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

Ananth Jillepalli, Daniel Conte de Leon
Ctr. for Secure and Dependable Sys.
University of Idaho
Email: {ajillepalli, dcontedeleon}@ieee.org

Stuart Steiner
Ctr. for Secure and Dependable Sys.
University of Idaho
Email: ssteiner@vandals.uidaho.edu

Frederick T. Sheldon
Dept. of Computer Science
University of Idaho
Email: sheldon@ieee.org

***Abstract*—In this article, 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.**

I. Introduction

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 [1]. 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 paper.

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 [3]. Extensions or plug-ins add another vulnerability vector [4]. 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 [5]. 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 paper.

This paper is organized as follows: An expanded description of the problem that current organizations face when securing the web-based ecosystem is described in section II. Our approach and proposed solution, which the HERMES language is a part of, is described in section III. The characteristics, structure, and grammar of the HERMES language are described in detail in section IV. A case study of a fictional organization, that demonstrates and discusses the practical usage of the HERMES language, is presented in section VI. The integration of HERMES into the HiFiPol:Browser [5] policy management system is described in section VII. Related work is discussed in section VIII, with a description of differences between our work and previous related research or systems. Future work is presented in section IX. Finally, our conclusion is provided in section X.

II. 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 [6], [7]. 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 [8], [9]. 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.

III. 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.

The HiFiPol:Browser Policy Management System is divided into two subsystems: *Development* and *Deployment* [5]. Figure 1 is a pictorial representation of this. The policy

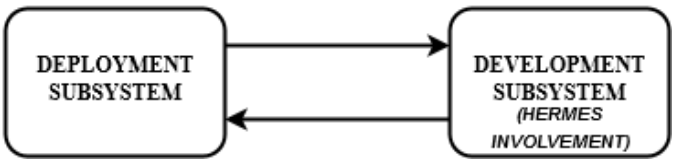

Figure 1: HiFiPol:Browser Subsystems

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 [5]. 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 [5]. Integration of HERMES into HiFiPol:Browser is described in section VII. For a more detailed description of the HiFiPol:Browser architecture we refer the reader to the work of Jillepalli et. al. [5].

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 [8], [9]; (2) We have developed a complete prototype, entitled "Open Browser GP", of a GUI-based tool that allows remote configuration of web browsers [8], [9]. 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. [5]. A detailed description of our plans for future research work is presented in Section IX.

IV. 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 [5].

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.

V. 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 [10], definite-clause [11], and block-like grammar format [12]. Most of the BNF (Backus-Naur Form) specification of HERMES is provided in Listing 1.

```
// Policy Rules:
// A policy is a set of domain
// or policy defining entities.

<EntityBlockListDefinition> :=
<RightSquare><EntityBlockList><LeftSquare>

<EntityBlockList> :=
<EntityBlock> |
<EntityBlock> , <EntityBlockList>

// Policy Entity Rules:
// A policy entity is comprised of
// an identifier, followed by a colon,
// then followed by a name. After the name
// a list of attributes is enclosed by { }.

<EntityBlock> :=
<EntityIdentifierName><Colon>
<EntityIdentifierValue>
<RightCurly>
<EntityMemberListDefinition>
<LeftCurly>

<EntityMemberListDefinition> :=
<RightSquare><EntityMemberList><LeftSquare>

<EntityMemberList> :=
<EntityMember> |
<EntityMember> , <EntityMemberList>

<EntityMember> :=
<FieldComponent> |
<AttributeComponent> |
<EntityComponent>

// Policy Entity Attribute Rules:
// A policy entity attribute is a
// field name and corresponding value
// separated by a colon.

<FieldComponent> :=
<FieldName><Colon><FieldValue>
```

Listing 1: HERMES BNF Notation

VI. 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.

A. HERMES Policy Composition

In HERMES, a policy is a set of relations between four different entities, each characterized by Figures 2, 3, 4, and 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:

B. Case Study Organizational Hierarchies

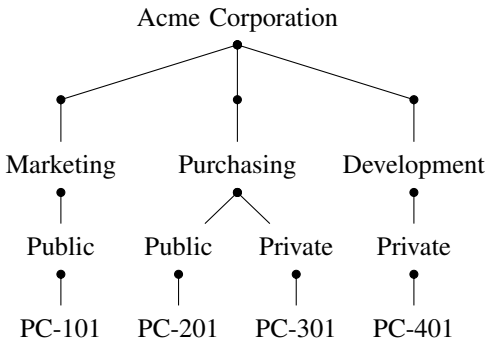

Figure 2: Organizational Hierarchy Graph.

The hierarchical representation os Acme's sub-domains and devices is represented in Figure 2.

