting` coloumn in the top of middle pane. Doing so will categorize all policy settings into ascending order. Set all policy settings starting with the characters `A`, `D`, and `J`, to the value `Disable`. Doing so will essentially render all web domains (excluding Acme's) non-functional apart from simple text-based web-pages.

Google Chrome::

1. Navigate to the default location of Google Chrome policies (Sub-Section 2.4.1). In `Google Chrome` folder, search for a folder named `Content Settings`. Double-click on it. Now, search for a policy setting named `Allow JavaScript on these sites`. Once found, double-click on it.

2. In the resultant box, select the radio button `Enabled`. Doing so will activate the `Show...`-labeled button under `Options` pane, beside the string `Allow JavaScript on these sites`. Click on the `Show...` button.

3. A `Show Contents` box will pop-up. In this box, under the `Value` column, type the domain name pattern `[*.]example`
   `.org` to white-list JavaScript functionality for all subdomains of `example.org/`. Then click `OK`.

4. Back in `Content` folder, search for a policy setting named `Block cookies on these sites`. Once found, double-click on it. In the resultant box, select the radio button `Enabled`. Doing so will activate the `Show...`-labeled button under `Options` pane, beside the string `Block cookies on these sites`. Click on the `Show...` button. A `Show Contents` box will pop-up. In this box, under the `Value` column, type the organizational web domain name pattern, for example - `[*.]acme.org`.

5. Repeat the above step for all policy settings starting with character `B` (as of 03/21/2017) in the `Content` folder. Doing so will essentially render organizational web domains (e.g: Acme's) non-functional apart from text-based web-pages.

6. Back in `Google Chrome` folder, search for a policy setting named `Block third party cookies`. Once found, double-click on it. In the resultant box, select the radio button `Enabled`. Then click `OK`. This policy setting will disable third party cookies across all we domains.

All the modified configuration-policy settings will be in effect upon restart or re-log of domain user accounts. Alternatively, system administrators can force group-policy update for all entities of an OU; by right-clicking on the OU in `Group Policy Management` dialog box and in the resultant pop-up menu, select the option `Group Policy Update...`. Provide confirmation in the resultant `Force Group Policy update` box, by clicking on the `Yes` button.

2.5 Chapter Conclusion

The walled-castle approach to protecting the enteprise has been successful at preventing most

cyber attacks that are launched from the outside. Today, from an attacker's point-of-view, the path of least resistance, is tricking users to launch the attack from within the security perimeter. Time has come to secure the enterprise at the device- and application-levels.

In this tutorial, we describe how to use Microsoft's Active Directory Domain Services and Group Policy to remotely configure Internet Explorer, Google Chrome, and Microsoft Edge web browsers. AD DS is the most used system in the enterpirse for configuring and managing Windows-based clients. We hope that organizations using AD DS make it a priority to begin configuring their client browsing infrastructure with organization-tailored secure configurations. This will help mitigate the problem described in Subsection 2.1.2. In related publications the authors describe a policy-oriented approach, language (HERMES), and tool-set (HiFiPol) for high-granularity, device-, user-, and application-specific, and platform-independent secure application configurations [8, 9, 13].

CHAPTER 3

An Architecture for a Policy-Oriented Web Browser Management System: HiFiPol:Browser

3.1 Introduction

Web browsers are a necessity of today's economy and government. This success is attributed to their flexibility, which is afforded by Turing-complete execution and powerful graphic capabilities, both accessible through the network to trusted and untrusted sites. These capabilities, if maliciously undermined, have high potential for data or system compromise. An approach that can be successfully applied to prevent and mitigate compromise is tailoring browser security settings according to device, user/role, and domain. To make such a high-fidelity security configurations practical, we are designing and implementing HiFiPol: Browser: a policy-oriented and multi-platform Hi-Fidelity security Policy management system for web Browsers. In this chapter, we describe the architecture of HiFiPol: Browser. We describe in detail all components of the architecture, the tasks needed to implement it in a fully operational system, and the current status on the progress of each task. HiFiPol: Browser has been designed to provide: a) a human-friendly and high-level policy specification language and environment, b) security policy conflict detection and resolution, c) automatic instantiation of high-level policies into configurations, and d) distributed browser configuration deployment. We believe that HiFiPol: Browser will enable the design and implementation of domain-, application-, device-, and user- tailored secure policies within a technically diverse organization.

Currently, it is an extremely difficult task, for most organizations, to develop one set of high-level secure browser policies, instantiate this set into a hi-fidelity user-, device-, and domain-tailored, configuration, and deploy resulting configurations into individual client devices [19]. After analyzing browser configuration options, V.A. Bhandari found that out of

approximately 3000 configuration options configurable in all three major browsers, Internet Explorer, Google Chrome, and Mozilla Firefox, only 17 configurations had common names and most had distinct semantics [20].

In this chapter, we begin to address this problem by describing the architecture of HiFiPol:Browser, a High-Fidelity security Policy management system that is intended to provide a high-level policy specification language and a semi-automatic browser configuration environment.

The rest of this chapter is organized as follows. In Sections 3.3, 3.4, 3.5, 3.6 we describe HiFiPol: Browser's architecture and its components and connections. In Section 3.2 we describe the current state-of-the-practice.

3.2 Existing Tools

Current practice is focused on remote configuration tools. Leading tools are described here.

The leading remote configuration tool is Microsoft (MS) Group Policy (GP) [18]. MS-GP uses administrative templates (ADM) [21] as a way of locally or remotely configuring applications in Windows client systems. ADMX and ADML are XML-based files provided by Microsoft or the application vendor [20].

Remote configuration of most settings in Internet Explorer (IE) may be performed using the GP Management Console and ADMX templates [20]. A certain set of settings in Google Chrome can be configured using ADMX template files [22, 23]. A very reduced set of settings in Firefox can be configured using ADMX but only after installing a third party add-on [24]. In Linux environments, Red Hat's "FreeIPA" [25], and in Apple environments, "Mobile Device Management (MDM)" [26], provides similar platform-dependent and configuration-based environments.

The tools briefly described above are well-suited for configuration deployment. However, these tools offer no help when developing high-level security policies. They also offer no help when instantiating high-level policies into high-fidelity user/role-, device-, and application-

specific low-level configurations.

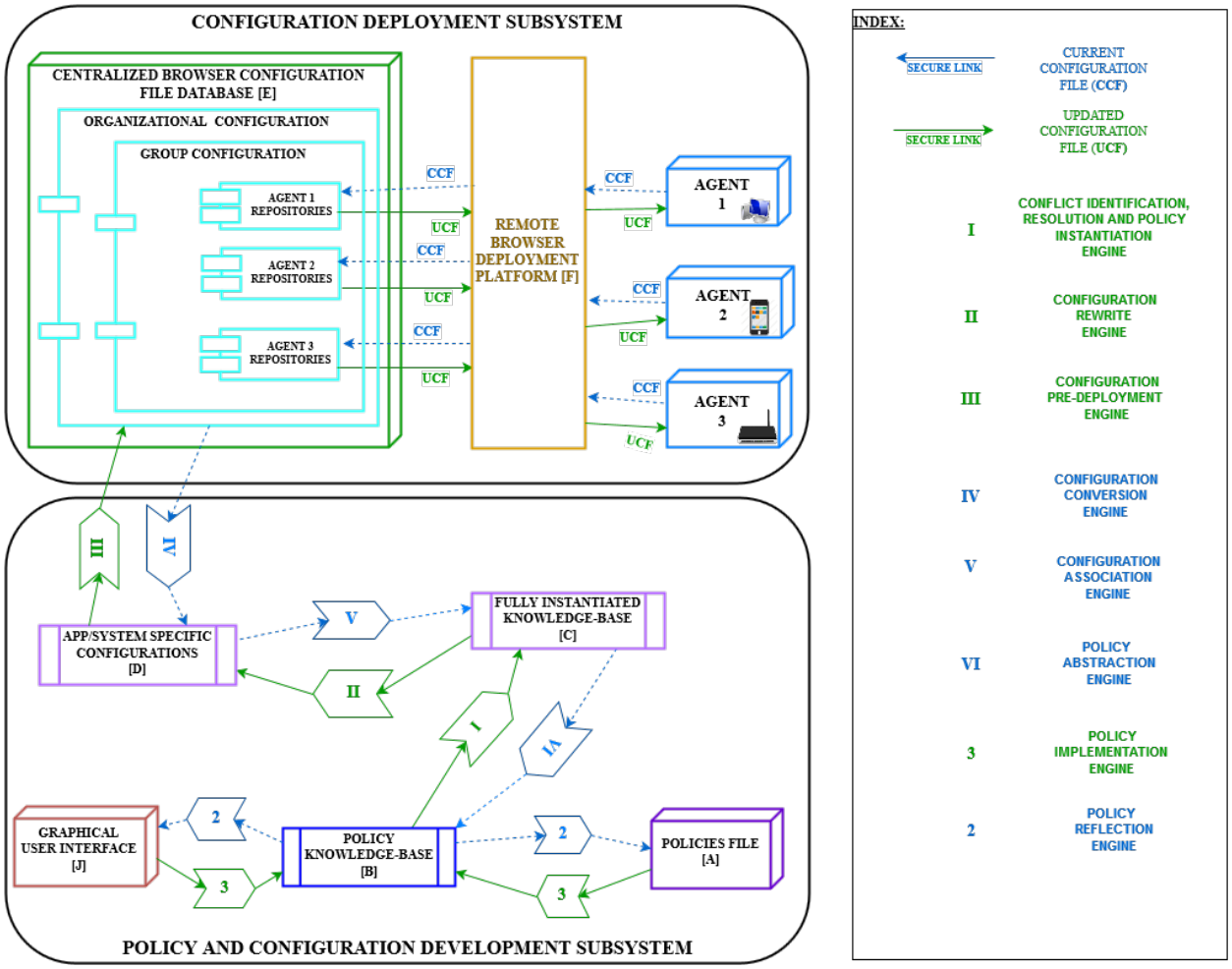

Figure 3.1: HiFiPol:Browser architecture

3.3 HiFiPol:Browser Architecture

The architecture is divided into two subsystems: (A) Policy and Configuration Development and (B) Configuration Deployment. Each subsystem is composed of entities of one of these types: (1) Knowledge Entities, (3) Process Entities, and (3) Agents.

