2015

Security assessment of IoT devices: The case of two smart TVs

Maxim Chernyshev
Security Research Institute, Edith Cowan University

Peter Hannay
Security Research Institute, Edith Cowan University

DOI: 10.4225/75/57b3fa87fb88d
13th Australian Digital Forensics Conference, held from the 30 November – 2 December, 2015 (pp. 85-94), Edith Cowan University Joondalup Campus, Perth, Western Australia.

SECURITY ASSESSMENT OF IOT DEVICES: THE CASE OF TWO SMART TVS

Maxim Chernyshev, Peter Hannay
Security Research Institute, Edith Cowan University, Perth, Western Australia
m.chernyshev@ecu.edu.au, p.hannay@ecu.edu.au

Abstract

Being increasingly complex devices, smart TVs are becoming more capable and have the potential to receive, store, process and transmit considerable amounts of personal data. These capabilities also represent several diverse attack surfaces potentially rendering these devices highly vulnerable. The emergence and high adoption rate of smart TVs have been drawing notable interest from security researchers and industry. We utilise an attack surface area-based approach to assess the security of two modern smart TVs from different vendors and describe some of the possible multi-surface attacks that can be carried out against these devices.

Keywords
IoT security, smart TV, inter-protocol exploitation

INTRODUCTION

Integration of modern technologies and Internet connectivity has been driving the evolution of once isolated traditional television (TV) sets into connected and feature rich living room hubs. The so-called "smart" TVs incorporate various elements to offer an interactive, more engaging and personalised experience. In addition to standard viewing features, these devices generally support local media playback, web browsing, streaming of external media, support for voice commands as well as consumption of third party applications and content services. Gartner (2012) state that there will be around 200 million smart TVs sold in 2016 and at least 1 in 8 Australians now have a smart TV in their home (Roy Morgan Research, 2015). The rich feature set provided by these devices represents a diverse range of potential attack surfaces. Given two smart TVs by different manufacturers, we examine some of these surfaces and discuss potential attack scenarios.

RELATED WORK

A smart TV is a device that integrates traditional television technology with a computing platform (Sutherland, Read, & Xynos, 2014). Arguably, smart TVs can be classified as Internet of Things (IoT) devices. In our study, we employ the definition by Whitehouse (2014) who describes IoT as "a global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities…" assuming data capture, event transfer, network connectivity and interoperability as its key characteristics. As discussed later, devices examined by us were found to exhibit all of these traits.

IoT Security

The business-oriented IoT application space often dictates the need for cost-effective business models that can result in overlooking security to reduce time to market and lower the production expenditure. IoT security is an active research area with a number of unaddressed challenges. According to Suo, Wan, Zou, and Liu (2012), these challenges emerge because:

1. IoT extends the traditional internet using a variety of established and emerging protocols;
2. Nodes in IoT deployments have internet connectivity; and
3. Nodes are also potentially interconnected.

Being possibly inter-linked extensions of the Internet, IoT deployments present an expanded attack surface. This notion is made worse by the fact that large-scale deployments consist of an array of diverse nodes, which are often characterised by limited computational resources and lack of convenient physical access (Gang, Zeyong, & Jun, 2011). From the data handling perspective, privacy issues represent a significant concern, with the European Commission stating that, in the first instance, IoT deployments are unlikely to be designed to meet the relevant requirements, such as the right of deletion and the right to be forgotten (Whitehouse, 2014). Finally, from the legislation perspective, there is a need for new and improved regulation reflecting the unique challenges of IoT

through a "heterogeneous and differentiated legal framework" that is able to handle the global and ubiquitous nature of IoT (Weber, 2010).

