Raspberry MoCA: an Automated Penetration Platform

Andrew Hunt

Volgeneau School of Engineering, George Mason University
Fairfax, VA
ahunt5@masonlive.gmu.edu

Abstract— **Media Over Coaxial Alliance (MoCA) is a protocol encapsulating Ethernet protocols over coaxial cabling common to interior television wiring. Previous work discussed the vulnerabilities presented by common implementations of the protocol. In this paper, these vulnerabilities are realized with the development of Raspberry MoCA, an embedded device that provides a drop-in, automated exploitation kit which can be installed outside the target structure in less than five minutes, providing remote access and complete control over the connecting LAN.**

I. Introduction

Prior work on Media over Coaxial Alliance (MoCA) protocol analysis revealed a major vulnerability in common implementations [1]. The logical separation between the local area network (LAN) and wide area network (WAN) is defeated by the use of a single physical cable to transmit both signals. Most operational service providers (OSPs), such as Verizon and Cox, present the termination point of their services to the optical network terminator (ONT) attached to the exterior of the serviced building for easy maintenance. This unit converts the signal to a coaxial cable, using the MoCA protocol to encapsulate the Ethernet packets to a receiving MoCA bridge embedded in the provider's provided network router within the building.

The router binds both MoCA LAN services for video devices and MoCA WAN services from the OSP to the same cable wiring used within the structure. It also bridges the MoCA LAN to the other LAN networks – wireless and Ethernet devices. Because the WAN signal runs on the cable, it is necessary to run this coaxial cable outside to the ONT. When this occurs, the LAN signal is also exposed, as depicted in Figure 1. This presents a physical attack vector to any attacker willing to disconnect the ONT and insert a coaxial splitter to an attached MoCA-to-Ethernet bridge, and any Ethernet device.

Building on the evidence presented in the prior work, an exploitation platform was built on an embedded processing system. The developed platform includes automated enabling of remote connection from the Internet to the newly installed LAN device, as well as demonstration tools to provide to safely demonstrate the ease of which the attacker can attach, enumerate, profile, and exploit the target LAN.

II. Platform Design

Several elements were considered in the design of a platform to engage in extra-domicile attack. First, attaching to the MoCA network outside the domicile requires a power source. The attacker cannot assume that the coaxial splitter or ONT will be conveniently located to a power source. Tapping the electrical utility stack requires specialized knowledge, risk of electrocution, and most importantly time to conduct. The longer the attacker is around the property, the more likely they are to be detected. Therefore, the attacker must assume that they will provide power to the device. This can be achieved with an inexpensive universal power supply (UPS) system, widely available in many stores. An APC BackUPS 350 ES was selected for its wide availability, available management software, and low cost [2,3]. With 200 volt-amps (VA) of stored power, a typical 3.5VA embedded unit would last about 60 hours.

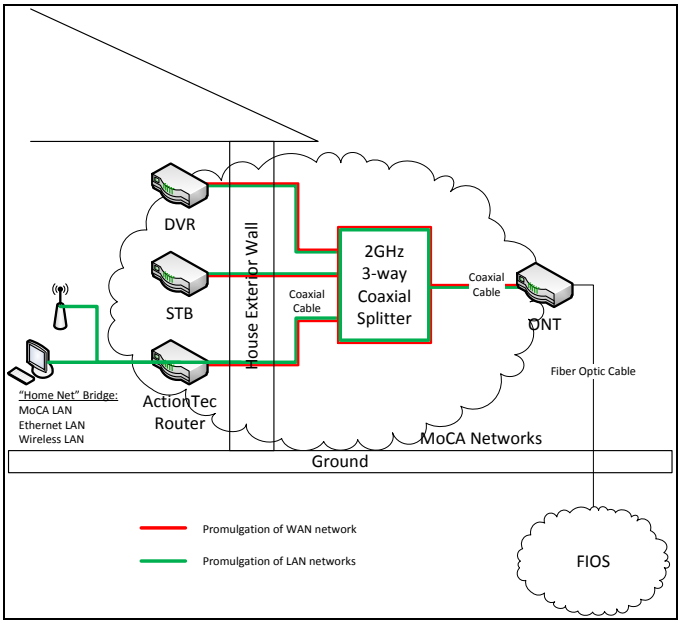

Figure 1: Promulgation of LAN and WAN over MoCA

The supply of power from a UPS is limited by the watts drawn from the battery over time against its storage capacity. This illustrates the importance of the second consideration in design, the device power footprint. The attack unit would draw minimal power to maximize the available time to attack. This would allow the attacker to plant the device, then engage in other routine activities or travel before connecting to the unit. This allows the attacker to maintain a less obvious persona and reduce the risk of immediate detection.