3.4 Knowledge Entities

Knowledge Entities are further divided into three types:

1. *Application-Interactive Entities*

   (a) Remote Browser Deployment Platform

   (b) Centralized Browser Config. File Database

2. *Data Hubs*

   (a) Policy Knowledge-Base

   (b) Fully Instantiated Knowledge-Base

   (c) App/System Specific Configurations

3. *User-Interactive Entities*

   (a) Policy Files

   (b) Graphical User Interface

3.4.1 Remote Browser Deployment Platform

Remote Browser Deployment Platform, shown in 3.1 with label [F], is a set of software components that implements a secure tunnel between the agent systems and the configuration database. This component can be implemented using current tools such as GP, OSSEC, Puppet, Chef, or SSH. We have a preliminary implementation using SSH and plan to move to using Puppet or Chef.

3.4.2 Centralized Browser Configuration File Database

The *Centralized Browser Configuration File Database*, shown in 3.1 with label [E] is a storage component intended to store all fully-instantiated configuration files for each device, user/-role, and application in the organization. These files, containing high-fidelity configurations, are read by the Deployment Platform. Our current implementation of this component is a simply a hierarchical file repository.

- Organizational Config. (Domain and Sub-domain)

- Group Config.

- Agent Repositories

Organizational Config. is the top most tier of hierarchy, emulating real-life corporations. Configurations pushed through this repository interface have the ability to deploy browser configurations for the agent systems all over and in any system of the specified organization, which are covered in the *Domain* and *Sub-domain* definitions of policy in listing 4.2.

Group Config. is the intermediary tier of hierarchy. Usually, a corporate organization has multiple departments. We have emulated the department structure by naming the interface as "Groups". As such, configurations pushed through this repository interface have the ability to deploy browser configurations for all the agent systems categorized under a specific group.

Agent Repositories is the lowest tier of hierarchy. Each agent machine in the organization is assigned a specific repository containing the directories of all the users of the machine and with permissions, such that an Agent can only access its own repository and not the repositories of other agents. This way, our design encourages compartmentalization.

3.4.3 File Transition Processes

In between all the three entities of Central Database, Remote Browser Configuration Platform and Agent systems, a variety of file exchanges will happen. The transactions take place in the form of two distinct processes. They are:

- Current Config. File Process (CCF)
- Updated Config. File Process (UCF)

Current Config. File Process (CCF) is the procedural transfer of browser configuration files from agent machines to the respective agent repository. In the architecture, this process is represented with dashed (blue colored) pointed arrows (as seen in Figure 3.1.

Updated Config. File Process (UCF) is the incremental deployment of new browser configurations instantiated through high-level policies, by transferring a newly generated config.

file from database to agent machines. To enhance the possibility of restoring previous browser configurations and to enable rolling back of states, our architecture's design advocates the backup of every CCF received from agent machines, by time-stamping and storing the CCF in a separate directory, which we named as "Snapshots".

3.4.4 Application/System Specific Configurations

This entity (which can be identified in Fig. 3.1 with the label [D]), is a compositional collection of multiple web browser configurations which have either been derived from CCF processes (available in the Central Database) by processing the content of CCF using Configuration Conversion Engine [IV] process. Or, they have been created from UCF processes by Configuration Rewrite Engine [III]. These are the browser-compatible-language (low-level) configurations.

3.4.5 Fully Instantiated Knowledge-Base

The *Fully Instantiated Knowledge-Base* (which can be identified in Fig. 3.1 with the label [C]), is a comprehensive collection of intermediary-language level browser settings, which have been either been completely instantiated with respect to the specific requirement of policies by *Conflict Identification, Resolution and Policy Instantiation Engine [I]*. Or, they are compiled by Configuration Association Engine [V].

3.4.6 Policy Knowledge-Base

The *Policy Knowledge-Base* (which can be identified in the Fig. 3.1 with the label [B]), is a storehouse of fact-like policy statements, created as a result of Policy Abstraction Engine [VI] or as a result of Policy Implementation Engine [3]. The facts available in configuration knowledge-base are abstracted into higher level policies through execution of Policy Abstraction Engine [VI] process. Or, the high-level policy specifications in "Policy Files" [A] are implemented and turned into facts, to be instantiated further.

3.4.7 Policy Files

Policy Files, also called as "Policies Specification File" (which can be identified in the Fig. 3.1 with the label [A]), is a file or set of files, which houses the highest-level specification policies, produced as a result of either "Policy Reflection Engine" [2] or as written by humans. Our intention behind designing a high level language policy is so as to enable people without much technological knowledge or computer skills, also be capable of issuing configuration changes, while at the same time, reading existing configurations in a human-readable specification language.

In our proposed architecture, a Policy is an easy-to-understand fact or rule reflecting specific browser configuration(s), which can be issued by an administrator/user, who can issue updates or changes to browser configurations of agents/clients without taking into consideration, the agent machine's operating system, type of browser and browser version. During our investigation, we have evaluated XML-based languages and JavaScript-based languages. But, none of them were easily readable by an average human. As such, we proceeded to develop our own language, based on YAML, to be used for writing policy specifications.

3.4.8 Graphical User Interface

Graphical User Interface (GUI), shown with label [F] in Figure 3.1 is an alternative source of interaction between an administrator and our proposed tool. Specification policies are shown in human-readable format with added GUI functionality. Policy specifications drawn from both, *Policy Reflection Engine [2]* and Dynamically composed human-written policies will be available for interaction in this GUI module.

As of now, we designed the GUI, so that it displays the agent machines in a tree hierarchy. In regard to flexibility in configuring browsers and to allow employees to retain their customized configurations, we designed architecture's GUI to have the options below. (GUI has been designed, not developed or implemented)

- *Overwrite*: Deploy centralized configuration, completely overwriting the current settings.
- *Modify*: Retain current configuration for some features and change for other features.
- *Ignore*: Do not make any changes. (Leaving current setting as it is.)

We plan to extend these with the additional options below.

Import: Download configuration file from agent machines to centralized database. This feature is useful to parse and, display in the GUI, the current settings used by agent machines and also to maintain a backup copy of agent machine's configurations.

3.5 Process Entities

3.5.1 Policy Implementation Engine

The *Policy Implementation Engine*, [3] in Figure 3.1, receives high-level policies available in a *Policy Files [A]* and translates them into logic-based policy facts, which are stored in the *Policy Knowledge-Base [B]*. At this time these facts are represented using Prolog facts.

3.5.2 Conflict Identification, Resolution and Policy Instantiation Engine

The *Conflict Identification, Resolution and Policy Instantiation Engine*, [I] in Figure 3.1, instantiates the policy statements in the *Policy Knowledge-Base [B]*. This instantiation is dependent on resolving any inherent conflicts in the policy statement facts. As such, conflict resolution algorithms must be applied to fact-like policy statements before beginning the process of instantiation. The fully instantiated policy statements are in the form of factoids. Factoids are more elaborate and detail-oriented than facts. These expanded factoids are then housed in *Fully Instantiated Knowledge-Base*.

3.5.3 Configuration Rewrite Engine

The *Configuration Rewrite Engine* (which can be identified in the Fig. 3.1 with the Roman

numeral label [II]), in our architecture, is the first transformation engine with the responsibility to transform specification policy content into browser specific configurations. This process engine takes completely instantiated policy factoids from *Fully Instantiated Knowledge-Base* as input and transforms them into low-level configuration depending on the browser specification, through the use of predefined templates. These low-level configurations are then sent to *Application/System Specific Configurations* data hub.

3.5.4 Configuration Pre-Deployment Engine

In our architecture, the *Configuration Pre-Deployment Engine* (which can be identified in the Fig. 3.1 with the Roman numeral label [III]), is the only process engine which handles the responsibilities of segregating, writing configurations to respective configuration files, and distributing browser configuration files from *Application/System Configurations* to respective agent repositories present in *Centralized Browser Configuration File Database*. Subsequently, the browser configurations are then deployed into agent machines through a remote deployment platform, thus fulfilling, the secure and remote deployment of web browser configurations. Since this process is a penultimate deployment stage of browser configurations, we call it is as pre-deployment process. Due to the nature of its functionality, it is classified as one of the two processes, along with [IV], which overlap in both the development and deployment subsystems.

3.5.5 Configuration Conversion Engine

The *Configuration Conversion Engine*, (which can be identified in the Fig. 3.1 with Roman numeral label [IV]), is the process engine in our architecture, which grabs the existing configurations from different, and possibly multiple, agent repositories on User's command and culminates the various configuration files into application specific configurations, matching the pattern language and stored in the context of *App/System Specific Configurations [D]*. Due to the nature of its functionality, it is classified as one of the two processes, along with [III], which overlap in both the development and deployment subsystems.

3.5.6 Configuration Association Engine

In our architecture, the *Configuration Association Engine*, (which can be identified in the Fig. 3.1 with Roman numeral label [V]), is first of the three "reverse" transformation engines. It obtains software specific configurations data from *App/System Specific Configurations [D]* and transforms them into relatively complete instantiated factoids, through the technique of function association. The consequent factoids are then housed in *Fully Instantiated Knowledge-Base [C]*.

3.5.7 Policy Abstraction Engine

The *Policy Abstraction Engine*, (which can be identified in the Fig. 3.1 with Roman numeral label [VI]), is the second of the three "reverse" transformation engines. It obtains completely-instantiated factoids from *Fully Instantiated Knowledge-Base [C]* and transforms them into respectively relevant policy fact statements, through the use of inverse statement mapping technique. The resultant fact statements are then transferred to and housed in *Policy Knowledge-Base [B]*.

