Using a Knowledge-based Security Orchestration Tool to Reduce the Risk of Browser Compromise

Daniel Conte de Leon
Ctr. Secure and Dep. Sys.
University of Idaho
dcontedeleon@ieee.org

Venkata A. Bhandari
Ctr. Secure and Dep. Sys.
University of Idaho
bhan1451@vandals.uidaho.edu

Ananth Jillepalli
Ctr. Secure and Dep. Sys.
University of Idaho
ajillepalli@ieee.org

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

***Abstract*—Today, web browsers are used to access and modify sensitive data and systems including intranets and critical control systems. Due to their computational capabilities and network connectivity, browsers are vulnerable to several types of attacks, even when fully patched. Browsers are also the main target of phishing attacks. Many browser attacks, including phishing, could be prevented or mitigated by using site-, user-, and device-specific security configurations in a diverse browsing ecosystem. However, in our research, we discovered that all major browsers expose disparate security configuration procedures, option names, values, and semantics. This results in an extremely hard to secure browsing ecosystem. We analyzed in detail more than a thousand browser security configuration options in three major browsers and found that only 17 had common names with common semantics. In this paper, we describe the results of this in-depth analysis. We also describe a knowledge-based solution, Open Browser GP, that would enable organizations to implement highly-granular secure configurations for their information and operational technology (IT/OT) browsing ecosystem.**

I. Introduction

Malicious actors have been targeting web browsers because they are currently one of the most exposed parts of the enterprise Information Technology and Operational Technology (IT/OT) ecosystem. Web browsers, when configured by default, are vulnerable to various types of attacks such as Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and other JavaScript code injection and data exfiltration and modification attacks. These attacks can be carried-out even when a browser has been fully patched. Browsers are also the primary target and compromise entry-point for Phishing and Spear-Phishing attacks.

Using a victim's web browser, when configured by default, malicious actors would only need victims to click on a link to execute attacker controlled code. This attacker controlled code has the potential for exfiltrating and modifying confidential data using the current user's credentials or permissions. In some cases, such unauthorized access may include changes to critical or control systems with far reaching consequences. Data exfiltration attacks of this kind are extremely difficult to detect because the unauthorized access is blended with normal system operations and network access. In addition, a data modification attack performed in this way may also bypass log-based checks since system access appears to be from a valid user session. Is is also noteworthy that all of these attacks bypass firewall rules, since they are application level attacks and users initiate the connections to the outside world using common protocols, such as HTTP and HTTPS.

The use of a diverse browser ecosystem coupled with high-granularity and tailored secure configuration, rather than the current prevalent default browser security configurations, can help prevent and mitigate most of these attacks. By using tailored secure configuration of browsers, for example, restricting execution of Remote JavaScript code and Permanent Browser Storage to a trusted combination of client, server, domain, user/role, and client browser, in combination with a diverse browser ecosystem, these attacks could be prevented and mitigated. An example of such a high-granularity client browser configuration would be: *enabling JavaScript for user John, on JohnsPC, while using Internet Explorer, and when accessing intranet.example.edu and disabling JavaScript and access to all other sites using the same browser for the same user, combined with enabling JavaScript on Firefox, while preventing access to the intranet*. Browser features such as JavaScript code execution and permanent storage cannot be disabled using a blanket enterprise-wide configuration because this would most likely render most trusted sites unusable. Almost all cloud-based and enterprise-based systems today require browsers to have most functionality enabled. However, using today's approach of default browser configurations where the full set of browser functions are available to all sites, trusted and untrusted, results in an insecure and almost impossible to secure browsing ecosystem. High-granularity, enterprise-wide, security-tailored web browser configuration is greatly needed to adequately secure the browsing ecosystem in the enterprise.

About two years ago, we embarked on the task of determining what was needed in order to enable organizations to be able to configure their client browser ecosystems in a secure manner by using a tailored and high-granularity set of configurations. We wanted to find out why organizations were not already doing this and what we could do to help. We began by performing the needed research to answer the followed questions:

1) Which security-relevant configuration options are available in today's major browsers?
2) How similar or different are the available configuration options in the major browsers?
3) Which are the procedures and knowledge needed in order to configure these available options?

4) How similar or different are the procedures and know-how needed to configure the available options?
5) What tools are available today to help IT/OT system administrators to configure browsers using a high-granularity approach?

The answers to all of these questions where reported in a Master's Thesis publication [1]. This paper presents a summary of some of the findings.