Low-power devices also present a smaller physical footprint, the third major consideration. Smaller devices are easier to obscure beneath or behind utility meters or decorative plants frequently employed to hide unsightly wiring installations.

Embedded devices come in a variety of sizes and capabilities. However, almost all are based on an ARM processing architecture due to its efficient assembly instruction set and low power requirements [4]. Since no off-the-shelf ARM units include a MoCA coaxial adapter with the board, an Ethernet-capable unit would need to be used with a Moca-to-Ethernet bridge. Our existing Netgear MCAB1001 and related cabling was employed for this purpose, slightly increasing the power load on the UPS.

Alleviating the need to support the MoCA chipset directly opened up the available operating systems to choose from. However, the recently released Kali Linux succeeds BackTrack as the standard high-quality, supported distribution platform with a tested suite of tools for an attacker to employ [5]. Kali also provides ARM versions of their distribution, enabling its use to bootstrap this development. Using a standard toolkit is important to an attacker as it reduces the number of observables that might reveal them should the device be discovered and analysed, the third design consideration. Upon discovery, anyone would be able to tell the unit was an attack device. However, there would be far fewer artifacts to reveal the identity of the attacker on a standard build than on a customized build unique to the choices of the builder. The attacker has some assurance that without external sources of information, the discovery of the unit would reveal nothing more than the owner was attacked.

Table 1: Raspberry MoCA Platform Components

Vendor	Model	Description
APC	BackUPS 350 ES	200 VA Universal Power Supply
Netgear	MCAB1001	MoCA Coax-to-Ethernet Adapter
RCA	DH24SPR	2-way 5-2400 MHz coaxial splitter
RCA	VH606N	2x Digital RG-6 coaxial cables
Element14	Raspberry Pi Model B R2	Embedded ARM board w/ Ethernet
Transcend	TS8GSDHC10E	Transcend 8 GB Class 10 SDHC Flash Memory Card
	Raspberry Pi board case	
Motorola	5V micro-USB phone charger	
Belkin	Cat-5e patch cable	Connect the embedded device to the MoCA adapter

The final consideration was performing the development on a standard, widely available board at minimal cost. Element14's Raspberry Pi was selected as it is an inexpensive development ARM platform that is directly supported by Kali Linux. The unit also has a wide community of support providing accessories to weatherize, power, and protect the device with only a small investment. Providing a unit that the attacker does not mind losing the cost of enables the aggressive deployment of the units, commoditizing the attacker's costs and increasing the odds of a successful engagement. Table 1 shows the final list of components used to create the Raspberry MoCA Platform.

With the materials assembled, Kali Linux was downloaded and flashed to the SD card. A USB keyboard and video cable, whether component or HDMI, and display device were required to complete the initial build. Once running, rageweb's disk expansion procedure was employed to expand the Kali installation to the entire available disk space on the SD card, a 60% gain [6]. With available space, additional tools could now be loaded. However, these needed to be selective as not to overwhelm the device's performance capabilities. First the LAN environment needed to be assessed to determine what services would be needed to achieve remote accessibility to the device, enumeration of the network, and eventual subversion of the LAN.

III. Assessing the MoCA LAN

The MoCA LAN is provided by the MoCA root node, typically located on the OSP's provided network router. A Verizon Actiontec MI424WR router was used to assess the capabilities of a typical MoCA router. This device was found to support Universal Plug-n-Play (UPnP), a technology that allows for the automated discovery of devices and services within a broadcast domain [7,8]. UPnP typically operates over UDP port 1900, providing and receiving broadcast information about and interpreting commands from other devices [9]. UPnP supports many protocols, including the Simple Service Discovery Protocol (SSDP) and Digital Network Living Alliance (DLNA), that support intra-LAN service establishment [10-16].

UPnP-Inspector and Miranda were used to assess the UPnP services active upon the network [17,18]. Both are Python-based toolkits that can actively probe or passively monitor the UPnP broadcast space to enumerate discovered machines and services offered. UPnP-Inspector also offers the ability to graphically browse and query specific environment settings provided by discovered devices [19]. While many devices provided information about the services they provided, it was determined that the focus of this effort would be on the gateway and the establishment of access, leaving the enumeration of other attached SSDP devices as an exercise for the attacker using tools like GUPnP, Rygel, Brisa, and Coherence [20-23]. This decision also reduced the footprint on the embedded device from these heavy, GUI-based programs. The gateway merely provided an interface via the Internet Gateway Device (IGD) protocol.