3.5.8 Policy Reflection Engine

The *Policy Reflection Engine*, (which can be identified in the Fig. 3.1 with numeral label [2]), is the third and final of the three "reverse" transformation engines. It obtains fact-like policy statements from *Policy Knowledge-Base [B]* and in a final step, transforms the fact-like statements into human-readable policy specifications. These high-level specifications of policies are then transferred either into *Graphical User Interface [J]*, where the policy specifications are modulated into a UI design or the policy specifications are transferred into a *Policy Files [A]*. Here, they are automatically composed into human-written-like policy specifications, through the use of similarity mapping. Subsequently, allowing humans to read the existing configurations of browsers very easily.

The eight Process Engines work in combination with the three Knowledge Entities to

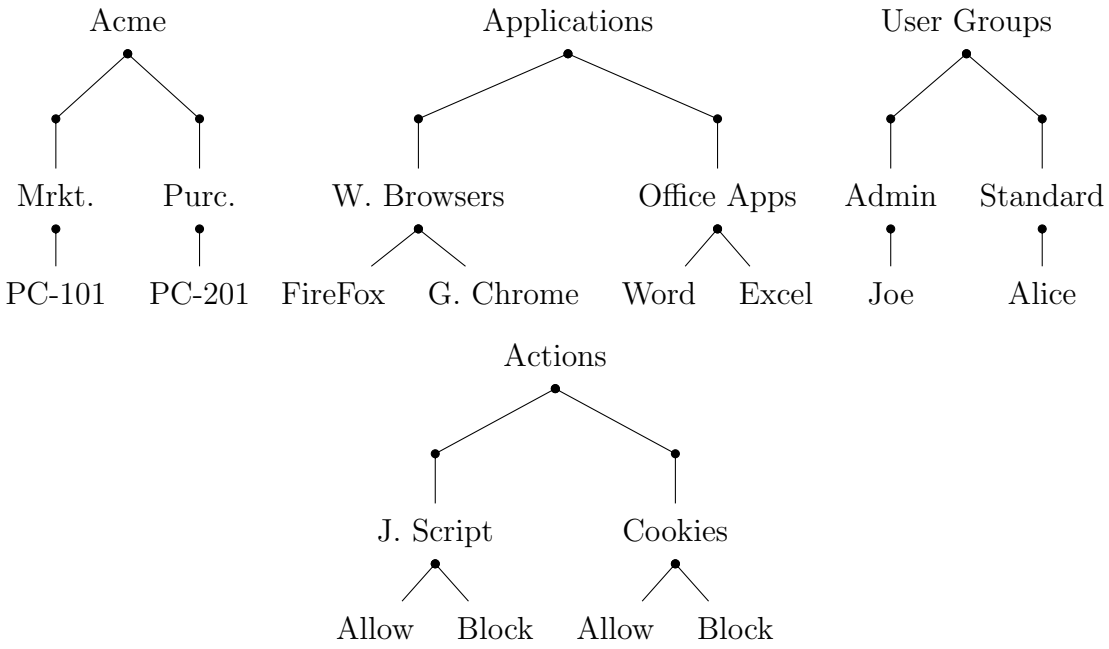

Figure 3.2: An Illustration of Acme Organizational structure: (Left) Domain/Devices ; (Middle-Left) Applications ; (Middle-Right) Users/Roles ; (Right) Actions.

allow the administrator/user of the tool to remotely and securely configure browsers of the agent machines through utilization of high level, human readable policy specification language.

3.6 Agent Entities

We expect our target organization to own or control devices of different types and running a variety of operating systems such as Android, Linux, Mac OS, and Windows (including Windows Phone). We also assume that these devices host multiple applications, including different mobile and desktop web browsers such as Firefox, Google Chrome, Internet Explorer, Chromium, and Opera.

3.7 Chapter Conclusion

In this chapter, we have described the architecture of HiFiPol: Browser and described the status of the research and development work being performed in order to make this management tool a reality. We hope that this endeavor will contribute to solving the problem of

enabling the high fidelity secure configuration of web browsers and hence, help prevent and mitigate browser attacks, which are common in today's browsing environment.

CHAPTER 4

HERMES: A High-Level Policy Language for High-Granularity Enterprise-wide Secure Browser Configuration Management

4.1 Introduction to the Chapter

In this chapter, we describe the characteristics, structure, and uses of HERMES. HERMES is a high-level security policy description language. Its characteristics are: (1) enable the specification of organizational domain knowledge in a hierarchical manner; (2) enable the specification of security policies at desired granularity levels within the organizational IT and OT infrastructure; (3) enable security policies to be automatically instantiated into security configurations; (4) it is human-centered and designed for ease of use; (5) it is application and device independent. We show an example of using HERMES to write a high-level policy and show examples of how such policy can be instantiated into a domain and device, user and role, application and action specific security configuration. We also describe the integration of HERMES within the HiFiPol:Browser policy management system. We believe HERMES is a necessary step toward securing the client side of the web ecosystem and prevent or mitigate the current onslaught of web browser-based attacks, such as phishing.

This chapter is organized as follows: An expanded description of the problem that current organizations face when securing the web-based ecosystem is described in section 4.2. Our approach and proposed solution, which the HERMES language is a part of, is described in section 4.3. The characteristics, structure, and grammar of the HERMES language are described in detail in section 4.4. A case study of a fictional organization, that demonstrates and discusses the practical usage of the HERMES language, is presented in section 4.6. The integration of HERMES into the HiFiPol:Browser [8] policy management system is described in section 4.7. Related work is discussed in section 4.8, with a description of differences between our work and previous related research or systems. Future work is presented in

section 4.9. Finally, our conclusion is provided in section 4.10.

4.2 Chapter's Problem

Today's web browsers offer worldwide network accessibility, turing-complete execution capabilities, advanced native graphics, and access to permanent storage. These features, combined with widespread adoption and daily use, make browsers a primary target for exploitation [3, 4]. In addition, web browsers are a primary target of cyber-attacks due to the lack of adequate domain separation. In today's web ecosystem, browsers are used to access critical and confidential organizational systems and data and, at the same time, used to browse the web at large, trusted and untrusted. For example, an organizational user may use a web browser in a portable device, with a default configuration, to read the news and connect untrusted websites. Soon after, the same user in the same device, connects into the organizational VPN and, using the same browser, accesses a sensitive system. In another example, a user, unaware that is being the target of a spear phishing attack, clicks on the provided hyperlink. As a result, the same web browser used to browse the intranet before opens up, using a permissive configuration.

Currently, hi-fidelity and high-granularity security policy design and implementation is practically impossible to achieve in today's technologically diverse organizations. Developing, implementing, and maintaining tailored security configurations considering the number of devices, applications, roles and users, and systems and domains, and their combinations thereof would be unmanageable for most organizations. For example, an organization with 5000 employees, 3 roles per employee, 10000 devices or endpoints, 5 different browsers, and 30 web-applications, would need to develop and maintain hundreds of thousands of hi-fidelity configurations; Or up to millions, depending on the needed group granularity since not all possible combinations would be needed. To compound the problem, today's web browsers use disparate configuration languages and semantics[20, 13]. However, if (a) security policies were described in an easy to understand high-level policy language, (b)

checked for consistency and automatically instantiated into hi-fidelity configurations, and (c) automatically deployed to each endpoint and application, then, we strongly believe that this situation would be reversed.

4.3 Chapter's Approach

In this section, we briefly describe the approach, technologies, and tools that we are currently developing in order to enable IT/OT security administrators to develop and implement hi-fidelity security configurations starting with a high-level description of their organization, users and roles, devices, applications, and their corresponding tailored browser security policies.

The proposed approach is to study and develop a Policy-oriented enterprise-wide policy management system: HiFiPol:Browser. Such a system will enable IT/OT security personnel to write high-level policies, check them for consistency and completeness, generate needed high-granularity secure browser configurations, and then automatically deploy these configurations into the corresponding devices or endpoints.

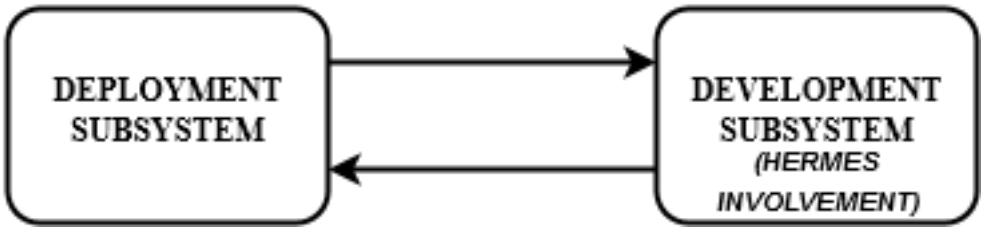

Figure 4.1: HiFiPol:Browser subsystems

The HiFiPol:Browser Policy Management System is divided into two subsystems: *Development* and *Deployment* [8]. Figure 4.1 is a pictorial representation of this. The policy *Development* subsystem enables the development of high-level policies and the transformation of these platform and application independent security policies into domain-, device-, user-, role-, application-, and action-specific web browser configurations [8]. HERMES plays a crucial part in the *Development* subsystem, by acting as the vehicle through which security policies can be specified using a human-centered language. The *Deployment* subsystem

enables the automatic deployment of configurations, which are generated by the *Development* subsystem, across client devices in the organization [8]. Integration of HERMES into HiFiPol:Browser is described in section 4.7. For a more detailed description of the HiFiPol:Browser architecture we refer the reader to the work of Jillepalli et. al.[8].

To put this work in context we briefly describe here the status of this endeavor. (1) We have analyzed current browser security policies for all 3 major browsers and found that there is almost no syntactic or procedural homogeneity in the way current browsers manage security configurations[20, 13]; (2) We have developed a complete prototype, entitled "Open Browser GP", of a GUI-based tool that allows remote configuration of web browsers[20, 13]. From that prototype we learned that a GUI-based tool is a good tool to manage and deploy configurations. However, we also found that a high-level policy is also needed to achieve our goals. (3) We have created HERMES (V0.1), a high-level and policy-oriented language for the specification of such policies. This work was described in a recent publication by Jillepalli et. al.[8]. A detailed description of our plans for future research work is presented in Section 4.9.