II. Detailed Analysis of Security Configuration Options in Web Browsers

A. Classification of Configuration Options

Since our focus was the configuration of security settings, to reduce the number of configuration options that we needed to analyze and compare to a manageable number, we first classified all available configuration options. See Table I for a list of categories. We classified Machine-Level configuration options of each major browser into GUI or NO-GUI and SECurity or NO-SECurity related. Results of this classification are shown in Table II. Machine-Level configuration options refer to configuration options that apply to a complete device or machine, rather than to the individual users of that device. In order to create this classification we analyzed 770 Internet Explorer, 158 Google Chrome, and 263 Mozilla Firefox configuration options. After classifying these configuration options, we did not analyze configuration options that were in the NON-SECurity category. Then, we used Python scripts to extract the data and the predetermined configuration option names from ADMX and ADML files of Internet Explorer and Google Chrome. Firefox configuration options were extracted by processing the entries described in the Mozilla database describing Firefox internal configuration options (about:config).

To make the scope of our task manageable we chose to work on the following three browsers, which we refer as major browsers: Internet Explorer (IE), Google Chrome (Chrome), and Mozilla Firefox (Firefox). These are the 3 most popular desktop web browsers as reported by StatCounter [2]. The versions we used for this research and associated experiments were: Internet Explorer version 10.0.9200, Google Chrome version 37.0.2062, and Mozilla Firefox version 33.0.2. We believe that no significant change in the naming and semantics of available configuration options has occurred for this browsers since the versions we analyzed. The problem we are describing in this article with respect to the variability of names and semantics in Browser configuration options is still very much unsolved.

Name	Description
GUI-SEC	GUI and Security setting.
GUI-NO-SEC	GUI and Non-Security setting.
NO-GUI-SEC	Non-GUI and Security setting.
NO-GUI-NO-SEC	Non-GUI and Non-Security sett.

Table I: Web Browser Configuration Analysis Categories

Browser	Classification	Count
Internet Explorer	GUI-SEC	260
	GUI-NO-SEC	300
	NO-GUI-SEC	210
	NO-GUI-NO-SEC	N/A
	Total	770
Google Chrome	GUI-SEC	58
	GUI-NO-SEC	78
	NO-GUI-SEC	22
	NO-GUI-NO-SEC	N/A
	Total	158
Mozilla Firefox	GUI-SEC	47
	GUI-NO-SEC	147
	NO-GUI-SEC	69
	NO-GUI-NO-SEC	N/A
	Total	263

Table II: Number of Analyzed Web Browser Settings per Category and per Major Web Browser

B. Common Configuration Options

After analyzing Machine Level configuration options in the three most popular desktop web browsers we were able to determine which options where common. The results of the mapping of similar configuration options is shown in Tables IV and V. Among these 17 settings, only six are common to all three major browsers and the rest are common between any two. This is a policy-to-configuration option mapping, which shows corresponding configuration options in different browsers.

Each row in Tables IV and V is numbered in ascending order and consists of two sub-rows. The first column of the first sub-row consists of a common *All Browsers* configuration option category that we created to group all semantically similar options. The second, third, and fourth columns of the first sub-row present configuration option names with respect to Internet Explorer, Google Chrome, and Mozilla Firefox respectively. The second sub-row of each row provides a description of configuration option. This description was created manually for each common configuration option based on the descriptions for each browser option. N/A indicates that

Category	Classification	Count
All_-Browsers	GUI-SEC	8
	GUI-NO-SEC	7
	NO-GUI-SEC	2
	NO-GUI-NO-SEC	N/A
	Total	17

Table III: Number of Syntactically and Semantically Equivalent Settings across the Three Major Desktop Browsers per Category. These 17 configuration options are described in Tables IV and V.