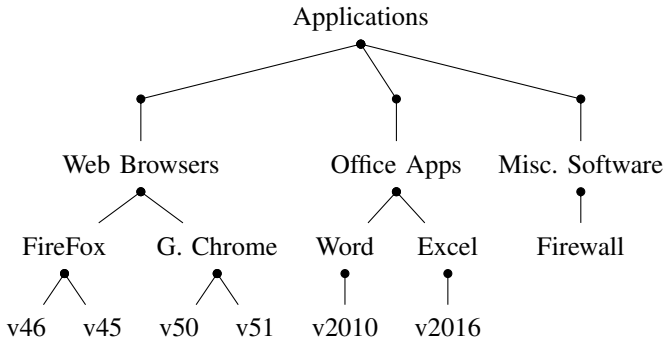

Figure 3: Application Hierarchy Graph.

Categorization of applications used by Acme and the respective versions of those applications, is represented in Figure 3.

The structure and grouping of users and their respective role assignment is represented by Figure 4.

A set of relevant configuration actions applicable to applications used by Acme are represented by Figure 5.

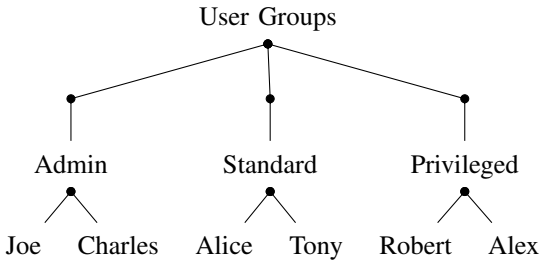

Figure 4: User-Group Hierarchy Graph.

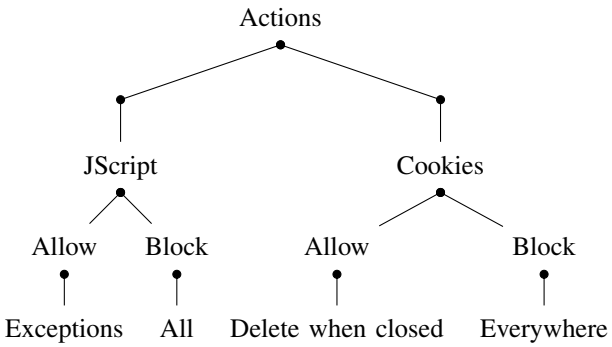

Figure 5: Possible-Actions Hierarchy Graph.

C. 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:

1) Using HERMES to specify domain data: Listing 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 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 2.

2) Using HERMES to specify action policies: Listing 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.

```
Domain: Acme
{
Description: "Acme Organization."
List: [MKT, PUR, DEV]
}

SubDomain: MKT
{
Description: "Marketing Department."
DeviceList: [PC-101]
Parent: Acme
}

SubDomain: PRCH
{
Description: "Purchasing Department."
DeviceList: [PC-201, PC-301]
Parent: Acme
}

SubDomain: DEV
{
Description: "Development Department."
DeviceList: [PC-401]
Parent: Acme
}
```

Listing 2: HERMES excerpt in specifying organizational domain knowledge for Acme.

Listing 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 3 and 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 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 IX).

```

Policy: PID_01_000
{
  Description: "Enabling JavaScript
  functionality."
  Rationale: "Marketing staff requires
  JavaScript for designing ads."
  Status: "Enabled"
  Field: JavaScript: "Enabled"
  ApplyTo: "MKT"
  ToApp: "APP.ALL"
}

Policy: PID_02_000
{
  Description: "Enabling all cookie
  functionality."
  Rationale: "Purchasing websites require
  cookie functionality."
  Status: "Enabled"
  Field: Cookie: "Enabled"
  ApplyTo: "PRCH"
  ToApp: "APP.ALL"
}

Policy: PID_02_001
{
  Description: "Disabling all cookie
  functionality."
  Rationale: "Purchasing websites require
  security against cookie injection
  attacks."
  Status: "Enabled"
  Field: Cookie: "Disabled"
  ApplyTo: "PRCH"
  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"
}
```

Listing 3: HERMES excerpt in specifying security policies for Acme 's applications.

