UNIVERSITY OF AMSTERDAM

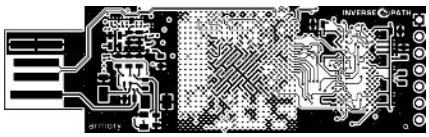
Faculty of Science
MSc. System and Network Engineering
Offensive Technologies Term Project

USB Armory as an Offensive Attack Platform

Jeroen van Kessel
jeroen.vankessel@os3.nl

Nikolaos Petros Triantafyllidis
nikolaos.triantafyllidis@os3.nl

May 31, 2015



Abstract

This research explores the feasibility of performing attacks on computer systems with the use of USB Armory, a newly introduced device which is an ARM computer in the size of a USB stick. Exploiting the USB emulation capabilities of the device we propose and test an attack scenario using a rogue DHCP server installed on the device. Based on the success of this attack we extend the scenario to DNS hijacking and traffic diversion setups with the injection of malicious static routes into the routing tables of the victim machines. This attack was successfully executed on the latest versions of Ubuntu 14.04 and Windows 8.1. The premise of the attacks as well as the scenarios themselves are explained in detail throughout the extent of this report.

Keywords: USB Armory, BadUSB, Traffic Diversion Attack, Rogue DHCP, DNS Hijacking

Image source: Inverse Path

Contents

1 Introduction

On the third of October 2014, Inverse Path an Italian IT security firm, announced 'USB Armory' an open source hardware design, implementing a flash drive sized computer [1]. This device contains an i.MX53 ARM processor, half a gigabyte of memory and a micro-SD card slot that serves as main storage.

USB Armory is designed to be used in numerous applications, mostly related to security. Some examples of these applications, according to the manufacturer, include the following [2]:

- Mass storage with advanced security features (automatic encryption, virus scanning, etc.)
- Open SSH client and agent to be used in untrusted environments.
- Routing for end-to-end VPN tunnelling
- Password management
- Bitcoin wallet
- Authentication Token
- Portable penentration testing platform

Moreover, USB Armory has built in USB emulation capabilities, meaning that the device can emulate Ethernet controllers, mass storage devices and Human Interface Devices (HIDs).

Emulating a trusted HID device such as a keyboard can impose great risks to the end-user by opening the door to the execution of malicious attack vectors. These attacks are plausible since any USB device claiming to be a HID keyboard or mouse is automatically accepted and trusted by most operating systems [3]. However, things get more interesting when Ethernet over USB emulation is enabled. In this case, a regular Linux host is connected to the victim's host machine, which exposes it to all sorts of Layer 2/3/4 attacks. That means that with minimal effort from the attacker's part a stealth IP network is connected to the victim's machine and can interact with it as a regular TCP/IP server. This project will look into the feasibility of such an attack by diverting DNS traffic and answering the DNS queries via the USB Armory device. The attack will be performed with the use of a rogue DHCP server installed on the USB Armory.

This attack has two phases it must comply to. First, the user hast to plug-in the USB Armory into their computer. Secondly, the user must be directed to a platform where they will be asked to enter their credentials, which will be later collected by the malicious party.

The next section describes the related work. Furthermore, the research question is defined and the methodology is explained based on attack trees. Moreover, the attack is explained in more detail, step by step. The last three sections summarise the conclusions of this report, initiate a brief discussion around the topic and lastly suggest future extensions to this project.

Ethical concerns

No ethical concerns were raised during this project. The attack scenarios were performed locally, on controlled university equipment. No attack scenario infected the machines with malicious software. All the attacks were terminated and the machines returned to their original state once the USB Armory devices were plugged out.

2 Related Work

Security Research Labs: Bad USB

Researchers Karsten Nohl, Sascha Krißler and Jakob Lell from Security Research Labs in Germany, have presented their work on USB devices used for malicious purposes, which they have named 'Bad USB'. Their approach mostly focuses on reprogramming the firmware of the USB sticks to hijack the normal initialisation process of the USB enabled device. They describe a process of finding leaked firmware code online, reverse engineering it by using heuristic methods in order to find hooking points. Next, they injected modified functionality assembly code in unused areas of the original firmware through a custom linker script. After having modified the firmware they propose the following attack scenarios [4]:

- *Windows infects USB stick which then takes over Linux machine*

  This scenario is about providing privilege escalation to malware that runs on Linux. In this case we need the sudo password in order to be able to infect other USB sticks that will be connected to the host in the future. Their approach says that the device restarts the screen saver with a password stealer loaded via an `LD_PRELOAD` library. Once the user enters their password to unlock the screen, the malware intercepts it and gains root access. This attack is based on keyboard emulation.