IGD is essentially a command wrapper to forward ports through the firewall at the behest of requesting devices [24]. However, the assumption of IGD implementations is that

every device on the LAN side of the firewall is trusted [25]. With no validation of requests, the firewall will open any arbitrary port and forward it to an internal device without authentication [26]. This makes the LAN vulnerable to devices that make illegitimate requests to open or close ports that affect other devices, client-side attacks that inject UPnP packets to the network, or nodes added to the network with malicious intent [27]. All nodes on the LAN are trusted nodes.

IV. Enabling Remote Access

With the discovery of the IGD protocol in play on the router, Kali was assessed for its ability to support the crafting of the UPnP IGD command to forward a port to the Raspberry MoCA's running SSH session. This was achievable via Miranda, but it was a multi-step manual process to achieve. Another tool, MiniUPnP was acquired and found to be the smallest, most efficient UPnP tool encountered [28]. Like most Unix tools, MiniUPnP was designed to be feature specific and fast. Having the ability to be called from the command line with the necessary arguments for port redirection made MiniUPnP an excellent tool for scripting the port forwarding procedure.

Table 2: rc.local code to establish port forwarding and reporting through a UPnP firewall

```
   #!/bin/sh -e
   #
   # rc.local

   sleep 120;
   upnpc -a `ip addr | fgrep "inet " \
   | fgrep -v "host lo" | awk '{print $2}' \
   | awk -F\/ '{print $1}'` 22 22 tcp \
   | tee /tmp/report | mailx -s `ip addr \
   | fgrep "inet " | fgrep -v "host lo" \
   | awk '{print $2}' \
   | awk -F\/ '{print $1}'`.report
surreptitiously.delicious@gmail.com

exit 0
```

The short summary of the changes returned also provided the necessary information about the external IP address of the firewall, the port forwarded, and the internal IP address forwarded to that the attacker needs to connect to the Raspberry MoCA device. This was collected and codified into the rc.local script to execute, establish forwarding, and report the pertinent information to the attacker's email address at power-on, as depicted in Table 2.

Email was chosen as the delivery method for the establishment data to employ several advantages. Free email accounts are readily available and are difficult to attribute. Email constitutes a large amount of legitimate traffic to hide within, reducing the likely visibility of a small message. The protocol also transmits asynchronously, having many available tools to ensure delivery of the important data should there be a disruption in service, to ensure delivery of this critical message. While the chosen method sends the data in the clear, other methods, such as Google or Yahoo's IMAP(S) services could be employed to provide more reliable, encrypted channels that blend into common LAN device, e.g. smartphone, communication streams [29].

V. Exploiting the MoCA Network

With remote access to the LAN, the attacker has the advantage in assessing and choosing targets. With a limited time window to operate before the platform exhausts its stored power, the attacker would likely want to establish a more permanent foothold on one of the other network devices. Ettercap is a packet-spoofing and manipulation tool that is provided with the Kali distribution [30,31]. The tool provides a modular framework from which to commit a variety of network routing and addressing attacks. Of these, Address Resolution Protocol (ARP) spoofing provides the capability to redirect the entire local broadcast domain efficiently [32-34]. Utilizing the ARP man-in-the-middle (MITM) module, Ettercap enables the attacker to direct all non-gateway device traffic to the Raspberry MoCA unit. The tool includes a native forwarding capability, which passes the packets through its filters, then out to the gateway. The same works in reverse, creating a bi-directional packet capture and manipulation capability for the attacker.

With access to LAN devices' traffic, the attacker can profile the devices specifically, gaining detailed knowledge about device versions and services that can be exploited. The attacker may employ Metasploit for a direct attack upon a discovered vulnerability [35]. They may instead choose to use the BeEF framework to manipulate web traffic bound for a client device to inject redirects or exploitation code [36]. Karmetasploit is an integration of the Metasploit Framework and Karmeta, a tool to poison software upgrade requests for many common programs, enabling the attacker to silently corrupt network nodes with no user interaction [37-39].

With the establishment of alternative backchannels, the attacker would no longer need the Raspberry MoCA for primary use. They could retrieve it with a few minutes effort to disconnect the coaxial cable and charge it for a future engagement. Should the attacker lose access to the target, they only need to redeploy the Raspberry MoCA Platform to regain control.