4.4 Characteristics of the HERMES Language

HERMES allows IT/OT security personnel to describe their organization and security policies based on the description of four domains: Organizational Domains and Devices, Groups of Users and Roles, Applications, and Actions. Each of these domains can be defined using a hierarchical structure. Policies are declared as actions applied to a given combination of (sub)domains or devices, roles or users, and applications. The HERMES name was coined from the phrase High-level, Easy to use, and Reconfigurable Machine Environment Specification language.

The specific needs which HERMES was designed to address are presented in the form of five primary characteristics: (1) enable the specification of organizational domain knowledge in a hierarchical manner; (2) enable the specification of security policies at desired granularity

levels within the organizational IT/OT infrastructure; (3) enable security policies to be automatically instantiated into security configurations; (4) be human-centered and designed for ease of use; (5) be application and device independent.

Domain Knowledge: HERMES enables the specification of organizational and domain hierarchies natively. For example, an organization can be specified by its name Acme. Then it's Marketing department can be specified as a child of Acme and referred to as Acme.MKTG. A tree or graph structure can be defined at any desired level of granularity. Organizations are well accustomed to using tree-like organizational hierarchies. This hierarchical specification of organizational structures also applies to all policy domains such as groups of devices and users and roles. For example, Acme.ENG.CS.REG.PC1 represents a device and Acme.ENG.ECE.Alice represents a user. This is a well known notation for system administrators. This characteristic is a major difference between HERMES and other previous policy languages. Most other policy languages allow the specifier to declare policy but not the organizational structures that the policy applies to.

Granular Policies: HERMES enables the specification of policies at any desired level of abstraction. Using HERMES a policy designer can specify that a given action or prohibition must apply to all browsers within an organization or to a single browser (application domain), in a single device (device domain), for a single user (role or user domain). For example, we could specify that APP.ALL browsers (application domain) in all devices declared in Acme.ENG.CS (device domain) should have JavaScript disabled (action domain). In another instance, we could specify that Acme.BROWSER.Firefox for user Acme.FINANCE.ACCOUNTANT.Bob should have history disabled.

From Policy to Configuration: HERMES enables the automatic generation of browser policy configurations based on a high-level policy specification. The system that we are building to carry such complex objective is called HiFiPol:Browser. The architecture of this system has been described in [8].

Human-Centred and Designed for Ease of Use: HERMES was designed with the goal

of being written and read by humans rather than computers. Many other policy specification languages in the literature and in use are based on XML or other verbose markup languages, HERMES is not. XML-based languages are great for machine-to-machine communication but maybe not ideal for human-to-human communication. We believe and expect that HERMES will enable policies described in it to be used as the basis for IT/OT personnel design and discussions about organizational security policies. This aspect of being easy to read and write by humans, multiplies the usability of HERMES as a policy specification language, since policies are designed by humans.

Platform Independent: HERMES is a text-based language capable of specifying security policies for all browsers in any platform in any type of organization. This exposes the complete set of source code management tools available to system developers. Tools that many IT personnel and system administrators have began to use in what is called DevOps.

4.5 Formal Grammar of HERMES

An HERMES policy specification is written in entity blocks. Entities have two components: head and body. A Entity Head corresponds to entity type and an identifier. An Entity Body defines a set of fields or attributes and the order of these does not change the semantics. HERMES was designed using a context-free [27], definite-clause [28], and block-like grammar format [29]. Most of the BNF (Backus-Naur Form) specification of HERMES is provided in Listing 4.1.

Listing 4.1: HERMES BNF Notation

```
1 // Policy Rules:
2 // A policy is a set of domain
3 // or policy defining entities.
4 <EntityBlockListDefinition> :=
5 <RightSquare><EntityBlockList><LeftSquare>
6 <EntityBlockList> :=
7 <EntityBlock> |
```

```
8  <EntityBlock> , <EntityBlockList>
9
10 // Policy Entity Rules:
11 // A policy entity is comprised of
12 // an identifier, followed by a colon,
13 // then followed by a name. After the name
14 // a list of attributes is enclosed by { }.
15 <EntityBlock> :=
16 <EntityIdentifierName><Colon>
17 <EntityIdentifierValue>
18 <RightCurly>
19 <EntityMemberListDefinition>
20 <LeftCurly>
21
22 <EntityMemberListDefinition> :=
23 <RightSquare><EntityMemberList><LeftSquare>
24
25 <EntityMemberList> :=
26 <EntityMember> |
27 <EntityMember> , <EntityMemberList>
28
29 <EntityMember> :=
30 <FieldComponent> |
31 <AttributeComponent> |
32 <EntityComponent>
33
34 // Policy Entity Attribute Rules:
```

```
35 // A policy entity attribute is a
36 // field name and corresponding value
37 // separated by a colon.
38 <FieldComponent> :=
39 <FieldName><Colon><FieldValue>
```

4.6 Case Study

In this section, a discussion about the practical uses of the HERMES specification language is presented by using a case study and a set of example polices. The final resulting web browser configurations, respective to the set of example policies, are also shown in this section.

4.6.1 HERMES Policy Composition

In HERMES, a policy is a set of relations between four different entities, each characterized by Figures 4.2, 4.3, 4.4, and 4.5. In these figures, Dev, App, Usr, and Act correspond to Domain, Subdomain, or Device, Application, User ro Role, and Action. A policy can be defined as a subset of the Cartesian product of all entity sets.

$$\therefore Policy \in (Dev \times App \times Usr \times Act)$$

An example domain and policy specification for a fictional organization, named Acme is presented here. The example domain structure, which will be reflected in the HERMES specification, is represented through the following graph structures:

4.6.2 Case Study Organizational Hierarchies

The hierarchical representation os Acme's sub-domains and devices is represented in Figure 4.2.

Categorization of applications used by Acme and the respective versions of those applications, is represented in Figure 4.3.

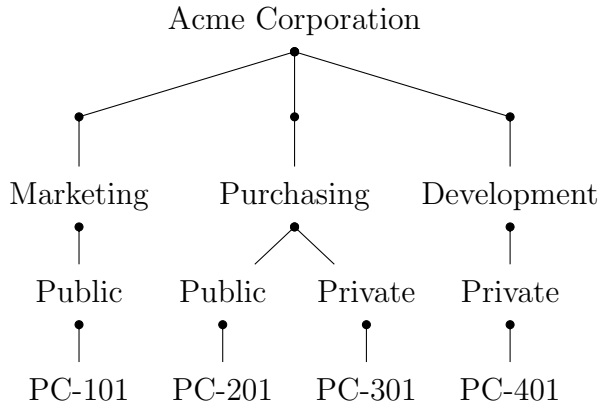

Figure 4.2: Organizational hierarchy graph.

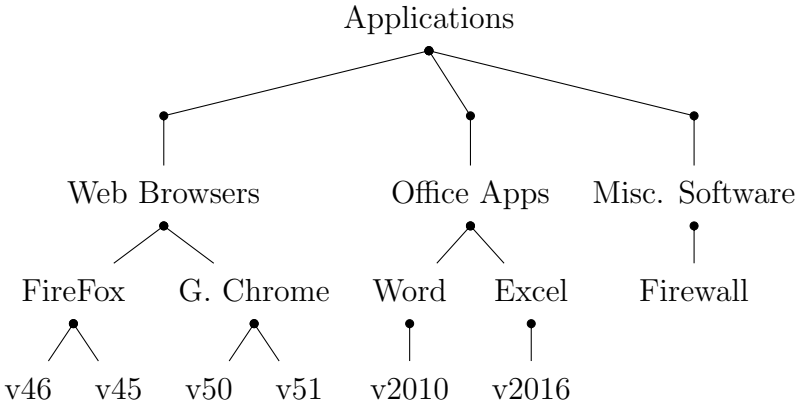

Figure 4.3: Application hierarchy graph.

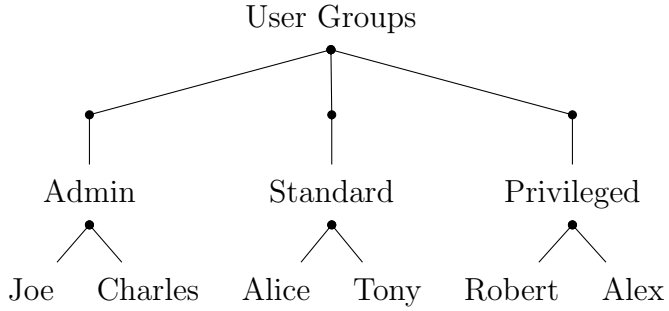

Figure 4.4: User-Group hierarchy graph.

The structure and grouping of users and their respective role assignment is represented by Figure 4.4.

A set of relevant configuration actions applicable to applications used by Acme are represented by Figure 4.5.

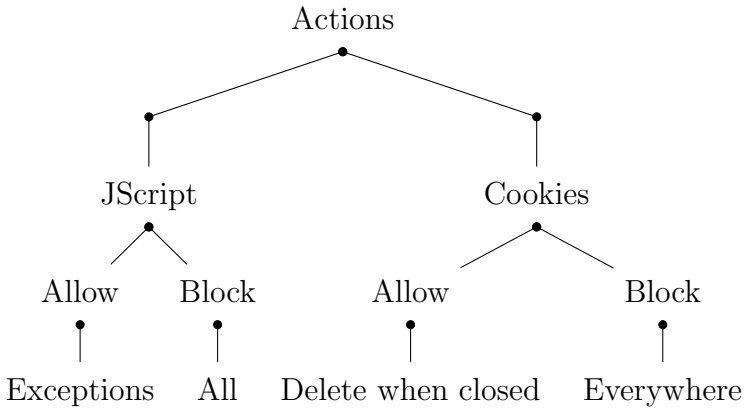

Figure 4.5: Possible-Actions hierarchy graph.