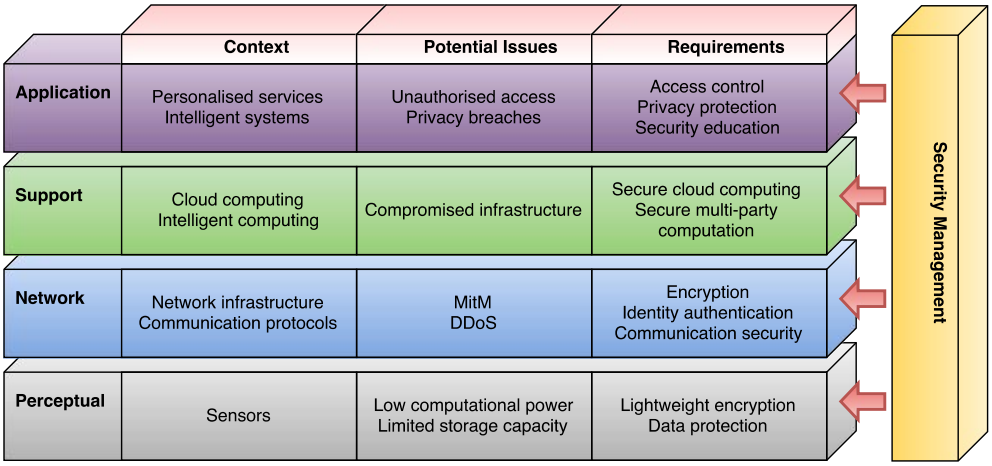

Figure 1. Layered IoT Security Architecture. Based on Suo et al. (2012, pp. 648-649).

To systemise the diversity of inherent security challenges, a number of researchers described similar IoT security architectures with one example presented in Figure 1 (Heer et al., 2011; Jing, Vasilakos, Wan, Lu, & Qiu, 2014; Suo et al., 2012). However, it may not be practical to structure applied security assessments based on a high-level architecture alone.

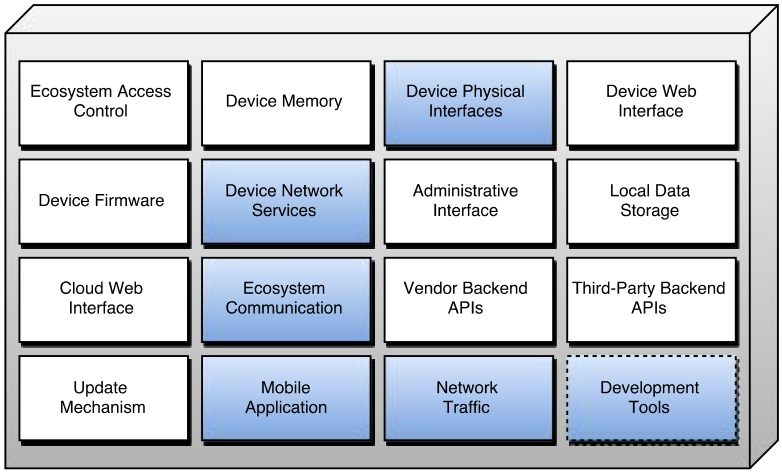

Figure 2. IoT Attack Surface Areas. Based on Miessler (2015).

To support and inform practical assessments, the community behind the Open Web Application Security Project (OWASP) introduced the "IoT Top 10" list of common security issues (OWASP, 2015). The listing has been subsequently complemented by an array of fifteen IoT attack surface areas to facilitate a universal approach, as shown in Figure 2 (Miessler, 2015). Thus, we structure our study based on a subset of these areas. We also propose an additional item called "Development Tools" to reflect the fact that examination of developer resources can provide its own benefits, as discussed later in this paper.

Smart TV Security

Grattafiori and Yavor (2013) provide a comprehensive analysis of Samsung smart TV security and demonstrate how specific components such as firmware, applications and web browser can be easily attacked in a targeted fashion due to a "systemic problem within the platform". Lee and Kim (2013) provide a vendor-agnostic overview, labelling smart TVs an ideal target for surveillance activities. However, the TV does not need necessarily to be compromised to enable surveillance, as the feature may already be present by design. For

instance, certain smart TVs have been found to be sending voice data to a third party service over an unencrypted channel for the stated purpose of voice recognition (Munro, 2015).

Oren and Keromytis (2014) describe a method of abusing the Hybrid Broadcast-Broadband Television (HbbTV) standard to inject malicious payloads into the TV content stream, which is rendered by a web browser engine. The discussed method could be used to mount a large-scale attack targeting in excess of 20,000 devices from a single location for under US$450. Unfortunately, this research was dismissed as being potentially insignificant due to stated reasons of difficulty of wide scale deployment and monetization of such an attack.

Smart TV Forensics

The feature set provided by smart TVs has the potential to facilitate misuse, making the device a possible yet neglected source of digital evidence. Guidance on forensic examination of smart TVs at the time of writing appears to be scarce and the process generally requires access to specialised expertise and hardware (Sutherland et al., 2014). Possibly the most comprehensive guide at the time of writing is provided by Boztas, Riethoven, and Roeloffs (2015). The guide suggests that smart TVs should be treated like any other embedded system and describes a number of acquisition methods. However, as the study is limited to the specific make and model, and as such additional research in this area is still needed.