VI. Demonstration

The primary goal of this project was to provide a finished penetration platform that could be used as a training tool to espouse knowledge of the vulnerability of the MoCA protocol as commonly deployed. This would allow for demonstration of the aforementioned subversion techniques, potentially on live networks, in a non-impactful, public, and open way. Security researcher Joshua Wright presented his work on a similar distribution called "I Love My Neighbors," which performed traffic manipulation upon an open wireless honeypot to demonstrate to non-technical users the dangers of using unprotected open wireless networks [40]. The extent of those manipulations were simple, obvious, and many times humorous image modifications or web page redirections.

Figure 2: Manipulated image demonstrates the nascent capability of Raspberry MoCA

To enable this on the Raspberry MoCA, the squid configuration and URL rewriting scripts were acquired from Wright's provided honeypot system image. Wright's service configuration script was also modified heavily to fit the unified forwarding environment of the single-homed device. With the establishment of iptables port redirection the unit could receive and manipulate web traffic explicitly proxied to it [41]. Adding an Ettercap command to establish ARP MITM redirection for the entire LAN ensured that all hosts to be subverted, producing obvious yet innocuous picture manipulations, as seen in Figure 2.

VII. Results

Raspberry MoCA successfully redirects the entire MoCA LAN segment and its bridged wireless and Ethernet segments to the attacking device. Unwitting devices have their web traffic passed through the transparent Squid proxy and the images manipulated with the URL rewrite function [42].

The single-core ARM11 processor of the Raspberry Pi base drags noticeably when performing image manipulation as configured by the original scripts. This improved when the number of available processes was reduced from 15 Squid url_rewrites to five and 25 Apache processes to five. This reduced the memory footprint and freed up some of the interrupt contention of the processor.

Redirection of traffic via Ettercap's ARP MITM module operated flawlessly. The device was capable of managing the ARP poisons for targeted and LAN-wide subversions. Filters were successfully applied to test packet data manipulation of the word 'dog' to 'cat'. This demonstrates that the simple insertion of an iframe or javascript redirect to malicious code into a target's web stream would not provide a noticeable latency to the user.

VIII. Mitigations

As mentioned in prior work, monitoring of valid ARP announcements, MoCA and DHCP rogue nodes would provide indicators of this attack [1]. However, most home users have neither the knowledge or capability to employ these defences or monitor them effectively. Further sensing strategies that may have detected this attack include the creation of a monitoring script to dump the router IGD forwarding state and compare it for changes. These will happen from time to time, but with a log of the activity they can later be analysed or profiled to alert on suspicious mappings to known sensitive ports.

Another strategy would be to test the router's firewall implementation regardless of its reporting. An host external to the firewall would be needed to scan the external facing interface for open ports. This tool should report on unauthorized or unknown openings.

The last mitigation strategy acquiesces to the notion that the MoCA LAN is not defendable as deployed. However, its risk to the greater network can be reduced through the use of a third-party firewall. The ONT must first be configured to use an Ethernet connection to bridge the connection of the building to the OSP instead of MoCA WAN. However, with this accommodation, the ActionTec router and its untrustable MoCA LAN can be isolated to an untrusted network zone on the independent firewall. With other more trusted networks connected to other zones and traffic between the networks denied unless explicitly defined, the impact of a MoCA LAN subversion can be limited.

IX. Conclusions

The Raspberry MoCA Platform provides an effective automated penetration kit at a cost minimal enough to consider disposable. With the integration of a transparent proxy server, it also serves as an affordable education vehicle to demonstrate the threat to MoCA networks. While the manipulation of large files at layer 7 proved to be a drag on performance for this single-core, low-power ARM processor, traffic manipulation and injection were accomplished with ease.

References

[1] A. Hunt. "Media Over Coaxial Alliance (MoCA): Overview and Security Posture." Available by request.

[2] "Back-UPS ES - Product Information," *APC by Schneider Electric*. [Online]. Available: http://www.apc.com/products/family/index.cfm?id=21. [Accessed: 26-Apr-2013].

[3] A. Kropelin, "Apcupsd, a daemon for controlling APC UPSes," 13-Sep-2011. [Online]. Available: http://www.apcupsd.com/. [Accessed: 25-Apr-2013].

[4] W. Wang and T. Dey, "A Survey on ARM Cortex A Processors." [Online]. Available: http://www.cs.virginia.edu/~skadron/cs8535_s11/ARM_Cortex.pdf. [Accessed: 26-Apr-2013].