4.6.3 Usage of HERMES and Subsequent Results

The following are a collection of four listings, each demonstrating the HERMES implementation for (a) organization's environment specification, (b) security policy specification with respect to organization's different departments, (c) policy knowledge-base (collection of instantiated policies) and (d) final generated configuration corresponding to the specific policies (as mentioned in point (b)). These listings are as follows:

Using HERMES to specify domain data

Listing 4.2 is an excerpt, demonstrating the usage of HERMES language in specification of organizational domain knowledge for the fictional Acme Corporation. All four of the specification matrix layers [Domain/Device, Application, User/Role, and Action] can be specified using HERMES in such a hierarchical manner.

Listing 4.2 explanation: Let us assume there are three departments in Acme Corp: Marketing (`MKT`), Purchasing (`PRCH`), and Development (`DEV`). Let us also assume there are four devices in Acme, across the three above-mentioned departments: `PC-101` in `MKT`, `PC-201` and `PC-301` in `PRCH`, and `PC-401` in `DEV`. The HERMES specification conveying this infrastructural information is present in the Listing 4.2.

Using HERMES to specify action policies

Listing 4.3 shows the usage of the HERMES language for the specification of security policies for web browsers in different departments of Acme. HERMES allows a wide range of specifications inside the policy block, ranging from descriptions and rationales to applicability and inheritance. `ApplyTo` and `ToApp` denote "apply to which group of users/roles" and "apply to which applications" respectively.

Listing 4.2: HERMES excerpt in specifying organizational domain knowledge for Acme.

```
1  Domain: Acme
2  {
3  Description: "Acme Organization."
4  List: [MKT, PUR, DEV]
5  }
6
7  SubDomain: MKT
8  {
9  Description: "Marketing Department."
10 DeviceList: [PC-101]
11 Parent: Acme
12 }
13
14 SubDomain: PRCH
15 {
16 Description: "Purchasing Department."
17 DeviceList: [PC-201, PC-301]
18 Parent: Acme
19 }
20
```

```
21 SubDomain: DEV
22 {
23 Description: "Development Department."
24 DeviceList: [PC-401]
25 Parent: Acme
26 }
```

Listing 4.3 explanation: The HERMES policies in the listing demonstrates the security policy specifications for devices of three departments in Acme, the departments being: Marketing, Purchasing, and Development. These five policies specify configurations for two web browsers (Mozilla Firefox and Google Chrome) so that the web browsers are specified to perform: (a) enable JavaScript functionality in the Marketing department's devices (`PID_01_000`), (b) enable the cookie functionality, while disabling third-party cookies, in the Purchasing department's devices (`PID_02_000`, `PID_02_001`, and `PID_02_002`), and (c) disable all JavaScript functionality in the Development department's devices (`PID_03_000`).

In Listings 4.3 and 4.5, we can observe that in due course of the processing of policies, as presented in the example case study, policy conflicts arise, which need to be detected, identified, and resolved in order to enable the smooth and seamless transition of policies from HERMES to deployable configurations. Particularly in Listing 4.3, we can observe a direct conflict of policy, because policies `PID_02_000` and `PID_02_001` are opposite to each other and thus, creates a conflict.

Likewise, policy `PID_02_002` is a child of, and is in conflict with, `PID_02_001`. Thus, resolution algorithms must be developed to solve such conflicts. Using these algorithms, the knowledge-base will be instantiated accordingly, to deal with conflicts and to associate the facts of knowledge-base with respective entities. (More on conflicts in section 4.9).

Listing 4.3: HERMES excerpt in specifying security policies for Acme 's applications.

```
1 Policy: PID_01_000
2 {
```

```
3 Description: "Enabling JavaScript functionality."
4 Rationale: "Marketing staff need JavaScript to design ads."
5 Status: "Enabled"
6 Field: JavaScript: "Enabled"
7 ApplyTo: "MKT"
8 ToApp: "APP.ALL"
9 }
10
11 Policy: PID_02_000
12 {
13 Description: "Enabling all cookie functionality."
14 Rationale: "Purchasing websites need cookie functionality."
15 Status: "Enabled"
16 Field: Cookie: "Enabled"
17 ApplyTo: "PRCH"
18 ToApp: "APP.ALL"
19 }
20
21 Policy: PID_02_001
22 {
23 Description: "Disabling all cookie functionality."
24 Rationale: "Purchasing websites require security against
25 cookie injection attacks."
26 Status: "Enabled"
27 Field: Cookie: "Disabled"
28 ApplyTo: "PRCH"
29 ToApp: "APP.ALL"
```

```
}

Policy: PID_02_002
{
Description: "Disabling all third party cookies."
Rationale: "Purchasing data requires privacy."
Status: "Enabled"
Field: Cookie.Third: "Disabled"
ApplyTo: "PRCH"
ToApp: "APP.ALL"
Parent: "PID_02_000"
}

Policy: PID_03_000
{
Description: "Disabling JavasScript functionality."
Rationale: "Development Environments
need protection against script attacks."
Status: "Enabled"
Field: JavaScript: "Disabled"
ApplyTo: "DEV"
ToApp: "APP.ALL"
}
```

The process of instantiation from policy to final deployable configurations is composed of multiple stages. This is currently ongoing research work. More detail on the different forms of instantiation in section 4.7 and the process is shown in Fig. 4.7.

Listing 4.4: Excerpt of policy knowledge-base, which is a collection of facts, instantiated from policies in listing 4.3.

```
// Domain data facts
domainDescription('policy.yaml','Acme',
'Acme Corporation').
domainList('policy.yaml','Acme','MKT, PUR, DEV').
subDomainDescription('policy.yaml','MKT',
'Marketing Department.').
subDomainDeviceList('policy.yaml','MKT','PC-101').
subDomainParent('policy.yaml','MKT','Acme').
subDomainDescription('policy.yaml','PRCH',
'Purchasing Department.').
subDomainDeviceList('policy.yaml','PRCH','PC-201, PC-301').
subDomainParent('policy.yaml','PRCH','Acme').
subDomainDescription('policy.yaml','DEV',
'Development Department.').
subDomainDeviceList('policy.yaml','DEV','PC-401').
subDomainParent('policy.yaml','DEV','Acme').

// Policy specification facts
policyDescription('policy.yaml',
'PID_01_000','Enabling JavaScript functionality.').
policyRationale('policy.yaml','PID_01_001',
'Marketing staff requires JavaScript for designing ads.').
policyStatus('policy.yaml','PID_01_001','Enabled').
policyField('policy.yaml','PID_01_001',
'Cookie.Lifetime,"Disabled').
```

```
26 policyApplyTo('policy.yaml','PID_01_001','RLE.admin').
27 policyToApp('policy.yaml','PID_01_001','APP.Google_Chrome').
28 policyParent('policy.yaml','PID_01_001','PID_01_000').
```

Policy Knowledge-Base generated from HERMES

Listing 4.4 shows the resultant policy knowledge-base, which has been generated as a result of parsing the domain knowledge (like the one given in listing 4.2) and policy specifications (like the ones given in listing 4.3). The knowledge-base is a collection of facts, which have been instantiated on the basis of given policies. In Listing 4.4, the facts reflect domain data of Listing 4.2 and policy `PID_01_000` from listing 4.3.

Final Instantiated Web Browser Configurations

Listing 4.5 demonstrates the final resultant configuration corresponding to the policies specified in listing 4.3. These configurations are rewritten after a series of instantiations similar to the one showed in listing 4.4. The process of rewriting is manual now, but is planned to be automatic in the future. The configuration is shown for two web browsers, Mozilla Firefox and Google Chrome (as we are assuming Acme uses the two web browsers only).

Listing 4.5: Resultant actual web browser configurations corresponding to the policies given in listing 4.3.

```
1 For Mozilla Firefox browsers in Marketing dept.'s devices:
2 user_prefs("javascript.enabled", true);
3
4 For Mozilla Firefox browsers in Purchasing dept.'s devices:
5 user_prefs("network.cookie.cookieBehavior", 0);
6 user_prefs("network.cookie.cookieBehavior", 2);
7 user_prefs("network.cookie.thirdparty.sessionOnly", false);
8
```

```
For Mozilla Firefox browsers in Development dept.'s devices:
user_prefs("javascript.enabled", false);

For Google Chrome browsers in Marketing dept.'s devices:
"default_content_setting_values":{"javascript":0},
"default_content_settings":{"javascript":0},

For Google Chrome browsers in
Purchasing department's devices:
"default_content_setting_values":{"cookies":0,"cookies":2},
"default_content_settings":{"cookies":0,"cookies":2},
"block_third_party_cookies":true,

For Google Chrome browsers in Development dept.'s devices:
"default_content_setting_values":{"javascript":2},
"default_content_settings":{"javascript":2},
```

Listing 4.5 explanation: The text in specified listing demonstrates actual web browser configuration file syntax for Mozilla Firefox and Google Chrome, for devices of the three departments in Acme: Marketing, Purchasing, and Development. these configurations instruct web browsers to: (a) enable all JavaScript functionality in Marketing department's devices, (b) enable cookie functionality, while disabling third-party cookies, in Purchasing department's devices, and (c) disable all JavaScript functionality in Development department's devices.

Though there exists a conflicting cookie configuration setting in Purchasing department's configurations for both Mozilla Firefox and Google Chrome web browsers, the browsers give priority to the configuration occurring first in the configuration file. Therefore, since enabling cookies is the first occurring configuration in both the web browsers, it is set as the web

browsers' setting.

4.6.4 Deployment of Final Configurations

Proposed Repository Structure

The final configuration files would be relocated to respective agent repositories on a server.