Problem Statement and Significance

Smart TVs can act as intermediaries between the Internet and the local home or corporate network. A compromised device could be used to mount additional attacks against other smart appliances and traditional devices. In a surveillance context, the camera and microphone of a compromised smart TV could be used to observe residents or workers – an activity that was found to be the biggest fear of a smart TV owner (Lee & Kim, 2013).

At the time of writing, most of the research into smart TV forensics and security is based on Samsung devices, possibly due to their popularity and rootkit availability (SamyGo Forum, 2013). Yet, other vendors use different technology components meaning that specific findings may not be universally applicable. For example, modern LG devices are based on a derivative of the open source webOS operating system (LG Electronics, 2015c). Panasonic smart TVs manufactured between 2010 and 2014 are based on the proprietary Viera Connect platform, which utilises cloud-based application delivery model and requires an active Internet connection to provide most of its features (Lane, 2013). Our hypothesis is that we can use open-source security assessment tools and commodity hardware components to identify potential attack scenarios for smart TVs other than Samsung.

MATERIALS AND METHODS

The following materials were used to complete the assessment:

- Panasonic Viera THL50ET60A smart TV
- LG 55UF770T smart TV
- Apple Macbook Pro running Kali 2.0 penetration testing distribution
- Asus Nexus 7 running Android 4.4.4 with Kali NetHunter 1.2 mobile penetration testing distribution
- TP-LINK TL-WN722N USB wireless network interface card (NIC)

In order to achieve the required depth of assessment for the research at hand we focussed scope to that with the greatest potential impact. Additionally, some areas of scope were eliminated due to the non-availability of agreements required to access certain vendor and third party APIs. Subsequently, we concentrated our efforts on the following areas:

- Device physical interfaces
- Device network services
- Ecosystem communication
- Mobile application
- Network traffic
- Development tools

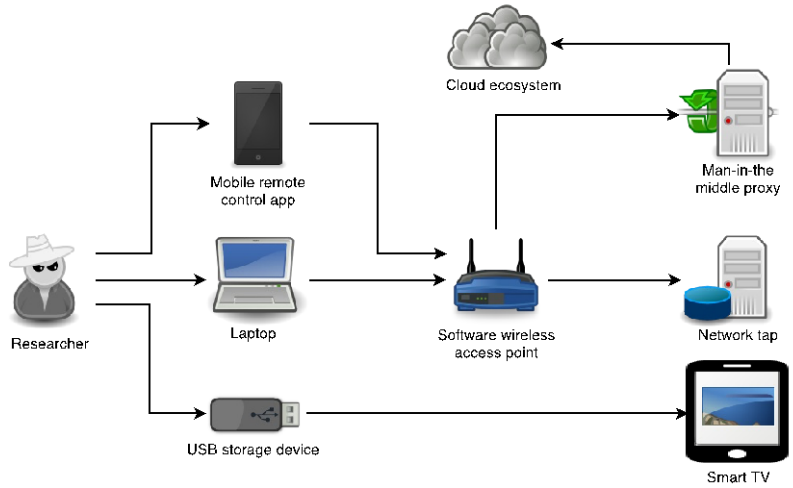

Figure 3. Assessment test bed based on the described materials.

We utilised a mix of open-source tools and commodity hardware to create the assessment test bed presented in Figure 3. We provide an outline of the examination approach for every area in Table 1. The undertaken research was conducted using a quasi-experimental methodology with empirical components. This methodology was selected, as the nature of IoT devices is that they require data from external sources that are out of the author's control. While steps were taken to mitigate the impact of these variables it is not possible to control all of them due to the connected nature of such devices.

Table 1. Assessment approach details for every examined surface area.

Surface Area	*Method*	*Tools*	*Notes*
Device physical interfaces	Deliver a Human Interface Device (HID) payload in an attempt to initiate a background terminal session with a reverse TCP shell.	HID Ducky Script Attack using Kali NetHunter 1.2 and wired USB connection to smart TV.	The examination was limited to external USB storage ports.
Device network services	Enumerate and fingerprint available network services using common network reconnaissance techniques.	Nmap (6.49BETA4) and OpenVAS (8.0).	
Ecosystem communication / Network traffic	Intercept and inspect network payloads and metadata exchanged between devices and their cloud ecosystems and other endpoints.	Mitmproxy (0.13), tcpdump (4.6.2), Wireshark (1.12.6) and Mana toolkit (20150707).	Encrypted payloads were not examined due to inability to install custom trusted certificates.
Mobile application	Intercept and inspect network payloads and metadata exchanged between the device and remote control mobile app.	tcpdump (4.6.2), Wireshark (1.12.6) and Mana toolkit (20150707).	Traffic was captured en masse for subsequent analysis.
Development tools	Examine the provided device emulator and software development kit (SDK).	LG webOS TV Emulator (2.0.0), LG webOS TV CLI	Due to lack of equivalent tools for Panasonic devices, only the LG emulator was examined.