[5] Offensive Security, "Install Kali ARM on a Raspberry Pi," *Kali Linux Official Documentation*. [Online]. Available: http://docs.kali.org/armel-armhf/install-kali-linux-arm-raspberry-pi. [Accessed: 18-Apr-2013].

[6] rageweb, "Raspi-config in Kali," *Hypothetically Planned Trajectory*, 21-Mar-2013. [Online]. Available:

http://rageweb.info/2013/03/21/raspi-config-in-kali/. [Accessed: 18-Apr-2013].

[7] A. Presser, L. Farrell, D. Kemp, and W. Lupton, "UPnP Device Architecture 1.1," *UPnP Forum*, 15-Oct-2008. [Online]. Available: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf. [Accessed: 17-Feb-2013].

[8] "UPnP Forum," 2013. [Online]. Available: http://www.upnp.org/. [Accessed: 17-Feb-2013].

[9] S. Gibson, "GRC | Port Authority, for Internet Port 1900," *Gibson Research Corporation*. [Online]. Available: http://www.grc.com/port_1900.htm. [Accessed: 17-Feb-2013].

[10] "Simple Service Discovery Protocol," *Wikipedia, the free encyclopedia*. 13-Feb-2013.

[11] C. Gueguen, "Simple Service Discovery Protocol (SSDP)," *The Wireshark Wiki*, 02-Mar-2009. .

[12] Y. Goland, T. Cai, P. Leach, Y. Gu, and S. Albright, "Simple Service Discovery Protocol/1.0: Operating without an Arbiter," *Internet Engineering Task Force*, 28-Oct-1999. [Online]. Available: https://tools.ietf.org/html/draft-cai-ssdp-v1-03. [Accessed: 17-Feb-2013].

[13] R. Ahmed, L. Limam, J. Xiao, Y. Iraqi, and R. Boutaba, "Resource and Service Discovery in Large-Scale, Multi-Domain Networks," *IEEE Communications Surveys and Tutorials*, Quarter 2007.

[14] S. Cheshire, B. Aboba, and E. Guttman, "RFC 3927: Dynamic Configuration of IPv4 Link-Local Addresses," *Internet Engineering Task Force*, May-2005. [Online]. Available: http://www.ietf.org/rfc/rfc3927.txt. [Accessed: 17-Feb-2013].

[15] B. Langley, M. Paolucci, and K. Sycara, "Discovery of Infrastructure in Multi-Agent Systems." [Online]. Available: http://www.cs.cmu.edu/~softagents/papers/infrastructureDiscovery.pdf. [Accessed: 17-Feb-2013].

[16] E. Lachinov, "Digital Living Network Alliance," *Wikipedia, the free encyclopedia*. 14-Feb-2013.

[17] C. Heffner, "Miranda Readme File." [Online]. Available: file:///home/ahunt/Documents/isa564/paper/miranda-1.3/docs/readme.html. [Accessed: 18-Mar-2013].

[18] C. Heffner, "miranda-upnp - Python-based interactive UPnP client - Google Project Hosting." [Online]. Available: http://code.google.com/p/miranda-upnp/. [Accessed: 18-Mar-2013].

[19] F. Scholz, "UPnP-Inspector 0.2.2," *Python Package Index*. [Online]. Available: https://pypi.python.org/pypi/UPnP-Inspector/0.2.2. [Accessed: 18-Mar-2013].

[20] T. Potter, J.-M. Gurney, and Fluendo, "Coherence - a DLNA/UPnP Framework for the Digital Living - Trac," 02-Jan-2010. [Online]. Available: http://coherence.beebits.net/. [Accessed: 18-Mar-2013].

[21] J. Georg, J. Baayen, R. Burton, and Z. Ali, "GUPnP," *GNOME Live!* [Online]. Available: https://live.gnome.org/GUPnP/. [Accessed: 19-Feb-2013].

[22] Z. Ali, J. Georg, T. Vermier, and J. Henstridge, "Rygel," *GNOME Live!*, 11-Feb-2013. [Online]. Available: https://live.gnome.org/Rygel. [Accessed: 17-Mar-2013].

[23] "4. ssdp — SSDP Server implementation — python-brisa UPnP framework v0.10.0 documentation." [Online]. Available: http://brisa.garage.maemo.org/doc/html/upnp/ssdp.html. [Accessed: 17-Feb-2013].

[24] G. George, E. Wirt, and D. Blueman, "Linux UPnP Internet Gateway Device," 08-Feb-2007. [Online]. Available: http://linux-igd.sourceforge.net/documentation.php. [Accessed: 18-Mar-2013].