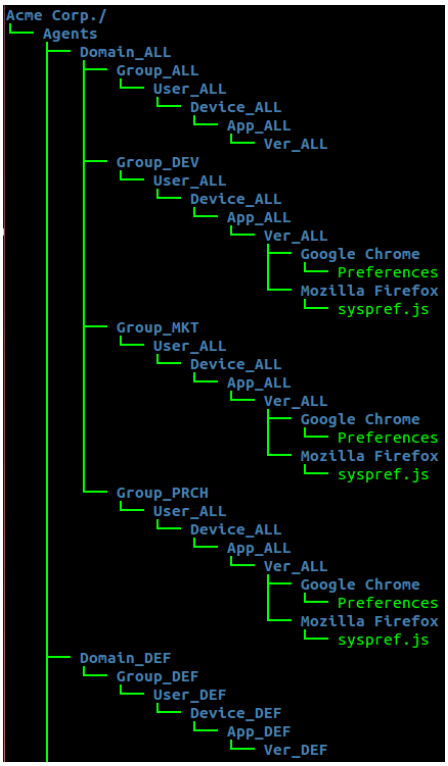

Figure 4.6: Proposed repository structure for configuration files deployment to agents through HiFiPol:Browser architecture design.

For the purpose of this case study, we have adopted the `ALL` deployment approach for the sake of simplicity. As can be seen in figure 4.6, under `Domain_ALL`, there are four groups: DEF, DEV, MKT, PRCH. Whereas: DEV, MKT, and PRCH are the group repositories for respective policies of Development, Marketing, and Purchasing departments within Acme , applied to all users, devices and applications (Firefox and Chrome in this case) in the respective departments.

Since we assumed that there would be only two browsers (Mozilla Firefox and Google Chrome) in the Acme , we can see in figure 4.6 that there are only two configuration files

(under respective web browser directories) for each of the specific department which have had policies specified (as exampled in listing 4.3). From these directories, the configuration files will be transferred to respective agent machines currently through SSH using python scripting. However, SSH is only intended to be a temporary prototype and we are looking at other sophisticated deployment platforms.

The `ALL` and `DEF` keywords in Fig. 4.6 correspond to unconditional policy application and default policy application, respectively.Now that we have seen the practical implementation potentiality of HERMES, we can see how HERMES can fit into a bigger picture (which is HiFiPol:Browser) in the next section.

4.7 Integration into the HiFiPol Framework

HERMES has been designed to enable text-based policy specification functionality in the framework of HiFiPol:Browser [8]. HERMES acts as the foothold medium between humans and machine understandable facts, which will be further processed into deployable configurations.

Fig. 4.7 shows the architecture of the *Development* subsystem.

Under our current research plan, HERMES is to be used in the Policies File, also called "Policies Specification File", which is a file or set of files, that house the specification policies. As seen in the Figure 4.7, HERMES, present in *Policies File* will be parsed into a *Policy Knowledge-Base* (PKB) through use of *Policy Implementation Engine*. The PKB then will undergo conflict detection, identification and resolution, resulting in *Conflict-Free Policy Knowledge-Base* (C-F PKB). The C-F PKB will be then processed by domain / device instantiation leading to creation of a *Level One Instantiated C-F PKB*.

At that stage, user/role instantiation will be carried out to result in a *Level Two Instantiated C-F PKB*. A final instantiation of actions would be carried on, resulting in creation of a *Fully Instantiated C-F PKB*, which would be a conflict-free policy knowledge-base with complete (four-layer) instantiation of given policies. These fully instantiated policies will be

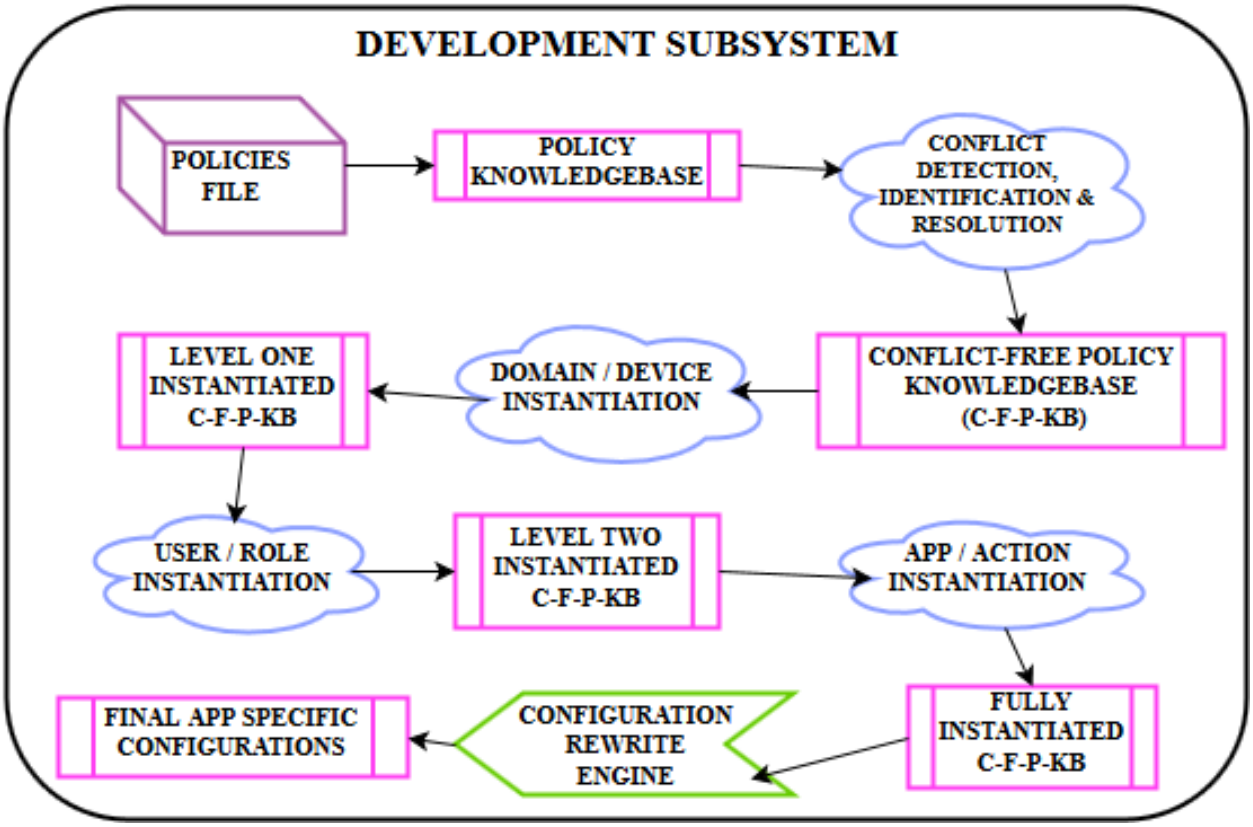

Figure 4.7: HiFiPol:Browser development subsystem

then rewritten into *Final Application Specific Configurations* using *Configuration Rewrite Engine*.

The further process, which succeeds the stage last discussed, where we have the final application specific configurations, is detailed in the sub-subsection 4.6.4. Therefore, in such a way, HERMES would play an important role (and would seamlessly integrate into the architecture design) by triggering the chain of transformations, which ultimately help in achieving the functionality of HiFiPol:Browser Architecture.

4.8 Tools Similar to HERMES

Several XML-based languages have been created for specification of configurations. Most relevant to our work are: Security Policy Assertion Language (SecPAL) [30], Open Vulnerability and Assessment Language (OVAL), eXtensible Access Control Markup Language (XACML) [31], Extensible Configuration Checklist Description Format (XCCDF)[32], Mar-

grave (focused on firewall analysis) [33, 34], and Security Policy Language (SPL) [35]. Some of these are part of the Security Content Automation Protocol (SCAP) specification [36].

The SCAP-related languages and protocols are designed for the purpose of configuration specification, auditing, and validation. In contrast, the HERMES language is oriented toward high-level security policy specification. SCAP languages specifications are highly detailed and XML-bases. This makes them well-suited for tool-to-tool communication and enables compatibility between tools. But, System and network administrators are used to text-based specification languages. For example, Linux and Cisco(R) IOS configuration files.

However, this also makes SCAP languages not well suited for specifying high-level policies that can be mechanically converted to low-level configurations and checked for validity. SCAP-based configuration specifications could be generated from HERMES high-level policies after policy instantiation has occurred.

The work discussed in "Security Policy Language" [35] by Dr. Perez et. al., about creating the Security Policy Language (SPL) and the Common Information Model (CIM) is similar to the contribution described in this chapter. However, SPL and CIM are also XML-based languages, thus are machine-oriented rather than human-focused.

The similarities between SPL and HERMES are: (a) clear and well defined semantics, (b) flexibility and extensibility of language, (c) readability, and (d) independence with respect to platform and domain. Perez et. al. also address the issues of conflict detection, refinement, and resolution.

The differences between the work of Perez et. al. and the work describe in this chapter are: (1) In SPL, system specifications are described in a separate language called as System Description Language (SDL), whereas, when using HERMES, there is no foreseen need for a second language, and (2) For security policy (not configuration) specification, HERMES fulfills the specific purpose of acting as a medium to express domain information and policy information using the same language. All involved entities within domain and policy can be expressed using HERMES.

"The Margrave Policy Analyzer"[33, 34], a tool developed by Kathi Fisher et. al. is similar to HiFiPol:Browser, and by extension, to HERMES. However, differences do exist. Some of the differences are: HERMES is human-centric and hierarchical in structure, where entities are not nested inside several other entities. In HERMES, relationship between entities are discussed using names, but not using nesting.

4.9 Extension of HERMES

In the recent past we have performed research work that resulted in the Open Browser GP tool [20, 13] for GUI editing and automatic deployment of browser configurations. A lesson learned from that work was that editing configurations in a GUI is extremely time consuming because of the sheer number of options. This was one of the reasons, among several other, that propelled us to create high-level language that could abstract most of the complexity. Retrospectively, this drawback is similar to the drawbacks of Microsoft's Policy Management tools used in the Active Directory system. These tools are not true policy management tools, they are configuration management tools.