EN	All Browsers	IE	Chrome	Firefox
	Description			
1	*Cache_Size_Setting*	DefaultDomainCacheLimitInMB	DiskCacheSize	*Cache_Size*
	Used to set the cache size in Internet Explorer, Google Chrome and Mozilla Firefox Browsers. In Internet Explorer it is set in MB, whereas in Google Chrome and Firefox it is set in KB. So set this configuration option setting according to the requirements and available system cache size. It corresponds to (Set default storage limits for websites) in Internet Explorer, (Set disk cache size in bytes) in Google Chrome and (Set Browser Cache Size) in Mozilla Firefox.			
2	*CrashRestore*	DisableACRPrompt	*N/A*	*Crash_restore*
	Allows us to configure the browser to prompt when the browser tries to recover from any crash sessions. It corresponds to (Turn off Automatic Crash Recovery) in Internet Explorer and (Crash Recovery) in Mozilla Firefox.			
3	*DNSPrefetching*	*N/A*	DnsPrefetchingEnabled	*DNS*
	Used to activate or deactivate DNS prefetching. If we enable this setting DNS prefetcing is activated and deactivated if we disable this setting. It corresponds to (Enable network prediction) in Google Chrome and (Disable DNS Prefetching) in Mozilla Firefox.			
4	*Default_Browser_Check*	*N/A*	DefaultBrowser- SettingEnabled	*Check_Default_- Browser*
	Configures to check whether the browser is the default browser in a given system in Google Chrome and Mozilla Firefox Browsers. If we enable this setting then the browser will prompt if it is not the default browser. It corresponds to (Set Chrome as Default Browser) in Google Chrome and (Check if firefox is the default browser) in Mozilla Firefox.			
5	*DeveloperTools*	DisableDeveloperTools	DeveloperToolsDisabled	*N/A*
	Configures whether a browser allows or disallows access to developer tools. If we enable this configuration option developer tools cannot be accessed by a user in Internet Explorer and Google Chrome. It corresponds to (Turn off Developer Tools) in Internet Explorer and (Disable Developer Tools) in Google Chrome.			
6	*Display_Images*	*N/A*	DefaultImagesSetting	*Permission_Images*
	Configures whether we can display images or not while pages load in Google Chrome and Mozilla Firefox Browsers. It corresponds to (Default images setting) in Google Chrome and (Allow or disallow images to load) in Mozilla Firefox.			
7	*Download- DirectorySetting*	*N/A*	DownloadDirectory	*Download_Dir*
	Used to set the download directory of the browser. It corresponds to (Set download directory) in Google Chrome and (Set Download Directory) in Mozilla Firefox.			
8	*Geo_Location_Setting*	GeolocationDisable	DefaultGeolocationSetting	*Geo_Location*
	Configures whether a browser can track GEO location of the system. If we enable this setting then GEO location is tracked by websites and disallowed if it is disabled in Internet Explorer, Google Chrome and Mozilla Firefox Browsers. It corresponds to (Turn off browser geolocation) in Internet Explorer, (Default geolocation setting) in Google Chrome and (Setting to enable or disable GEO location) in Mozilla Firefox.			
9	*HomePage*	*N/A*	HomepageLocation	*Home_Page*
	Configures home page of Google Chrome and Mozilla Firefox Browsers. It corresponds to (Configure the home page URL) in Google Chrome and (Home Page) in Mozilla Firefox.			
10	*JavaScript*	IZ_PolicyActiveScripting_1	DefaultJavaScriptSetting	*JavaScript- Enabled*
	Configures whether Javascript is enabled or disabled in Internet Explorer, Google Chrome and Mozilla Firefox Browsers. It corresponds to (Allow active scripting) in Internet Explorer, (Default JavaScript setting) in Google Chrome and (Setting to enable or disable Javascripts) in Mozilla Firefox. Internet Explorer has the same setting in different zones, we are using the setting available in (Internet Zone). If we want to map this setting to other zones we can change the mapping configuration option name in the database and change the zone name in the description to avoid confusion.			
11	*Max_Proxy_Setting*	*N/A*	MaxConnectionsPerProxy	*Max_Proxy*
	Used to set the maximum number of connections per proxy in Google Chrome and Mozilla Firefox Browsers. It corresponds to (Maximal number of concurrent connections to the proxy server) in Google Chrome and (Set maximum number of connections to proxy server) in Mozilla Firefox.			

Table IV: All Found Syntactically and Semantically Similar Configuration Options in the Three Major Browsers, Part A