Network Capture Collection and Analysis

Where network traffic interception and inspection are involved, the analysis was performed on the basis of captured packet files for each of the following scenarios:

- *Ecosystem communication / Network traffic* – sequential action execution (refer to the relevant Results section for action description) resulting in *lg-network.pcap* and *pana-network.pcap*.
- *Mobile application* – interaction with two remote control mobile apps for each TV (refer to the relevant Results section for app description) resulting in respective *lg-remote-*.pcap* and *pana-remote-*.pcap* files.

The overview of the captured packet traces is provided in Table 2. Subsequent analysis was performed with the aid of both graphical and command-line packet inspection tools available as part of the *Wireshark* suite of utilities. For the remainder of the areas, exploratory analysis was performed to determine feasible attack scenarios that could drive future investigations.

Table 2. Overview of captured pcap files used in assessing ecosystem communication / network traffic and mobile application surface areas.

File name	*Number of packets*	*File size (bytes)*	*Data size (bytes)*	*Capture duration (seconds)*	*Average packet size (bytes)*
lg-network.pcap	92222	89269227	87793651	992.667771	951.98
lg-remote-1.pcap	9691	4357427	4202347	151.865922	433.63
lg-remote-2.pcap	6196	2394510	2295350	206.704788	370.46
pana-network.pcap	10100	7261925	7100301	634.134583	703
pana-remote-1.pcap	5522	2762434	2674058	391.037781	484.26
pana-remote-2.pcap	6597	3192444	3086868	365.395744	467.92

RESULTS

We discuss our findings in the subsequent sections dedicated to each analysed surface area.

Device Physical Interfaces

Both devices undergoing evaluation had multiple USB ports capable of interfacing with external media storage devices and input peripherals such as keyboards. Therefore, we assumed potential susceptibility of these interfaces to Human Interface Device (HID) attacks such as those described by Crenshaw (2011). Kali NetHunter 1.2 supports Rubber Ducky syntax-based payloads that encapsulate series of keystrokes interpreted by the target system as keyboard input. While there are various ready-to-use payloads available, most of them are tailored to Windows or OS X-based systems. Knowing that our smart TVs are Linux-based, we attempted a number of payloads aimed at initiating a background terminal session with reverse TCP shell without success. We observed that both TVs reacted to these payloads in similar fashion by demonstrating rapid channel or application screen switches. A sample payload is presented in Figure 4.

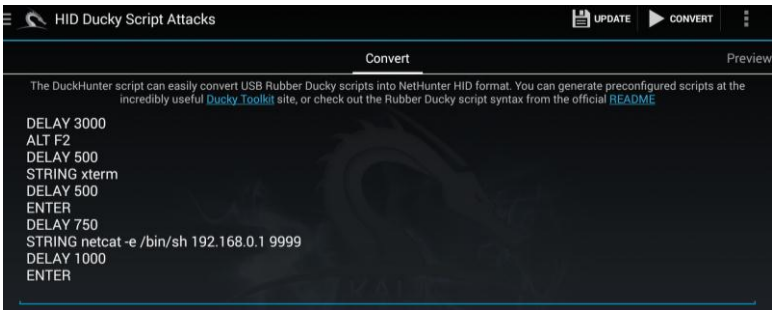

Figure 4. Kali NetHunter 1.2 HID Ducky Script Editor screen showing one of the delivered payloads.

Device Network Services

Smart TVs are discoverable on the network to enable media streaming and remote control by mobile applications. This functionality is achieved by exposing network services that could be enumerated using standard reconnaissance tools. In addition to network service discovery, we performed vulnerability scans based on identified ports, but the resulting findings were deemed insignificant. We categorised the discovered services into:

1. Universal Plug and Play (UPnP) endpoints, and
2. Ancillary services

The former enables standard services such as media rendering and remote control, whereas the latter is responsible for provisioning of specialised services, such as web page debugging. UPnP is a well-established

standard that is prone to known security issues, such as common misconfiguration or the ability to exploit specific implementations of the underlying libraries (Hemel, 2006; Moore, 2013). An example of an observed UPnP profile description is shown in Figure 5.