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 VII and the process is shown in Fig. 7.

```
1
2  // Domain data facts
3  domainDescription('policy.yaml','Acme',
4  'Acme Corporation').
5  domainList('policy.yaml','Acme',
6  'MKT, PUR, DEV').
7  subDomainDescription('policy.yaml','MKT',
8  'Marketing Department.').
9  subDomainDeviceList('policy.yaml','MKT',
10 'PC-101').
11 subDomainParent('policy.yaml','MKT',
12 'Acme').
13 subDomainDescription('policy.yaml','PRCH',
14 'Purchasing Department.').
15 subDomainDeviceList('policy.yaml','PRCH',
16 'PC-201, PC-301').
17 subDomainParent('policy.yaml','PRCH',
18 'Acme').
19 subDomainDescription('policy.yaml','DEV',
20 'Development Department.').
21 subDomainDeviceList('policy.yaml',
22 'DEV','PC-401').
23 subDomainParent('policy.yaml','DEV',
24 'Acme').
25
26 // Policy specification facts
27 policyDescription('policy.yaml',
28 'PID_01_000','Enabling JavaScript
29 functionality.').
30 policyRationale('policy.yaml',
31 'PID_01_001','Marketing staff requires
32 JavaScript for designing ads.').
33 policyStatus('policy.yaml',
34 'PID_01_001','Enabled').
35 policyField('policy.yaml',
36 'PID_01_001','Cookie.Lifetime,"Disabled').
37 policyApplyTo('policy.yaml',
38 'PID_01_001','RLE.admin').
39 policyToApp('policy.yaml',
40 'PID_01_001','APP.Google_Chrome').
41 policyParent('policy.yaml',
42 'PID_01_001','PID_01_000').
```

Listing 4: Excerpt of policy knowledge-base, which is a collection of facts, instantiated from policies in listing 3.

3) Policy Knowledge-Base generated from HERMES: Listing 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 2) and policy specifications (like the ones given in listing 3). The knowledge-base is a collection of facts, which have been instantiated on the basis of given policies. In Listing 4, the facts reflect domain data of Listing 2 and policy `PID_01_000` from listing 3.

4) Final Instantiated Web Browser Configurations: Listing 5 demonstrates the final resultant configuration corresponding to the policies specified in listing 3. These configurations are rewritten after a series of instantiations similar to the one showed in listing 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).

```
1
2  For Mozilla Firefox browsers in
3  Marketing department's devices:
4  user_prefs("javascript.enabled", true);
5
6  For Mozilla Firefox browsers in
7  Purchasing department's devices:
8  user_prefs("network.cookie.
9        cookieBehavior", 0);
10 user_prefs("network.cookie.
11       cookieBehavior", 2);
12 user_prefs("network.
13    cookie.thirdparty.sessionOnly", false);
14
15 For Mozilla Firefox browsers in
16 Development department's devices:
17 user_prefs("javascript.enabled", false);
18
19 For Google Chrome browsers in
20 Marketing department's devices:
21 "default_content_setting_values":
22 {"javascript":0},
23 "default_content_settings":
24 {"javascript":0},
25
26 For Google Chrome browsers in
27 Purchasing department's devices:
28 "default_content_setting_values":
29 {"cookies":0,"cookies":2},
30 "default_content_settings":{"cookies":
31 0,"cookies":2},
32 "block_third_party_cookies":true,
33
34 For Google Chrome browsers in
35 Development department's devices:
36 "default_content_setting_values":
37 {"javascript":2},
38 "default_content_settings":
39 {"javascript":2},
```

Listing 5: Resultant actual web browser configurations corresponding to the policies given in listing 3.

Listing 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.

D. Deployment of Final Configurations

The final and fully-instantiated configurations are then remotely deployed by deployment component of HiFiPol:Browser [5].

1) Proposed Repository Structure: The final configuration files would be relocated to respective agent repositories on a server.

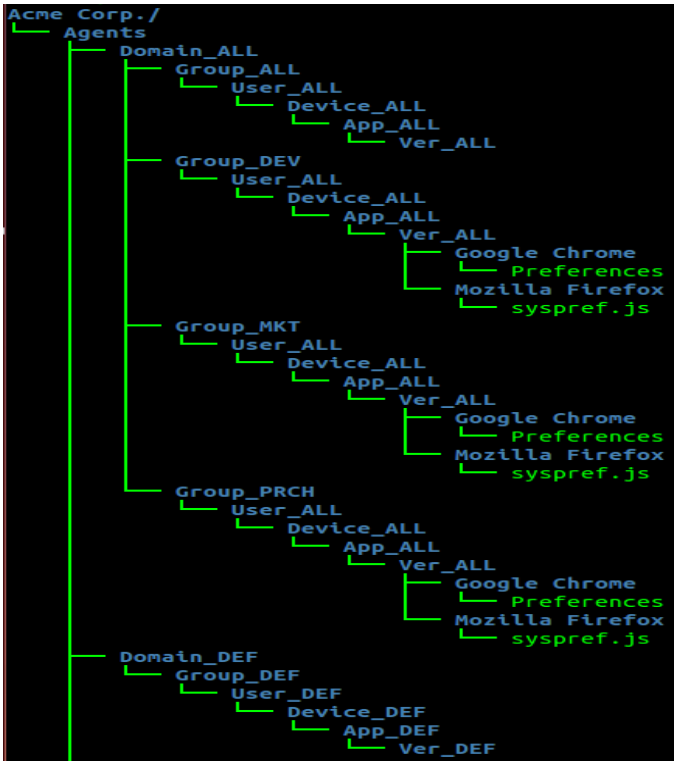

Figure 6: Proposed repository structure for configuration files deployment to agents through HiFiPol:Browser architecture design.