- *USB thumb drive changes DNS settings in Windows*

  In this scenario the USB device spoofs the Ethernet adapter and replies to the DHCP queries of the host with a fake DNS server but with no default gateway. This way all the Internet traffic is still routed normally through the default gateway, yet the DNS queries go to the server that was supplied by the USB stick.

- *Android diverts data traffic from Windows machine*

  This scenario is different than the ones mentioned above in the sense that it uses an Android device as its attack platform. Android phones come with Ethernet emulation installed and do not require tampering with the device firmware. Once the device is plugged into the host computer it supplies a default route via DHCP effectively intercepting all internet traffic.

Feasibility and Deployment of Bad USB

System and Network Engineering master student Stella Vouteva, explored in February 2015 the feasibility of a Bad USB attack as part of her Research Project [5]. The goal of this research was the design of an attack that can run in under 10 seconds, targeting Windows computers. The objective of the attack was to bypass Windows UAC and AV controls and gain access to the machine from a Kali Linux host.

The tools used were Kali Linux for running the exploits that would grant access to the machine and Arduino micro-controllers, disguised as HID devices, such as a keyboard or a mouse. On Windows, the malicious code triggered a GUI run-box, which then executes any commands at wish. This attack is however visible to the end user, but can be rapidly executed. While advance users might recognise this attack, most end-users will not. The research concludes that the attack is feasible for unlocked machines with certain limitations, such as keyboard settings (language, CAPS LOCK enabled, etc.), Driver Signature Verification, etc. Login screen circumvention was deemed unfeasible by this study.

USB Rubber Ducky

While S. Vouteva's research is based on the open platform of Arduino, a commercial product named USB Rubber Ducky, which is an embedded CPU disguised as a USB stick, can be used as a keystroke injection device and perform similar attacks [3]. Malicious code is written in a custom scripting language called Ducky Scripts.

All the aforementioned attacks are novel and sophisticated, however, they either require complex low level programming and in some cases even hardware soldering or are limited in their computing power and capabilities. In this report we will showcase that USB Armory can lift the complexity of performing similar attacks since it is a fully blown Linux computer, that can be attached to any host without user interaction.

3 Research Question

This project will attempt to investigate the feasibility of an attack by using the USB Armory device. Therefore the main research question is as follows:

> *How can the USB Armory be used to execute malicious code on a third-party machine by just plugging in the device?*

We will first investigate the functionality of the USB Armory. Based on our findings we will design plausible attack scenarios which will be mapped into attack trees to be presented with a certain level of abstraction.

4 Methodology

The victim machines were regular workstations running Ubuntu 14.10, and the default configuration. We also used two experimentation machines running on Lubuntu 15.04 and Windows 8.1.

This chapter describes the approach and methods used to execute the attack. Two USB Armory devices were provided by drs. ing. Jeroen van Beek RE and ing. Mark Bergman RE. Linux Debian 8.0 (Jessie) was installed on both USB Armory devices.

Rogue DHCP and DNS Hijacking

Because the USB Armory is a directly connected computer, a number of network attacks can be executed. This project makes use of a rogue DHCP server. No tampering with the original DHCP server will be required. The DHCP server will reside on the USB Armory, establishing host-to-host IP network connectivity between the USB Armory and the target host machine.

Secondly, a DNS hijacking attack will be executed. This attack is also known as a DNS diversion attack. This attack works by subverting the DNS queries to a DNS server under the control of the attacker which then resolves the domains. Again, the original DNS server will not be attacked. However, static routes will be injected into the routing table via DHCP protocol option 121. This way, a more specific route to the DNS server will be registered in the routing table, modified to point the USB Armory as a gateway.

These attacks are already known for years. Therefore, mitigation solutions have been published in order to secure an IT infrastructure against them. For example, one can protect themselves from a rogue DHCP server by configuring DHCP snooping on their switched network. However, because the USB Armory is directly connected on a previously unknown network interface, this will have no effect.

Attack Trees

The attack scenarios are modelled into attack trees shown below, according to the threat modelling methodology of Adam Shostack [6]. The attack trees describe threat scenarios to a system. The root node is the goal of the attack, while the leaf nodes are conditions to reach that goal. Sometimes, all child nodes need to be satisfied for the attack to be successful. This is denoted with an **AND** branch. In case that only one of the possible child node conditions is sufficient for the attack to succeed an **OR** branch is used.