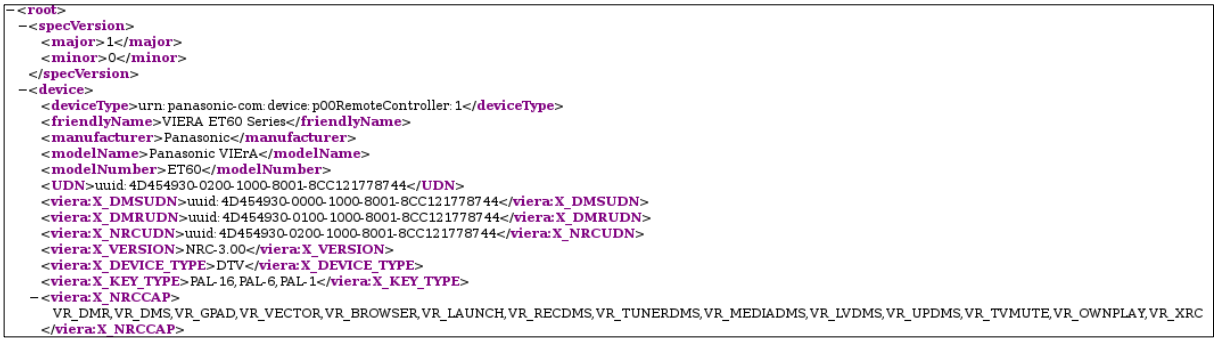

Figure 5. Subset of information available via UPnP service profile description at http://<device_ip>:55000/nrc/ddd.xml for Panasonic smart TV.

Given that UPnP should not be configured to be accessible externally, discovery and exploitation of these services requires presence on the internal network. While such actions may result in being able to send arbitrary control commands to the device, we anticipate that targets of higher significance would be pursued in that scenario. Nevertheless, information that can be obtained from UPnP endpoints can be used for device fingerprinting as various unique identifiers and device capabilities including the list of installed applications can be enumerated via endpoint interaction.

Ecosystem Communication and Network Traffic

Given that smart TVs rely on Internet access to enable most of the online features, we expected to encounter a diverse set of packet traces of potential significance during the capture experiments. To facilitate the collection, we executed a standardised set of sequential actions on each device:

- Switch through 10 channels sequentially
- Access the app store, install, use and remove an app (a weather app)
- Insert a USB storage device and play a media file
- Use voice control commands (LG smart TV only)

The high-level view of the resulting traffic flows is presented in Figure 6.

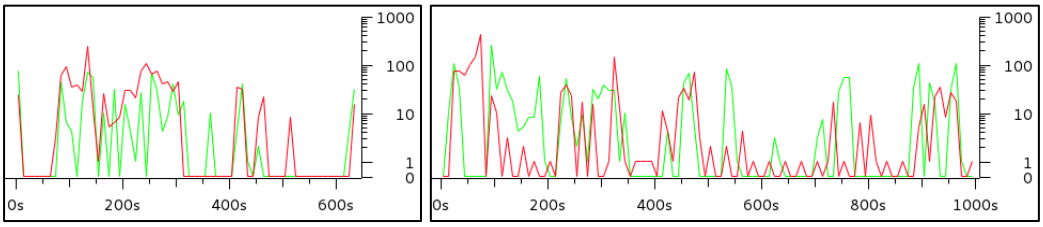

Figure 6. SSL and HTTP-based (plaintext) traffic flow statistics based on a pre-defined set of device interactions for Panasonic (left) and LG (right) smart TV. The Y scale shows the number of packets using a log scale. SSL-based interactions are shown in red.

Both devices rely on encrypted communications for the majority of interactions with their ecosystems. We observed exceptions in these cases - Internet connectivity check, latest firmware version check and licence server communication (LG only). However, respective payloads were represented by binary data decoded from base64 and would require prior knowledge about their structures for interpretation. Other plain-text traffic was observed during:

- Fetching of user interface elements (assets) and HTML-based application resources and data
- Fetching of HbbTV content for Australian digital TV channels

In the latter case, we spotted HbbTV content being served in plain text from a centralised Freeview Plus service that was launched in Australia in late 2014 (Healey, 2014). Both cases are equally significant, because they can

be exploited to perform JavaScript injection attacks into a browser-like execution environment. We describe a possible practical scenario later in this section.