EN	All Browsers	IE	Chrome	Firefox
	Description			
12	*Plugin_Prompt_Setting*	IZ_PolicyRunActiveXControls_1	DefaultPluginsSetting	*Plugin_Prompt*
	Configures whether a browser should run plugins only after click or run plugins automatically. If we enable this configuration option then we will get a prompt to run plugins in Internet Explorer, Google Chrome and Mozilla Firefox Browsers. It corresponds to (Run ActiveX controls and plugins) in Internet Explorer, (Default plugins setting) in Google Chrome and (Setting to run plugins only on click) in Mozilla Firefox. Internet Explorer has same setting in different zones, we are using the setting available in (Internet Zone). If we want to map this setting to other zones we can change the mapping configuration option name in the database and change the zone name in the description to avoid confusion.			
13	*Plugin_Setting*	IZ_PolicyRunActiveXControls_1	DefaultPluginsSetting	*N/A*
	Configures whether a browser allows or disallows plugins. If we enable this configuration option all plugins can run in Internet Explorer and Google Chrome. It corresponds to (Run ActiveX controls and plugins) in Internet Explorer and (Default plugins setting) in Google Chrome. Internet Explorer has same setting in different zones, we are using the setting available in (Internet Zone). If we want to map this setting to other zones we can change the mapping configuration option name in the database and change the zone name in the description to avoid confusion.			
14	*PopUpBlocker*	IZ_PolicyBlockPopupWindows_1	DefaultPopupsSetting	*PopUpsDisabled*
	Configures whether pop-ups are allowed or disallowed in Internet Explorer, Google Chrome and Mozilla Firefox Browsers. It corresponds to (Use Pop-up Blocker) in Internet Explorer, (Default popups setting) in Google Chrome and (Setting to configure Pop-ups) in Mozilla Firefox. Internet Explorer has same setting in different zones, we are using the setting available in (Internet Zone). If we want to map this setting to other zones we can change the mapping configuration option name in the database and change the zone name in the description to avoid confusion. If this setting is disabled it will allow pop-ups on white-listed pages in Mozilla firefox, it was mapped in that manner since it was a recommended setting. If you want to disable it on all sites change the disabled mapping value for Mozilla firefox to 3 instead of 2.			
15	*PrintSetting*	NoPrinting	PrintingEnabled	*N/A*
	Used to allow or disallow printing in Internet Explorer and Google Chrome Browsers. If we enable this setting then the user can print a webpage or document from the specified browser and if it is disabled users cannot print. It corresponds to (Turn off Print Menu) in Internet Explorer and (Enable printing) in Google Chrome.			
16	*Restore_Previous_Session*	ContinuousBrowsing	RestoreOnStartup	*Start_Up_Pages*
	Configures the browser such that, it restarts with the web pages from last browsing session. If we enable this setting the browsers will restart with last browsing session and if we disable this setting they will start with a blank page in Internet Explorer, Google Chrome and Mozilla Firefox Browsers. It corresponds to (Start Internet Explorer with tabs from last browsing session) in Internet Explorer, (Action on startup) in Google Chrome and (Set how the browser should start) in Mozilla Firefox.			
17	*SafeBrowsing- Setting*	*N/A*	SafeBrowsingEnabled	*Safe_Browsing*
	Used to activate or deactivate safe browsing to detect phishing malware while loading websites. If we enable this setting safe browsing is activated and deactivated if we disable this setting. It corresponds to (Enable Safe Browsing) in Google Chrome and (Enable Safe Browsing) in Mozilla Firefox.			

Table V: All Found Syntactically and Semantically Similar Configuration Options in the Three Major Browsers, Part A

this setting is not available in the corresponding browser [1].

Let's consider a scenario where we need to check the configuration option names corresponding to JavaScript in the three major web browsers. By observing the descriptions of each configuration option we can find that JavaScript appears in row 10 of Table IV. In this row, the common *JavaScript* label appears within the *All Browsers* category, the *IZ_PolicyActiveScripting_1* label corresponds to the Internet Explorer configuration option name, the *DefaultJavaScriptSetting* label corresponds to the Google Chrome configuration option name, and the *JavaScriptEnabled* label corresponds to the Mozilla Firefox configuration option name.

C. Analysis of Dissimilar Configuration Options

This section presents the results of analyzing dissimilarities between all available configuration options in all three major desktop web browsers. Some of the configuration options may have similar semantics but they may implement these semantics with varied degrees of disparity. We extracted, analyzed, and compared only security related settings for each browser. Three different tables comparing the names and semantics of each browser were the result of this analysis. The complete

RN	Chrome	IE	Firefox
	Description		
75	URLBlacklist (Block access to a list of URLs)	*Similar semantics can be achieved by modifying multiple settings at different zones*	*Similar semantics can be achieved by using third party Add-ons*
	Blocks access to the listed URLs and prevents the user from loading web pages from blacklisted URLs.		

Table VI: An Excerpt of Results of the Comparison of Security Related Settings for Google Chrome with Respect to Internet Explorer and Mozilla Firefox

RN	Firefox	IE	Chrome
	Description		
3	DNS (Disable DNS Prefetching)	*N/A*	*DnsPrefetchingEnabled (Enable network prediction)*
	Perform DNS prefetching proactively.		

Table VII: An Excerpt of Results of the Comparison of Security Related Settings for Mozilla Firefox with Respect to Internet Explorer and Google Chrome

RN	IE	Chrome	Firefox
	Description		
76	PopupBlocker_AllowList (Pop-up allow list)	*Similar semantics can be achieved by modifying multiple settings*	*Similar semantics can be achieved by using third party Add-ons.*
	Allows you to specify a list of web sites that will be allowed to open pop-up windows regardless of the Internet Explorer processs Pop-Up Blocker settings.		