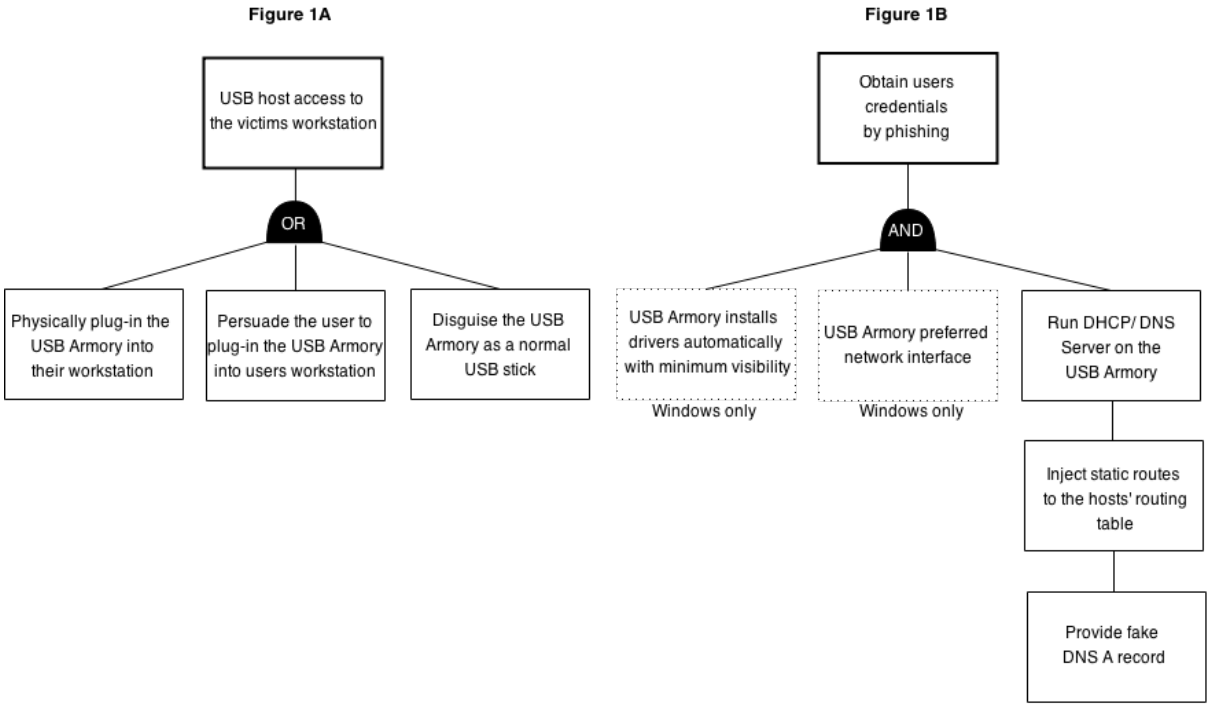


Figure 1: Attack Tree 1A: Access to the USB controller 1B: Getting users credentials

Attack tree 1A shows the conditions needed to gain access to the USB controller of the host machine. This can be achieved by simply plugging in the USB Armory, or persuading the user to plug the USB Armory into their workstation. It is also possible to trick the user by disguising the USB Armory as a normal USB stick. This method would require the emulation of mass storage on the device to be turned on so it can read and write data like a normal external storage device.

Attack tree 1B shows the conditions needed to obtain user credentials via a DNS spoofing attack. This scenario is executed by diverting DNS traffic to the USB Armory, which will resolve certain websites with a fake IP address. This way the user does not notice the change in the network configuration, and if the attack parameters are properly handled the victim machine returns in its normal state before the user is able to suspect that an attack is being performed. This attack will be explained in detail in Section 5.

5 Network diversion attack by DHCP

This attack scenario describes a DHCP network diversion attack. This attack scenario will be explained in five chronological steps. First, we explain the theory behind the drivers that emulate the Ethernet over USB network interface. Secondly, we explain how this interface will be the preferred interface of the host. Next we describe how TCP/IP connectivity is set-up between the USB Armory and the host. The fourth step is the DNS server running on the USB Armory sending DNS requests to a different DNS server. This diversion of traffic is explained more in-depth in the last step.

Figure 2 is used to explain this attack. This figure shows a host computer accessing `os3.nl` via the `eth0` interface. This interface is connected to a default gateway and a primary and a secondary DNS server. All network parameters in this setup are obtained via DHCP. However, this attack is also feasible if static parameters are used.

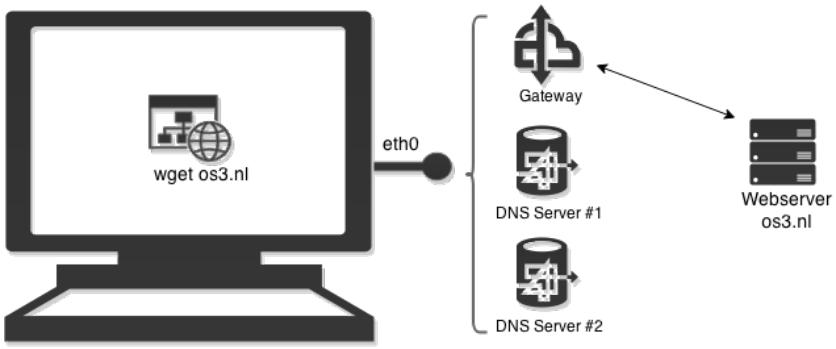


Figure 2: Environment

5.1 Drivers

Each hardware device needs a driver to communicate with the host machine. Therefore, driver communication is the first step of this attack. The USB Armory device emulates a 802.3 Ethernet over USB device network interface. In order to store data on the USB Armory, mass storage emulation can be enabled. This way, the USB Armory presents itself as a USB stick. Both Ubuntu 14.04 and Windows 8.1 react differently on the USB Armory and need different drivers. This process is described below in more detail.