In the future, we intend to investigate approaches to GUI editing of high-level security policies, rather than low-level configurations. These GUI tools should be able to show current and expected system configurations, and current and expected high-level security policies. Our work on policy abstraction and refinement will need to be considered within this context. In addition, new GUI tools should be able to read and write the HERMES language in order to enable synchronized GUI and text-based policy management.

Future research work that we have already began to tackle is the analysis of policy conflicts in HERMES and the conflict resolution rules, strategies, and algorithms. This is an essential step in order to be able to proceed with the automatic policy instantiation process. We plan to analyze, in-depth, the possible conflicts that could arise when writing a high-level policy. Then we plan to develop techniques and algorithms for conflict detection and resolution. Preliminary analysis demonstrates that most conflicts are either direct policy

conflicts or can be discovered and solved by applying inheritance rules.

With respect to design level optimizations, we have began devising a mechanism called *conditional instantiation*. In conditional instantiation there are two pre-defined configuration generation methods: *ALL* and *DEF* (for default). When ALL is used, policies are applied to all children objects and subjects in a group without having to duplicate configurations for each member of the group. When DEF instantiation is indicated, policies are applied if and only if there are no other policy specifications or actions already applied (or expected to be applied) to a particular group. Both of these methods will greatly reduce the number of generated policy instances and enable the specification of policies at any desired level of abstraction without incurring on a performance and storage penalty. Implementing a deployment model based on this conditional instantiation mechanism will improve the overall system efficiency by decreasing the number of configuration files to its minimum possible.

4.10 Chapter Conclusion

Currently, in a technically diverse organization, there are extremely difficult challenges associated with ensuring a secure web browser ecosystem. To aid with this problem we have designed HiFiPol:Browser - A policy-oriented and high-level web browser policy management system. At the core of HiFiPol:Browser is HERMES. HERMES is a high-level security policy specification language and it is the contribution presented in this chapter. The specific needs which HERMES is designed to address, and the trade-offs undertaken during design of HERMES are discussed in section 4.4. An example case study was introduced that demonstrates HERMES' usage. The case study (section 4.6) shows how both the hierarchical organizational domain knowledge and the policies of an organization can be represented in HERMES, in order to enable the designated security personnel to design policies at any desired level of granularity.

We also showed an example of how final configurations would look like after policy transformation and instantiation is performed. An explanation of how we plan to integrate HER-

MES into the HiFiPol:Browser system was given in section 4.7. A survey of related research was presented in section 4.8, describing the similarities and differences between HERMES and other existing configuration languages. Finally, in section 4.9, we described the list of future tasks. We strongly believe that HERMES and HiFiPol:Browser or a similar high-level policy language and associated system with similar objectives are very much needed. HERMES and HiFiPol:Browser have the potential to enable the implementation of *defense in depth* and *domain separation* [19, 37] secure design principles for browser security.

CHAPTER 5

Future Work

We envision six main areas where future work is needed. Firstly, a complete web browser security policy scenario of a fictional organization needs to be developed [1]. Secondly, a prototype (v0.1) of HiFiPol:Browser architecture needs to be developed, to demonstrate a *proof-of-concept* for HiFiPol:Browser. Thirdly, conflict detection and conflict resolution algorithms are needed to make feasible, the practical (real-world) implementation of HiFiPol:Browser. Fourthly, web browser security policies need to be semi-automatically transformed into configurations, at any/all levels of granularity. Fifthly, a stable and evolved version of HERMES language needs to be formalized and released. Lastly, a centralized repository for web browser configuration files needs to be developed. We expand on each of these six areas of future work, as follows:

5.1 Organizational Scenario

Based on our experience from the experiments conducted during development of HERMES v0.1, we realized that the nature of HiFiPol:Browser tool is extremely open-ended and possibilities of consideration are exponentially numerous. Therefore, so as to limit the possibilities of consideration and in order to produce a practical "*proof-of-concept*" in a reasonable amount of time, we suggest creation of a complete web-browser security policy scenario; for a fictional organization, to be the first priority in future of this project.

A complete web-browser security policy scenario of a fictional organization consisting of multiple case studies is needed; which explore the following: (a) a formal description of hierarchies of the fictional organization's resources (both human and computational): including the layout of the organization's infrastructure, (b) implementation/translation of

[1] We conducted a search to find an open-source and complete web-browser security policy scheme for an organization, but were not able to find any. What we found in our search included, but limited to: web-browser recommendations for organizations, outdated specifications of configurations (example, SCAP v2.0) and security suggestions for web-browser security in organizations.

web-browser security configuration recommendations into high-level security policies, (c) usage of HERMES language in writing high-level security policies and (d) most common issues and difficulties which are expected to be encountered on a regular basis by system administrator and possible solutions for those issues.

5.2 HiFiPol:Browser v0.1

A prototype of all HiFiPol:Browser processes, as discussed in this thesis (refer Chapter 3.3) needs to be developed; so that a "*proof-of-concept*" can be established for HiFiPol:Browser's architecture process; from start to finish. This includes endeavoring to develop a deployment mechanism, which is intended to be the delivery platform, through which remote distribution of web-browser configuration files takes place; from server to client machines.

Such an endeavor includes: evaluating and selecting a suitable DevOps amongst Chef, Puppet, and Ansible and through the use of that platform, develop a remote deployment mechanism for web-browser configuration files. The evaluation should primarily xonsider three characteristics: (a) ease of use (including ease of learning), (b) flexibility/modularity, and (c) scalability.

5.3 Conflict Detection and Resolution

Due to the nature of open-ended modularity of HERMES and HiFiPol:Browser, conflicts are bound to occur during practical implementation. Therefore, it is imperative that such policy conflicts be detected, identified, and resolved; because: not doing so will probably break the configurations of web browsers involved. The resolution could be semi-automatic; in the sense that the process still takes input and directives from the system administrator, regarding resolution preferences - while automatically resolving the conflicts.

To enable such detection, identification, and resolution of conflicts; algorithms are required to be designed- which incorporate some potentially common conflicts and the conflicts which are inherent in the developed organizational scenario (see Section 5.1). These algorithms, are then needed to be practically implemented, in code, for successful integration

of conflict detection, identification, and resolution functionality into HiFiPol:Browser.

5.4 Semi-Automatic Policy Instantiation

High-level (human-comprehensible) security policies for web-browser should be transformed into low-level (machine-comprehensible) configurations, at any/all levels of granularity. Every attribute of the policy specification should be taken into consideration while transforming policy into configuration. Since high-level security policies are expected to contain any attribute(s) from an organization's infrastructure, the transformation should be capable of associating the respective attribute(s) to the action(s) specified in policies. This process is called "*Policy Instantiation*".

If not completely automatic, at least semi-automatic (with some human-input) practical implementation of instantiation of policies into configurations is needed; for the given policies in the developed organizational scenario (see Section 5.1).

5.5 HERMES v1.0

Based on our experience from the experiments conducted during development of HERMES v0.1, we realized that progress of any research in the HiFiPol:Browser project will involve evolving and modifying the current HERMES v0.1 prototype. In this manner, after the HERMES prototype undergoes multiple iterative cycles of improvement and evolution, a stable and more mature version of HERMES needs to be released.

HERMES v1.0: A stable language which can be used to write high-level (human understandable) security policies for web browsers should be released. HERMES v1.0 should be powerful to the extent that it should be possible to use HERMES in writing security policies for web browsers at any level of granularity, including the ability to specify an organization's any desired infrastructural attributes in the security policy. We expect this HERMES v1.0 to ideally be powerful enough to write policies for any organization's infrastructure, but it should be made sure that it is powerful enough to write policies at least for the infrastructure of fictional organization designed (see Section 5.1).

5.6 Configuration Repository

Design and development of a centralized repository for web browser configuration files, residing in the central server machine; is needed. This repository should have a directory each, for all the client/agent machines connected to the server. The configuration files, which are produced as a result of the semi-automatic policy instantiation (see Section 5.4), should be then transfered to the respective client-machine's target folder in repository; where they should reside until the system administrator authorizes the deployment process of: transferring configuration files to client machines (see Section 5.2).

We expect the centralized repository to be powerful enough such that it can create directory structure of client machines for any organization, but it should be made sure that it is powerful enough to create directory structure of client machines; at least for the fictional organization designed (see Section 5.1).

CHAPTER 6

Thesis Conclusion

To conclude - in contemporary technological world, we would like to emphasize on the point that it is high time to start implementing a defense-in-depth approach, using policy-oriented, and high-granularity, remote configuration of systems/tools; especially web browsers.

The contributions of this thesis may be classified into three categories: 1) arguing for a potentially better approach, 2) materializing the better approach, and 3) advancing the better approach. The three contributions presented in this thesis fall into one of the aforementioned categories, as follows:

In the first category, *Arguing for a potentially better approach*, our contribution was to:

- Demonstrate the steps required to achieve a high-granular configuration using a currently popular method.
- Leveraging on that demonstration, argue that there is a requirement for designing a tool which utilizes the policy-oriented, and high-granularity remote web browser configuration paradigm.

In the second category, *Materializing the better approach*, our contribution was to:

- Design an architecture of a policy-oriented, and high-granularity remote web browser management system; called as "HiFiPol:Browser".
- Present a comprehensive explanation of intended functionality of each architectural component and interactions between the architectural components.

In the third category, *Advancing the better approach*, our contribution was to:

- argue for the contemporary necessity of a high-level, easy-to-use, and a platform-/application-independent policy specification language for web browsers.

- To help bolster the argument, we created a prototype (v0.1) of afore-mentioned specification language called HERMES.

Overall, the results of this research demonstrate that it is possible to further develop HiFiPol:Browser and subsequently, HERMES; through employing an iterative agile method of evolving the architecture and language, as research progresses. We are convinced that the contributions presented in this thesis will enable the society to at least inch towards a better secured web browsing operational environment; through utilization of policy-oriented, and high-granularity remote web browser configuration approach.

References

[1] Verizon Inc., "2016 data breach investigations report," online, February 2017. [Online]. Available: http://www.verizonenterprise.com/verizon-insights-lab/dbir/2016/

[2] "Internet usage statistics," http://www.internetworldstats.com/stats.htm, Nov. 2015, miniwatts Marketing Group.