[25] vwochnik, "Administrating Your Gateway Device Via UPnP," *HowtoForge - Linux Howtos and Tutorials*, 21-Apr-2009. [Online]. Available: http://www.howtoforge.com/administrating-your-gateway-device-via-upnp. [Accessed: 17-Mar-2013].

[26] S. Gibson, "GRC | UnPlug n' Pray - Disable the Dangerous UPnP Internet Server," *Gibson Research Corporation*, 03-Mar-2008. [Online]. Available: http://www.grc.com/unpnp/unpnp.htm. [Accessed: 17-Feb-2013].

[27] H. Moore, "Security Flaws in Universal Plug and Play: Unplug, Don't Play," 29-Jan-2013. [Online]. Available: https://community.rapid7.com/servlet/JiveServlet/download/2150-1-16596/SecurityFlawsUPnP.pdf. [Accessed: 17-Feb-2013].

[28] T. Bernard, "MiniUPnP Project HomePage," Feb-2013. [Online]. Available: http://miniupnp.free.fr/. [Accessed: 25-Apr-2013].

[29] "Google Apps Platform — Google Developers." [Online]. Available: https://developers.google.com/google-apps/gmail/imap_extensions. [Accessed: 26-Apr-2013].

[30] "Ettercap (computing)," Wikipedia, the free encyclopedia. 29-Oct-2012. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Ettercap_(computing)&oldid=520362724. [Accessed: 12-Nov-2012].

[31] A. Ornaghi and M. Valleri, "Ettercap." [Online]. Available: http://ettercap.sourceforge.net/. [Accessed: 12-Nov-2012].

[32] "ARP spoofing," Wikipedia, the free encyclopedia. 12-Nov-2012. [Online]. Available: http://en.wikipedia.org/w/index.php?title=ARP_spoofing&oldid=522187503. [Accessed: 12-Nov-2012].

[33] S. Whalen, "An Introduction to Arp Spoofing," Apr-2001. [Online]. Available: http://dl.packetstormsecurity.net/papers/protocols/intro_to_arp_spoofing.pdf. [Accessed: 12-Nov-2012].

[34] A. Ornaghi and M. Valleri, "Man In The Middle Attacks Demos," in *BlackHat Conference USA*, Las Vegas, NV, 2003. [Online]. Available: http://www.blackhat.com/presentations/bh-usa-03/bh-us-03-ornaghi-valleri.pdf. [Accessed: 12-Nov-2012].

[35] "Penetration Testing Software | Metasploit." [Online]. Available: http://www.metasploit.com/. [Accessed: 19-Nov-2012].

[36] W. Alcorn, "BeEF - The Browser Exploitation Framework Project." [Online]. Available: http://beefproject.com/. [Accessed: 19-Nov-2012].

[37] M. Vallentin and Y. Ben-David, "Persistent Browser Cache Poisoning," 2010. [Online]. Available: http://www.eecs.berkeley.edu/~yahel/papers/Browser-Cache-Poisoning.Song.Spring10.attack-project.pdf. [Accessed: 19-Nov-2012].

[38] F. Amato and F. Kirschbaum, "You STILL have pending upgrades!," in *DefCon 18*, Las Vegas, NV, 2010. [Online]. Available: https://www.defcon.org/images/defcon-18/dc-18-presentations/Amato-Kirschabum/DEFCON-18-Amato-Kirschabum-Evilgrade.pdf. [Accessed: 19-Nov-2012].

[39] V. Oezer, "The Evil Karmetasploit Upgrade," in Nullcon, Zuri, India, 2009. [Online]. Available: http://nullcon.net/nullcon2010presentation/Veysel_nullcon2010_Paper.pdf. [Accessed: 19-Nov-2012].

[40] J. Wright, "Hacking Your Friends and Neighbors For Fun… (no profit, just fun) - hacking-friends," *Will Hack for Sushi*, 18-Jan-2013. [Online]. Available: http://neighbor.willhackforsushi.com/hacking-friends.pdf. [Accessed: 12-Apr-2013].

[41] "Linux iptables: Port Redirection Example." [Online]. Available: http://www.cyberciti.biz/faq/linux-port-redirection-with-iptables/. [Accessed: 26-Apr-2013].

[42] "SquidFaq/InterceptionProxy - Squid Web Proxy Wiki." [Online]. Available: http://wiki.squid-cache.org/SquidFaq/InterceptionProxy. [Accessed: 26-Apr-2013].