Mobile Application

To diversify device control options, a user can choose to install a mobile app on their iOS or Android device. The summary of our examination of these apps for each TV is presented in Table 3.

Table 3. Findings associated with remote control mobile apps.

Description	*LG*	*Panasonic*
Mobile apps	LG TV Plus, webOS Magic Remote.	Panasonic TV Remote, Panasonic TV Remote 2.
Device discovery and communication method	UDP multicast discovery followed by HTTP-based UPnP endpoint communication using XML payloads. WebSockets-based communication for screen pointer controls.	UDP multicast discovery followed by HTTP-based UPnP endpoint communication using XML payloads.
Pairing requirement	Initial pairing triggers a 3-digit PIN displayed on the smart TV screen. The correct PIN needs to be supplied by the client to complete the pairing while the message is being displayed.	Not required.
TV control capabilities	TV remote equivalent with keyboard input.	TV remote equivalent with keyboard input.
Ancillary capabilities	Browser screen sharing, media sharing, application enumeration and launch, TV software version check and update deployment (LG TV Plus only).	Browser screen sharing, media sharing, application enumeration and launch (TV Remote 2 only).

Thus, clients can use XML payload replay attacks to execute control commands on the TV. Fortunately, the pairing process employed by the LG TV addresses this issue for untrusted clients.

Development Tools

Both devices allow installation of third party apps to personalise the TV and each vendor provides an app store for developers to distribute their apps. To aid the development process, various tools are offered to the community, including development guides, Application Programming Interface (API) references, sample code and, in the case of LG webOS, device emulators (LG Electronics, 2015d; Panasonic, 2015). We examined the available developer resources and focussed on the LG webOS 2.0.0 TV emulator as we expected it to be a reasonable representation of emulated physical devices.

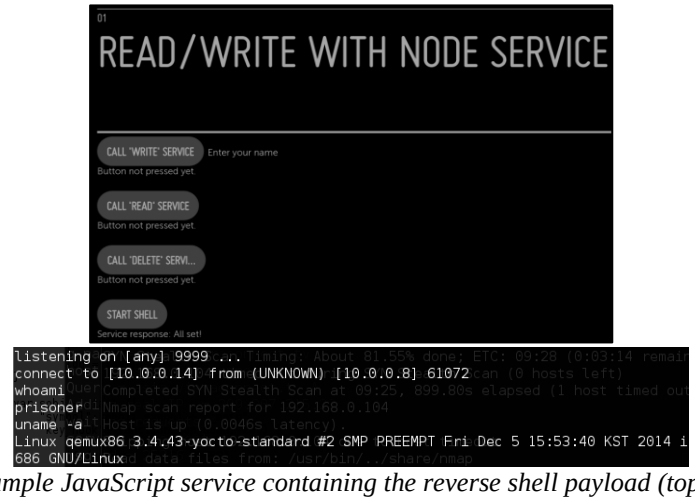

Figure 7. Modified sample JavaScript service containing the reverse shell payload (top) and output of simple commands executed in the resulting shell (bottom).

While the emulator is limited in its capabilities, it allows testing of all application types, including packaged applications, hosted applications and JavaScript services. We successfully tested the ability to initiate a reverse TCP shell to an arbitrary host and port by leveraging standard libraries available for the implementation of JavaScript services, as shown in Figure 7. While it may not be feasible to embed this functionality in a real application due to the possibility of being detected during the submission process, a malicious party could attempt to employ various evasion techniques to bypass the quality assurance controls, especially if mostly static analysis is used.

Furthermore, emulator app deployment is achieved over SSH. We were able to gain emulator shell access directly for an unprivileged account. We were also able to read the emulator root password hash, meaning that in theory one could obtain full access to the emulator given sufficient time and computational resources. We also suspect that it may be possible to achieve successful privilege escalation by tailoring, cross compiling and executing a targeted payload for the identified kernel version as an alternative to root password cracking. Even with unprivileged access, an interested party is able to analyse the local file system and service configuration files. Nevertheless, the LG device emulator could be used to gain additional insights into the possible inner workings of emulated smart TVs.