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 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 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 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. 6 correspond to unconditional policy application and default policy application, respectively. These will be discussed in detail in future articles.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.

VII. Integration into the HiFiPol Framework

HERMES has been designed to enable text-based policy specification functionality in the framework of HiFiPol:Browser [5]. HERMES acts as the foothold medium between humans and machine understandable facts, which will be further processed into deployable configurations.

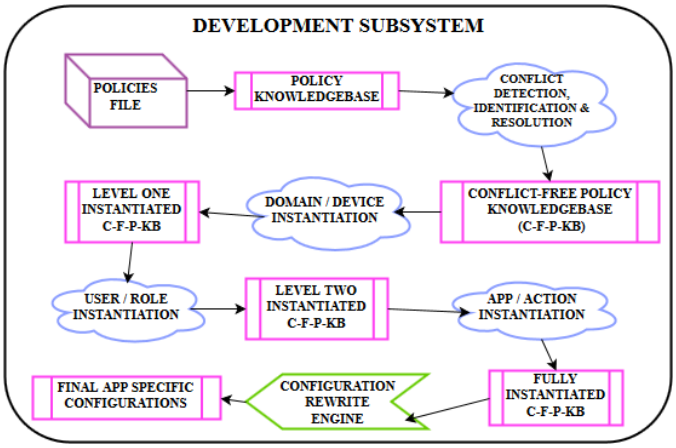

Figure 7: HiFiPol:Browser Development Subsystem

Fig. 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 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 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 VI-D1. 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.

VIII. Related Research

Several XML-based languages have been created for specification of configurations. The most relevant to our work are: Security Policy Assertion Language (SecPAL) [13], Open Vulnerability and Assessment Language (OVAL), eXtensible Access Control Markup Language (XACML) [14], Extensible Configuration Checklist Description Format (XCCDF) [15], Margrave (focused on firewall analysis) [16], [17], and Security Policy Language (SPL) [18]. Some of these are part of the Security Content Automation Protocol (SCAP) specification [19].

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" [18] 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 article. 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 article 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" [16], [17], 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.

IX. Future Work

In the recent past we have performed research work that resulted in the Open Browser GP tool [8], [9] 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.

X. 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 article. The specific needs which HERMES is designed to address, and the trade-offs undertaken during design of HERMES are discussed in section IV. An example case study was introduced that demonstrates HERMES' usage. The case study (section VI) 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 HERMES into the HiFiPol:Browser system was given in section VII. A survey of related research was presented in section VIII, describing the similarities and differences between HERMES and other existing configuration languages. Finally, in section IX, 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* [20], [21] secure design principles for browser security.

Acknowledgments

We would like to thank the U.S. National Science Foundation (NSF) for partially funding this research work under NSF award number 1027409 and 1565572. The State of Idaho under an IGEM grant for Cybersecurity Capacity Building also partially funded this research work. We would also like to thank the program committee, conference chairs, and reviewers for their help improving this paper. The opinions expressed in this paper are not those of the NSF or the State of Idaho.

References

[1] "2016 - state of the cloud report," http://www.rightscale.com/press-releases/rightscale-2016-state-of-the-cloud-report, Feb. 2016, RightScale Incorporation.

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

[3] 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.

[4] 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.

[5] A. A. Jillepalli and D. C. de Leon, "An architecture for a policy-oriented web browser configuration management system - HiFiPol: Browser," in *Proc. IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, Atlanta, Georgia, USA, 2016.

[6] 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.

[7] 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.

[8] 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.

[9] D. C. de Leon, V. A. Bhandari, A. Jillepalli, and F. T. Sheldon, "Using a knowledge-based security orchestration tool to reduce the risk of browser compromise," in *Proc. IEEE 07th Symposium Series Computational Intelligence (SSCI)*, Athens, Greece, 2016.

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

[11] 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.

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

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

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

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

[16] 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.

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

[18] 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.

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

[20] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts - 8th Edition*. John Wiley & Sons, Inc., 2009.

[21] 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.