Table VIII: An Excerpt of Comparison of Security Related Settings for Internet Explorer with Respect to Google Chrome and Mozilla Firefox

tables and extended descriptions of each configuration option are available in a Master's thesis by one of the authors [1].

Each row in Table VI is numbered with the corresponding configuration option number and consists of two sub-rows. The first column of the first sub-row consists of a Google Chrome configuration option name followed by its display name in parenthesis, the second and third column of the first sub-row provide information about the possible ways of configuring a similar setting in Internet Explorer and Mozilla Firefox. The second sub-row of each row provides a description of the Google Chrome setting.

Similarly, we created Table VII and Table VIII to describe can compare in detail security settings of Firefox and IE, respectively. In this paper, we present an excerpt from the complete tables. These tables will be a useful reference to verify whether a similar configuration in multiple browsers can be accomplished by modifying single or multiple available settings. In these excerpts, the text in `monospace font` is used to represent the difference between mechanically extracted data and the manual data entries. The full tables have 76, 112, and 171 entries for Chrome, Firefox, and Internet Explorer, respectively [1].

III. Current Tools for Remote Browser Configuration

A. Microsoft Group Policy and Domain Services

Administrative Templates and Group Policy Objects are the most popular way of configuring Windows Client systems in the enterprise. ADMX and ADML are the two types of administrative template files. ADMX files contain settings and ADML files contain textual descriptions of each setting [3]. Group Policy Objects and ADMX files can used in combination with an Active Directory Domain Services infrastructure, based on Windows Server, to configure client systems remotely, when the clients have been joined to a Windows Domain. The *Local Group Policy Editor* can be used to configure client settings locally [4]. The process of local and remote configuration using ADMX files and Group Policy Objects is a complex task carried-out by trained and experienced system administrators.

For Internet Explorer (IE), ADMX files are available in the *Policy Definitions* folder in all Windows operating systems and these files when IE is updated. For Google Chrome, ADMX and ADML files are not installed by default. However, they can be acquired from the Google Chrome developer website [5], [6]. Mozilla Firefox does not natively read settings from the Windows Registry and requires a third party add-on *GPO for Firefox* to connect Firefox to the Windows registry [7], [8]. In addition, ADMX files for Firefox are not created and maintained by Mozilla and most settings are not available unless different ADMX files are obtained from third-party developers.

The drawbacks of remote configuration using Group Policy and Domain Services are: 1) System administrators must still create the corresponding ADMX files or Group Policy Objects for each browser or application or manually configure each

browser in each client using the *Group Policy Management Console*. For example, if a system administrator is given the task to disable JavaScript in all browsers within an organizational unit, even though all browsers implement the JavaScript Enable/Disable setting, because they use different configuration option name, values, and procedures, separate ADMX files must be created and maintained for each; 2) These tools and procedures cannot be used to remotely configure Web Browsers in other popular client platforms such as Linux, Mac OS X, Apple iOS, and Android. As a result, most organizations use Group Policy to configure a few Windows System settings but not application level settings such as Web Browser security settings.

It may be possible for a few organizations to restrict client systems to Windows and Web Browser usage to Internet Explorer; Then, if having the adequate expertise, remotely configure this browser using Group Policy and Domain Services. However, most organizations allow users to use their preferred web browser and have a diverse client ecosystem. In addition, the use of a unique browser could make it more difficult to prevent Phishing attacks. Moreover, such environment may be more difficult to enforce in the future with the growth of mobile device usage and the system and application diversity that these bring.

B. FreeIPA

In order to expand the administrative tools to configure security configuration in Linux networked environment Red Hat [9] created FreeIPA [10]. FreeIPA provides remote configuration functionality for Linux systems. The drawbacks of FreeIPA for configuring security settings in multiple browsers in multiple platforms are: 1) FreeIPA requires a Windows Server with Active Directory Domain Services in order to configure browser settings in Windows clients. 2) The configuration workflow forces system administrators to only use FreeIPA and use Windows Server as an automated proxy; 2) Similarly to the Windows environment, FreeIPA does allow the configuration of different browsers using the same high-level configurations.

C. Dell KACE

The Dell KACE Systems Management Appliance is a tool for remote management of client devices in the enterprise. It is the only tool we found that can remotely manage a multi-platfrom IT/OT infrastructure. It provides: Inventory and IT/OT asset management, Systems deployment, Patch management and security, and Service desk. The Dell KACE Systems Management Appliance allows system administrators to create a client-server infrastructure in order to deploy software, run scripts, and manage security patches. It provides a secure DNS infrastructure and a user friendly GUI interface for systems administrators [11].