Targeting Multiple Areas Via Inter-protocol Communication

Smart TVs come with a fully-fledged web browser, however the exposure of this to the user is dependent on the device in question. Web browsers available on desktop and mobile platforms represent an attractive attack surface (Alcorn, Frichot, & Orru, 2014). Packages installed on smart TVs are not necessarily any different, because they are based on commonly available open source web rendering engines such as WebKit (Mautilus, n.d.). In the case of smart TVs, web browsers are also used to render apps, which are built using standard HTML5 technologies (LG Electronics, 2015b). An attacker could leverage existing browser exploitation tools such as the Browser Exploitation Framework (BeEF) by Alcorn (2014) and gain TV browser control in the following scenarios:

- Compromise of a centralised content delivery service (e.g. Freeview Plus) to deliver JavaScript payloads encapsulated inside HbbTV content
- JavaScript payload injection into a TV-based app that fetches external resources
- JavaScript payload injection into a hosted app

The last scenario is the least complex and can be easily demonstrated. A hosted app is an external HTML page that is launched inside a browser-equivalent application execution environment. While destination pages of hosted apps may be inspected during the store submission process, the provider can easily modify them after they have been accepted. An example of a scenario where an attacker has modified the hosted app source to inject the BeEF hook is shown in Figure 8. We were able to verify this scenario for both devices using the provided application development tools.

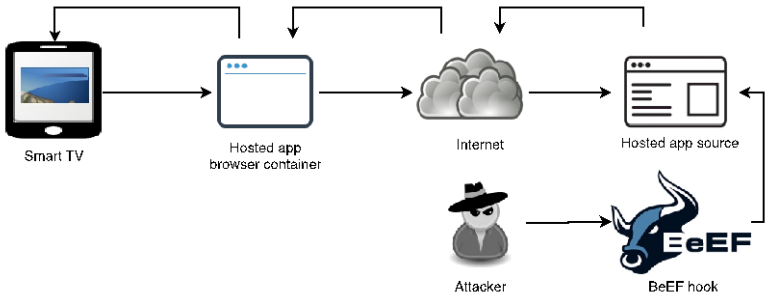

Figure 8. A possible attack scenario against an LG webOS smart TV using a hosted application injected with BeEF.

In the case of LG, we found that the execution environment for hosted apps appears identical to that used for TV-based apps, providing access to certain system services that can be used to access potentially sensitive information (LG Electronics, 2015a). Specifically, we were able to obtain the TV model name and firmware version details, the internal IP address of the device, Service Set Identifier (SSID) of the associated wireless access point and the IP address of the utilised Domain Name Server (DNS). In combination, this information could be used to mount other attacks with the potential to provide access to network on which the TV is located. Such access could be leveraged through interception of wireless traffic, firewall bypass technique's (via UPNP or NAT busting). Considering the unencrypted information leaks previously discussed this could lead to an

attacker listening in on conversations that occur within range of the device as well as discovering the viewing habits of the users. These scenarios are limited examples relating only to data that can be gathered from the Smart TV itself, access to the local network has significant further potential for breaches of privacy.

For example, based on advertised retail prices, the model number could be used to infer the socioeconomic status of the residents of the associated dwelling. Network SSID could be used to geolocate the access point using a public access point location database such as WiGLE.net (Wigle.net, 2015). Such information has the potential to facilitate the targeting of other crimes such as burglary or financial offenses. In the second case the Smart TV potentially provides both identification of high value targets and means to access the network in order to intercept banking and other financial transactions. A carefully crafted in-app popup mimicking native TV user interface could be presented to the user asking them to re-enter their wireless credentials for the discovered SSID due to a problem with network connectivity.

CONCLUSION

We followed a surface-area based approach to IoT device security assessment to examine the potential security issues of two smart TVs from different vendors and proposed the inclusion of an additional element in the framework. We inspected device physical interfaces, device network services, ecosystem communication, network traffic, mobile applications and development tools using a test bed facilitated by open-source tools and commodity hardware and identified a number of possible attack vectors. We conclude that inter-protocol communication and script injection into a browser-based execution environment powering the smart TV user experience provides an easy target that can be attacked to access device information not generally available in a standard browser environment. In the future, we plan to conduct a more thorough examination of the app execution environment and related attack scenarios.

REFERENCES

Alcorn, W. (2014). BeEF - The Browser Exploitation Framework Project. Retrieved from http://beefproject.com/

Alcorn, W., Frichot, C., & Orru, M. (2014). *The Browser Hacker's Handbook*

Crenshaw, A. (2011). *Plug and Prey: Malicious USB Devices*. Paper presented at the Shmoocon 2011, Washington, DC.