[3] J. Nielson, C. Williamson, and M. Arlitt, "Benchmarking modern web browsers," in *Proc. 2nd IEEE Workshop on Hot Topics in Web Systems and Technologies*, 2008.

[4] A. Wang, "Are you using the most secure web browser?" http://securitywatch.pcmag.com/web-browsers/325447-are-you-using-the-most-secure-web-browser, Aug. 2014.

[5] "2016 - state of the cloud report," http://www.rightscale.com/press-releases/rightscale-2016-state-of-the-cloud-report, Feb. 2016, rightScale Incorporation.

[6] A. Zammouri and A. A. Moussa, "Safebrowse: A new tool for strengthening and monitoring the security configuration of web browsers," in *Proc. International Conference on Information Technology for Organizations Development (IT4OD)*, 2016.

[7] H. Al-Maghrabi, ShahdShami, S. A. Sefri, and H. M., "Private browser mode: Secured! or unsecured?" *International Journal of Applied Engineering Research*, vol. 11, no. 14, pp. 8238–8242, 2016.

[8] A. A. Jillepalli and D. Conte de Leon, "An architecture for a policy-oriented web browser management system: HiFiPol: Browser," in *Proc. 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC-2016)*, June 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7552242/

[9] A. A. Jillepalli, D. Conte de Leon, S. Steiner, and F. T. Sheldon, "Hermes: A high-level policy language for high-granularity enterprise-wide secure browser

configuration management," in *Proc. 2016 IEEE 07th Symposium Series On Computational Intelligence (SSCI-2016)*, December 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7849914/

[10] R. Joyce, "Disrupting nation state attacks," in *USENIX Enigma Conference 2016*. USENIX, January 2016. [Online]. Available: https://www.youtube.com/watch?v=bDJb8WOJYdA

[11] A. Boyd, "Nsa hacker chief named white house cyber czar," March 2017. [Online]. Available: http://fifthdomain.com/2017/03/14/nsa-hacker-chief-named-white-house-cyber-czar/

[12] Microsoft Technet, "Active directory domain services overview," Online, April 2007. [Online]. Available: https://technet.microsoft.com/en-us/library/9a5cba91-7153-4265-adda-c70df2321982

[13] D. Conte de Leon, V. A. Bhandari, A. A. Jillepalli, and F. T. Sheldon, "Using a knowledge-based security orchestration tool to reduce the risk of browser compromise," in *Proc. 2016 IEEE 07th Symposium Series On Computational Intelligence (SSCI-2016)*, December 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7849910/

[14] Microsoft Technet, "Organizational units," https://technet.microsoft.com/en-us/library/cc978003.aspx, February 2017.

[15] Google Inc., "Google chrome enterprise installers," Online, February 2017. [Online]. Available: https://enterprise.google.com/chrome/chrome-browser/

[16] Google Inc., "Google chrome administrative templates," https://dl.google.com/dl/edgedl/chrome/policy/policy_templates.zip, February 2017.

[17] Microsoft Corp., "Microsoft edge administrative templates," https://www.microsoft.com/en-us/download/details.aspx?id=53430, February 2017.

[18] Microsoft Corp., "ADMX and ADML file structure," http://technet.microsoft.com/en-us/library/cc772507(v=ws.10).aspx, April 2007.

[19] A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating system concepts - 8th edition," 2009.

[20] V. A. Bhandari, "Analysis of security policies in major web browsers and development of a multibrowser and multiplatform browser configuration tool: Open browser gp," Master's thesis, University of Idaho, Moscow, Idaho, May 2015.

[21] J. Moskowitz. (2008, Jan.) Inside adm and admx templates for group policy. [Online]. Available: http://technet.microsoft.com/en-us/magazine/2008.01.layout.aspx

[22] J. Stromberg. (2013, Aug.) Configuring google chrome via group policy. [Online]. Available: http://jackstromberg.com/2013/08/configuring-google-chrome-via-group-policy/

[23] "Deploying and securing google chrome in a windows enterprise," https://www.nsa.gov/ia/_files/app/deploying_and_securing_google_chrome_in_a_windows_enterprise.pdf, Oct. 2012, national Security Agency / Central Security Service.

[24] "Gpo for firefox add-on," Jan. 2008. [Online]. Available: https://addons.mozilla.org/en-US/firefox/addon/gpo-for-firefox/

[25] Red Hat Inc., "Freeipa," http://www.freeipa.org/page/About/, November 2014.

[26] "Mobile device management software," https://www.apple.com/support/business-education/mdm/, Apr. 2008, apple Incorporation.

[27] N. Chomsky, "Three models for the description of language," in *Proc. Information Theory, IRE Transactions - Volume: 2, Issue: 3*, 1956, pp. 113–124.

[28] H. Shimazu and Y. Takashima, "Multimodal definite clause grammar," in *Proc. COLING '94 Proceedings of the 15th conference on Computational linguistics - Volume 2*, Stroudsburg, PA, USA, 1994, pp. 832–836.

[29] M. Johnson, "Two ways of formalizing grammars," in *Proc. Linguistics and Philosophy - Kluwer Academic Publishers - Volume 17*, Netherlands, 1994, pp. 221–248.

[30] "Policy language research," http://research.microsoft.com/en-us/projects/securitypolicy/, microsoft Research.

[31] "Testing and verification of security policy," https://sites.google.com/site/asergrp/projects/policy, tao Xie, Vincent Hu and Rick Kunh.

[32] "XCCDF - The Extensible Configuration Checklist Description Format," https://scap.nist.gov/specifications/xccdf/, National Institute of Standards and Technology.

[33] T. Nelson, C. Barratt, D. J. Dougherty, K. Fisler, and S. Krishnamurthi, "The margrave tool for firewall analysis," in *Proc. USENIX Large Installation System Administration Conference (LISA)*, 2010.

[34] "The margrave policy analyzer," http://www.margrave-tool.org/, Dan Dougherty and Kathi Fisler and Shriram Krishnamurthi.

[35] J. B. Bernabe, J. M. M. Perez, J. M. A. Calero, J. D. J. Re, F. J. Clemente, G. M. Perez, and A. F. Skarmeta, "Security policy specification," in *Network and Traffic Engineering in Emerging Distributed Computing Applications*, J. Abawajy, M. Pathan, M. Rahman, and M. M. Deris, Eds. Oxford: IGI Global, 2013, ch. 04, pp. 66–93.

[36] "Emerging specification listing," http://scap.nist.gov/emerging-specs/listing.html, national Institute of Standards and Technology.

[37] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

Appendix A: Copyright and Credit Notice

As listed in the Section 1.6, the Chapters 3, and 4 of this thesis are currently copyrighted by Institute of Electrical and Electronics Engineers Inc. (IEEE). Permission for reproduction of complete article, towards use in theses/dissertations has been given by the IEEE, provided the following statements are placed in the theses. Figures 6.1 and 6.2 provides proof of this permissiveness. The content of these reproduced articles has been changed to suit the format of a thesis; instead of a scholarly article. The numbering of figures and listings has changed, due to being reproduced in this thesis. However, the figures and listings, and their captions themselves remain unchanged.

Chapter 3 of this thesis ©2016 IEEE. Reprinted, with permission. Citation: Ananth A. Jillepalli, and Daniel Conte de Leon, "An Architecture for a Policy-Oriented Web Browser Management System: HiFiPol: Browser", In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, vol. 2, pp. 382-387. IEEE, 2016. DOI: 10.1109/COMPSAC.2016.50 [8].

Chapter 4 of this thesis ©2016 IEEE. Reprinted, with permission. Citation: Ananth A. Jillepalli, Daniel Conte de Leon, Stuart Steiner, and Frederick T. Sheldon, "HERMES: A high-level policy language for high-granularity enterprise-wide secure browser configuration management", *Computational Intelligence (SSCI) 2016 IEEE Symposium Series on*, pp. 1-9, 2016. DOI: 10.1109/SSCI.2016.7849914 [9].

For portions not copyrighted by IEEE or any other publisher (as of the date of this publication), and for this thesis as a whole; copyrights are retained by the author. Permission for not-for-profit and academic use is granted. For any other right to copy, transfer, reprint, republish, or for-profit use; permission must be sought from the author (Ananth A. Jillepalli).

https://s100.copyright.com/AppDispatchServlet#formTop

Copyright Clearance Center RightsLink®

Home | Create Account | Help

IEEE Requesting permission to reuse content from an IEEE publication

Title: HERMES: A high-level policy language for high-granularity enterprise-wide secure browser configuration management

Conference Proceedings: Computational Intelligence (SSCI), 2016 IEEE Symposium Series on

Author: Ananth Jillepalli

Publisher: IEEE

Date: Dec. 2016

Copyright © 2016, IEEE

LOGIN

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. Already a RightsLink user or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

Copyright © 2017 Copyright Clearance Center, Inc. All Rights Reserved. Privacy statement. Terms and Conditions.
Comments? We would like to hear from you. E-mail us at customercare@copyright.com

Figure 6.1: Permission from IEEE to reproduce an article as Chapter 2 of this thesis.

https://s100.copyright.com/AppDispatchServlet#formTop

Copyright Clearance Center RightsLink®

Home | Create Account | Help

IEEE Requesting permission to reuse content from an IEEE publication

Title: An Architecture for a Policy-Oriented Web Browser Management System: HiFiPol: Browser

Conference Proceedings: Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual

Author: Ananth A. Jillepalli

Publisher: IEEE

Date: June 2016

Copyright © 2016, IEEE

LOGIN

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. Already a RightsLink user or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

Copyright © 2017 Copyright Clearance Center, Inc. All Rights Reserved. Privacy statement. Terms and Conditions.
Comments? We would like to hear from you. E-mail us at customercare@copyright.com

Figure 6.2: Permission from IEEE to reproduce an article as Chapter 3 of this thesis.