However, like all current remote management and configuration tools known to the authors, the remote deployment of configurations is based on running scripts for configurations manually developed by system administrators. To implement high-granularity configuration options on multiple browsers in multiple platforms system administrators would need to manually create separate scripts for each browser and each configuration group. This means that organizations would still need systems administrators with in-depth knowledge of the naming and semantics of all configuration options for all browsers used within their organization.

The authors do not know of any publications reporting on case studies that describe successful and complete prevention and damage mitigation from Phishing and Spear-Phishing attacks and compromises through web browsers, in organizations with a diverse device and platform ecosystem and that allow the use of multiple client browsers, with mostly unrestricted access to the Internet.

IV. Related Work

Accuvant Labs performed a comparative analysis on the security of Google Chrome, Internet Explorer, and Mozilla FireFox browsers [12]. Accuvant Labs analyzed the effectiveness of browser features such as Sandboxing, Plug-in Security, JIT Hardening, and URL Blacklisting, with respect to other major browsers. The primary focus in their analysis was to provide information about which browser was more effective at implementing each security feature.

By contrast, our analysis was focused on discovering the syntactic, semantic, and procedural differences between available configuration options in the three major desktop web browsers. We found that, there is a lack of homogeneity, syntactically, semantically, and procedurally, with respect to browser configurations. If a systems administrator is to be able to remotely configure all major client browsers used within an enterprise IT/OT infrastructure it may need to become an expert on each and all web browsers. Even when using the leading remote client configuration tools. This situation makes it almost impossible, for most organizations, to keep the browsing infrastructure secure.

V. Open Browser GP

After our detailed analysis of configuration options, and in order to help solve the problem of configuring secure browsing settings in major browsers within a diverse enterprise, we developed a prototype tool called Open Browser GP. In this section, we describe *Open Browser GP: A Multiplatform and Multibrowser Enterprise Web Browser Setting Configuration Tool*.

A. Development of Open Browser GP in Brief

Firstly, we mechanically created an Erlang fact-base of available configuration options by collecting configuration settings from ADMX and ADML files of Internet Explorer and Google Chrome; We extracted default *about:config* entries for Mozilla Firefox using Python scripts. Secondly, we created a Web-server application using Yaws in Erlang [13] in Ubuntu to provide a Web-based GUI for Open Browser GP. Thirdly, based on Browser policies provided by the user through the use of the Open Browser GP GUI we automatically created batch scripts that would configure the client browsers. Finally,