Gang, G., Zeyong, L., & Jun, J. (2011). *Internet of things security analysis.* Paper presented at the 2011 International Conference on Internet Technology and Applications (iTAP).

Gartner. (2012). Gartner Says 85 Percent of All Flat-Panel TVs Will Be Internet-Connected Smart TVs by 2016. Retrieved from http://www.gartner.com/newsroom/id/2280617

Grattafiori, A., & Yavor, J. (2013). The outer limits: Hacking the samsung Smart TV. *Black Hat Briefings*.

Healey, N. (2014). Is your TV ready for Freeview Plus? Retrieved from http://www.cnet.com/au/news/is-your-tv-ready-for-freeview-plus/

Heer, T., Garcia-Morchon, O., Hummen, R., Keoh, S. L., Kumar, S. S., & Wehrle, K. (2011). Security Challenges in the IP-based Internet of Things. *Wireless Personal Communications, 61*(3), 527-542.

Hemel, A. (2006). *Universal Plug and Play: Dead simple or simply deadly?* Paper presented at the 5th System Administration and Network Engineering Conference, Delft, The Netherlands.

Jing, Q., Vasilakos, A. V., Wan, J., Lu, J., & Qiu, D. (2014). Security of the Internet of Things: perspectives and challenges. *Wireless Networks, 20*(8), 2481-2501.

Lane, A. (2013). What's on Panasonic Smart Viera? Retrieved from https://recombu.com/digital/article/whats-on-panasonic-smart-viera_M10911.html

Lee, S., & Kim, S. (2013). Hacking, Surveilling, and Deceiving Victims on Smart TV. *Blackhat USA*.

LG Electronics. (2015a). Luna Service API. Retrieved from http://developer.lge.com/webOSTV/api/webos-service-api/

LG Electronics. (2015b). Web Engine. Retrieved from http://developer.lge.com/webOSTV/develop/web-app/webos-tv-platform/web-engine/

LG Electronics. (2015c). WebOS. Retrieved from http://www.lg.com/uk/smarttv/webos

LG Electronics. (2015d). webOS TV Developer. Retrieved from http://developer.lge.com/webOSTV/

Mautilus. (n.d.). Specifications details for Smart TVs platforms. Retrieved from http://www.mautilus.com/knowhow/specifications-details-for-smart-tvs-platforms/

Miessler, D. (2015). *IoT Attack Surface Mapping*. Paper presented at the DEFCON 23, Las Vegas, USA.

Moore, H. (2013). Security flaws in universal plug and play: Unplug. don't play.

Munro, K. (2015). Is Your Smart TV Listening to You? - Update. Retrieved from https://www.pentestpartners.com/blog/is-your-samsung-tv-listening-to-you-update/

Oren, Y., & Keromytis, A. D. (2014). *From the Aether to the Ethernet–Attacking the Internet using Broadcast Digital Television*. Paper presented at the Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA.

OWASP. (2015). OWASP Internet of Things Top Ten Project. Retrieved from https://www.owasp.org/index.php/OWASP_Internet_of_Things_Top_Ten_Project

Panasonic. (2015). Panasonic IPTV apps developers. Retrieved from https://developer.vieraconnect.com/

Roy Morgan Research. (2015). The tube meets the net: more Australians connect their televisions online. Retrieved from http://www.roymorgan.com/findings/6031-households-with-internet-smart-television-tbox-apple-and-chromecast-october-2014-201501160248

SamyGo Forum. (2013). [How to] get root access on F series. Retrieved from http://forum.samygo.tv/viewtopic.php?t=6239

Suo, H., Wan, J., Zou, C., & Liu, J. (2012, 23-25 March 2012). *Security in the Internet of Things: A Review*. Paper presented at the 2012 International Conference on Computer Science and Electronics Engineering (ICCSEE).

Sutherland, I., Read, H., & Xynos, K. (2014). Forensic analysis of smart TV: A current issue and call to arms. *Digital Investigation, 11*(3), 175-178.

Weber, R. H. (2010). Internet of Things–New security and privacy challenges. *Computer Law & Security Review, 26*(1), 23-30. Retrieved from http://www.sciencedirect.com/science/article/pii/S0267364909001939

Whitehouse, O. (2014). *Security of Things: An Implementers' Guide to Cyber-Security for Internet of Things Devices and Beyond*. Retrieved from

Wigle.net. (2015). WiGLE: Wireless Geographic Logging Engine. Retrieved from http://wigle.net