[5] A. Barth, C. Jackson, and J. Mitchell. Securing frame communication in browsers. In *USENIX Security*, 2009.

[6] E. Chin, A. Felt, K. Greenwood, and D. Wagner. Analyzing inter-application communication in Android. In *MobiSys*, 2011.

[7] E. Chin and D. Wagner. Bifocals: Analyzing WebView vulnerabilities in Android applications. In *WISA*, 2013.

[8] Chrome app samples. https://github.com/GoogleChrome/chrome-app-samples, 2014.

[9] Permissions in Chrome apps and extensions. https://developer.chrome.com/apps/declare_permissions, 2014.

[10] Cordova platform support. http://cordova.apache.org/docs/en/3.4.0/guide_support_index.md.html, 2014.

[11] L. Davi, A. Dmitrienko, A. Sadeghi, and M. Winandy. Privilege escalation attacks on Android. In *ISC*, 2010.

[12] D. DeFreez, B. Shastry, H. Chen, and J. Seifert. A first look at Firefox OS security. In *MoST*, 2014.

[13] M. Dietz, S. Shekhar, Y. Pisetsky, A. Shu, and D. Wallach. QUIRE: Lightweight provenance for smart phone operating systems. In *USENIX Security*, 2011.

[14] A. Felt, H. Wang, A. Moshchuk, S. Hanna, and E. Chin. Permission re-delegation: Attacks and defenses. In *USENIX Security*, 2011.

[15] Firefox OS app permissions. https://developer.mozilla.org/en-US/Apps/Build/App_permissions, 2014.

[16] Firefox OS security model. https://developer.mozilla.org/en-US/Firefox_OS/Security/Security_model, 2014.

[17] M. Finifter, J. Weinberger, and A. Barth. Preventing capability leaks in secure JavaScript subsets. In *NDSS*, 2010.

[18] B. Ford and R. Cox. Vx32: Lightweight user-level sandboxing on the x86. In *USENIX ATC*, 2008.

[19] T. Garfinkel, B. Pfaff, and M. Rosenblum. Ostia: A delegating architecture for secure system call interposition. In *NDSS*, 2004.

[20] M. Georgiev, S. Jana, and V. Shmatikov. Breaking and fixing origin-based access control in hybrid Web/mobile application frameworks. In *NDSS*, 2014.

[21] I. Goldberg, D. Wagner, R. Thomas, and E. Brewer. A secure environment for untrusted helper applications: Confining the wily hacker. In *USENIX Security*, 1996.

[22] M. Grace, W. Zhou, X. Jiang, and A. Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *WiSec*, 2012.

[23] M. Grace, Y. Zhou, Z. Wang, and X. Jiang. Systematic detection of capability leaks in stock Android smartphones. In *NDSS*, 2012.

[24] N. Hardy. The Confused Deputy: (or why capabilities might have been invented). *ACM SIGOPS Operating Systems Review*, 1988.

[25] S. Jana, D. Porter, and V. Shmatikov. TxBox: Building secure, efficient sandboxes with system transactions. In *S&P*, 2011.

[26] X. Jin, X. Hu, K. Ying, W. Du, Y. Heng, and G. Peri. Code injection attacks on HTML5-based mobile apps: Characterization, detection and mitigation. In *CCS*, 2014.

[27] X. Jin, T. Luo, D. Tsui, and W. Du. Code injection attacks on HTML5-based mobile apps. In *MoST*, 2014.

[28] K. Lin, D. Chu, J. Mickens, L. Zhuang, F. Zhao, and J. Qiu. Gibraltar: Exposing hardware devices to Web pages using AJAX. In *WebApps*, 2012.

[29] M. Louw, K. Ganesh, and V. Venkatakrishnan. AdJail: Practical enforcement of confidentiality and integrity policies on Web advertisements. In *USENIX Security*, 2010.

[30] L. Lu, Z. Li, Z. Wu, W. Lee, and G. Jiang. Chex: Statically vetting Android apps for component hijacking vulnerabilities. In *CCS*, 2012.

[31] T. Luo, H. Hao, W. Du, Y. Wang, and H. Yin. Attacks on WebView in the Android system. In *ACSAC*, 2011.

[32] S. Maffeis and A. Taly. Language-based isolation of untrusted javascript. In *CSF*, 2009.

[33] M. Miller, M. Samuel, B. Laurie, I. Awad, and M. Stay. Caja: Safe active content in sanitized JavaScript. http://google-caja.googlecode.com, 2008.

[34] WebView addJavascriptInterface remote code execution. https://labs.mwrinfosecurity.com/blog/2013/09/24/webview-addjavascriptinterface-remote-code-execution/.

[35] NSA. Security-enhanced linux. http://www.nsa.gov/research/selinux/.

[36] Ubuntu OnlineAccounts API. http://developer.ubuntu.com/api/html5/sdk-14.04/OnlineAccounts.OnlineAccounts/, 2014.

[37] J. Politz, S. Eliopoulos, A. Guha, and S. Krishnamurthi. ADsafety: Type-based verification of JavaScript sandboxing. In *USENIX Security*, 2011.

[38] N. Provos. Improving host security with system call policies. In *USENIX Security*, 2003.

[39] E. Shapira. Analyzing an Android WebView exploit. http://blogs.avg.com/mobile/analyzing-android-webview-exploit/.

[40] S. Shekhar, M. Dietz, and D. Wallach. AdSplit: Separating smartphone advertising from applications. *USENIX Security*, 2012.

[41] K. Singh. Practical context-aware permission control for hybrid mobile applications. In *RAID*, 2013.

[42] K. Singh, A. Moshchuk, H. Wang, and W. Lee. On the incoherencies in Web browser access control policies. In *S&P*, 2010.

[43] S. Son and V. Shmatikov. The postman always rings twice: Attacking and defending postMessage in HTML5 websites. In *NDSS*, 2013.

[44] R. Stevens, C. Gibler, J. Crussell, J. Erickson, and H. Chen. Investigating user privacy in Android ad libraries. In *MoST*, 2012.

[45] W. Sun, Z. Liang, V. Venkatakrishnan, and R. Sekar. One-way isolation: An effective approach for realizing safe execution environments. In *NDSS*, 2005.

[46] SUSE. AppArmor Linux application security. https://www.suse.com/support/security/apparmor/.

[47] System applications working group charter. http://www.w3.org/2012/09/sysapps-wg-charter, 2014.

[48] A. Taly, U. Erlingsson, J. Mitchell, M. Miller, and J. Nagra. Automated analysis of security-critical JavaScript APIs. In *S&P*, 2011.

[49] Ubuntu Unity Web API. http://developer.ubuntu.com/api/devel/ubuntu-13.10/javascript/web-docs/, 2014.

[50] R. Wang, L. Xing, X. Wang, and S. Chen. Unauthorized origin crossing on mobile platforms: Threats and mitigation. In *CCS*, 2013.

[51] App capability declarations (Windows Runtime apps). http://msdn.microsoft.com/en-us/library/windows/apps/hh464936.aspx, 2014.

[52] M. Zalewski. Browser security handbook. https://code.google.com/p/browsersec/wiki/Main.