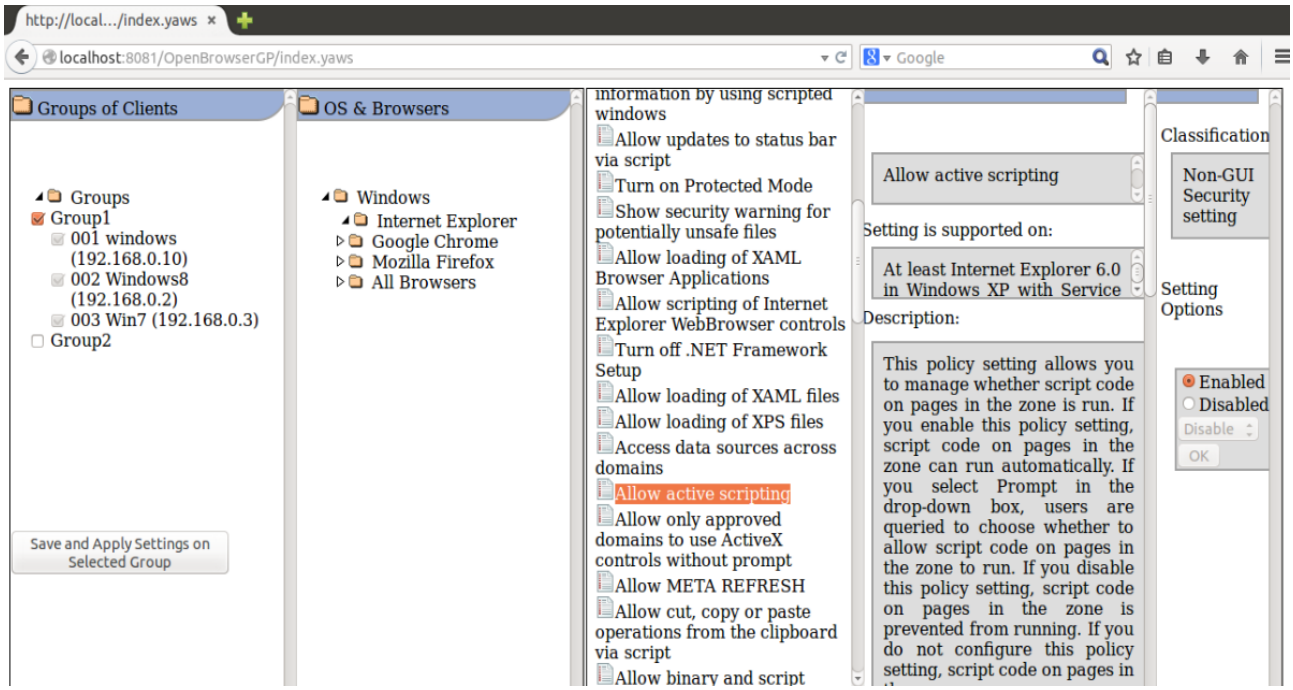


Figure 1: Open Browser GP: A Multiplatform and Multibrowser Policy Configuration Tool

we used OSSEC [14] to transfer and execute the corresponding batch scripts in a Windows client from the Open Browser GP Ubuntu server which also run the OSSEC Server, Yaws, and Erlang services.

B. Open Browser GP Tool Web Interface

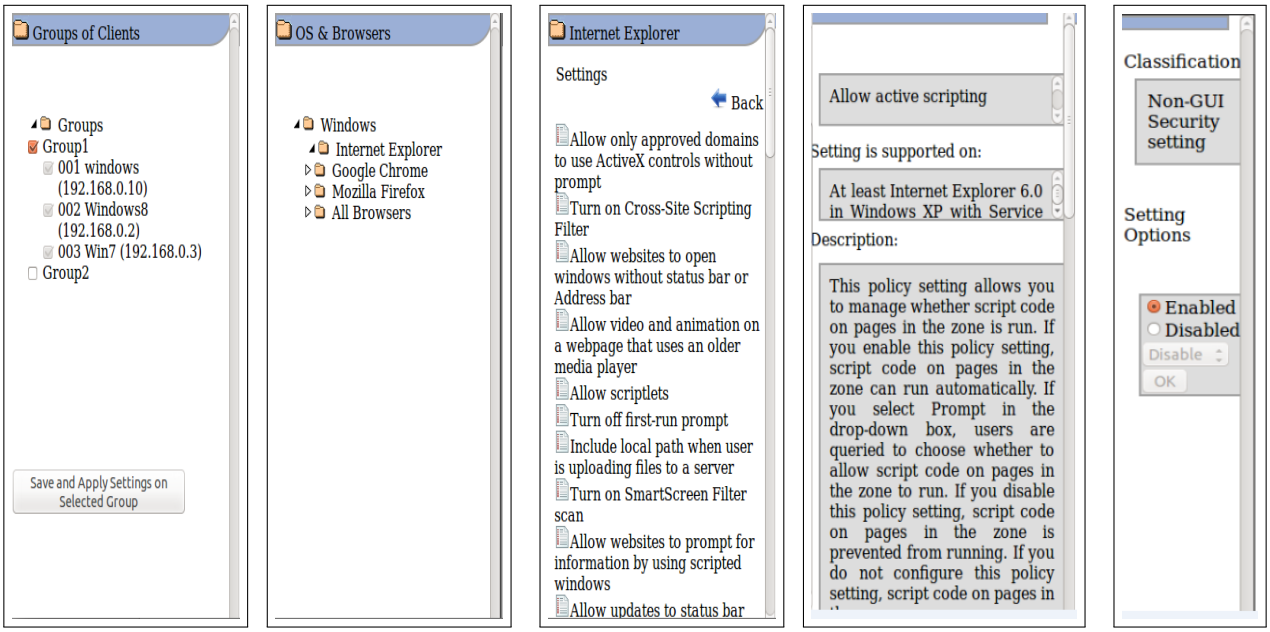
We designed the Open Browser GP web interface based on the single-screen concept. Figure 2 shows the web interface for Open Browser GP. In that Figure, we can observe that the GUI screen is divided into five sections:

1) *Client Groups* section as shown in Figure 2a allows the selection one of the groups of client systems connected to the server. Groups can be used to configure a given set of configurations for all devices in that group.
2) *OS and Browsers* section as shown in Figure 2b consists of operating systems and browsers for the system administrators to select required options.
3) *Settings in Corresponding Browsers* section as shown in Figure 2c consists of the available settings for each individual browser.
4) *Description of Browser Settings* section as shown in Figure 2d consists of configuration option display name of the selected setting, supported version, and description, users can hide this section by clicking on the standard view option.
5) *Classifications and Browser Setting Options* section as shown in Figure 2e consists of classification for each configuration option and different possible options to configure a selected setting.

The example showing in Figure 2 was used to configure the *Allow Active Scripting* configuration in Internet Explorer. In order to configure this setting using Open Browser GP system administrators need to follow these steps: 1) Load the web interface of Open Browser GP. 2) Select the required group of clients to configure, where each group can consist of a single client or a set of clients. 3) Select the required browser, in this case, Internet Explorer. 4) Select the required setting, in this case, Allow Active Scripting. Users can observe the selected setting description in the *Description of Browser Settings* section. 4) Depending on the requirement to enable or disable this configuration option, users can select accordingly in the *Classifications and Browser Setting Options* section. 5) Finally, the user must click on *Save and Apply Settings on Selected Groups*. This will save the new configuration and push the new configuration to the selected group of remote clients.

C. Advantages and Limitations of Open Browser GP

Open Browser GP enables IT/OT system administrators to remotely configure security configuration options in a diverse browsing ecosystem. Open Browser GP consolidates policy-equivalent configuration options from different browsers into one common configuration option. It uses OSSEC to connect to the remote clients for secure and authenticated communications. The current limitations of this prototype are: 1) Can only configure major browsers in Windows; 2) The OSSEC client cannot read user-level registry entries in client machines, hence Open Browser GP cannot configure user-level settings; 3) Open Browser GP uses the central deployment technique in OSSEC to transfer batch scripts automatically, hence, currently, scripts are transferred to all clients connected to the OSSEC server; 4) A third party add-on called is needed



(a) Client Groups (b) OS and Browsers (c) Settings in Corresponding Browsers (d) Description of Browser Settings (e) Classifications and Browser Setting Options

Figure 2: Individual Sections of the Open Browser GP Tool GUI: (a) Client Groups, (b) OS and Browsers, (c) Available Settings, (d) Description of Browser Settings and (e) Browser Settings

to modify configuration settings in Firefox. This is because Firefox does not natively read configuration options from the Windows Registry.

VI. Future Work

Encouraged by the results of the research work presented here, we are continuing to investigate and develop tools to help IT/OT system administrators implement and maintain a secure browsing ecosystem across the enterprise. The next generation of these tools called HiFiPol: Browser is already under development and more details may be found in [15] and [16].

Acknowledgments

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

References

[1] 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, U.S.A., May 2015. [Online]. Available: http://digital.lib.uidaho.edu/cdm/singleitem/collection/etd/id/864

[2] (2014, December) Statcounter global web statistics. Web Graphs. [Online]. Available: http://gs.statcounter.com/

[3] J. Moskowitz, "Inside adm and admx templates for group policy'," http://technet.microsoft.com/en-us/magazine/2008.01.layout.aspx, January 2008.

[4] J. Herman, "Managing group policy admx files step-by-step guide," http://msdn.microsoft.com/en-us/library/bb530196.aspx#manageadmxfiles_topic2, June 2007.

[5] J. Stromberg. (2013, August) Configuring google chrome via group policy. http://jackstromberg.com/2013/08/configuring-google-chrome-via-group-policy/.

[6] National Security Agency - Central Security Service, "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, October 2012.

[7] Redkitten Corp., "How to install a firefox add-on," http://www.redkitten.co.uk/firefox/how-to-install-a-firefox-add-on-for-all-users-remotely/.

[8] MozillaZine Knowledge Base, "About:config entries," http://kb.mozillazine.org/About:config_entries.

[9] Red Hat Inc., "Red Hat Inc." http://www.redhat.com/en, 2016.

[10] ——, "Freeipa," http://www.freeipa.org/page/About/, November 2014.

[11] Dell Software, "Dell kace: Systems management appliance," https://software.dell.com/kace/, June 2016.

[12] A. Labs, "Browser security comparison - a quantitative approach," http://files.accuvant.com/web/files/AccuvantBrowserSecCompar_FINAL.pdf, December 2011.

[13] Z. Kessin, "Building web applications with erlang," June 2012.

[14] A. Hay, D. Cid, and R. Bray, "Ossec host-based intrusion detection guide," 2008.

[15] A. Jillepalli and D. Conte de Leon, "An architecture for a policy-oriented web browser configuration management system: Hifipol: Browser," in *Proc. of the 40th IEEE Annual Computer Software and Applications Conference (COMPSAC-2016)*. IEEE, June 2016.

[16] 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. of the 2016 IEEE Symposium Series on Computational Intelligence (IEEE-SSCI-2016)*. IEEE, December 2016.