5.1.1 Windows 8.1

On Windows 8.1, the USB Armory requires the Remote Network Driver Interface Specification (RNDIS) driver to support the virtual Ethernet over USB network interface. RNDIS is a Microsoft proprietary protocol which resides in all Windows versions beginning with Windows XP, up to Windows 8.1 [7]. Windows should automatically install its drivers via the USB Plug and Play (PnP) protocol. This is technically possible because the mandatory driver is by default located in the `\%SystemRoot%\System32\drivers\` directory [7]. For some reason, the RNDIS Ethernet network interface is not automatically installed. It remains to be researched how to automate this installation since the drivers are signed by Microsoft.
Figure 3 shows the RNDIS mini-port driver named `usb8023.sys`, which implements the RNDIS message set and communicates with generic bus transport drivers, which in turn communicate with the appropriate bus driver [7]. As shown in figure 3, the driver is verified and signed by Microsoft Windows.

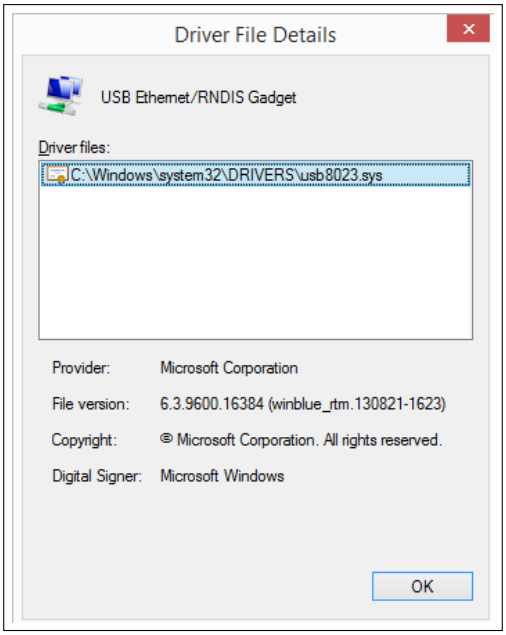


Figure 3: USB Armory Ethernet over USB network interface driver details on Windows 8.1

However, the driver is only communicating with the USB Armory Ethernet interface if only network emulation is active. In other words, when both network and mass storage emulation are active, a different multi-functional driver needs to be installed. This multi-functional device is not integrated and supported by Windows by default. How this specific driver could be installed automatically remains to be investigated in the future.

5.1.2 Ubuntu 14.04

Linux distributions use the `usbnet` driver. Linux hosts will recognise the usbnet interface automatically, since it is included in most of today's Linux kernels [8]. Both the mass storage device and the network emulation network interface work simultaneously without any interaction of the user.

Table 5.1.2 shows how Ubuntu 14.04 interprets the usbnet emulation network interface. The Vendor IDs of the USB controllers are presented to the host system depending on the type of emulation. For example, the `lsusb` utility shows how that the Ethernet network interface is shown to the host machine as a "Linux-USB Ethernet/RNDIS Gadget" while the Mass storage is shown as a generic "Linux-USB File Storage Gadget". The USB Armory is shown as a "Linux Foundation Multifunction Composite Gadget" if both the Ethernet and mass storage modules are enabled.

USB controller Vendor IDs

```
Ethernet emulation:

Bus 003 Device 018: ID 0525:a4a2 Netchip Technology,
Inc. Linux-USB Ethernet/RNDIS Gadget

Mass storage emulation:

Bus 003 Device 019: ID 0525:a4a5 Netchip Technology,
Inc. Linux-USB File Storage Gadget

Both Ethernet and Mass storage emulation:

Bus 003 Device 015: ID 1d6b:0104 Linux Foundation Multifunction Composite Gadget
```

5.2 Interface metric

USB Armory can emulate a 1 Gbit/s network interface. This section will explain how the network interface of the USB Armory will be preferred over any internal network interface(s). For Windows 8.1, the automatic network interface metric Windows feature will be abused make the emulated network interface the preferred one. On Linux we will exploit the way that the system chooses its preferred routes. Both methods are described below.

5.2.1 Windows 8.1

Windows assigns a metric value to a network interface based on the associated cost of this network interface [9]. The default configuration of any Windows operating system assigns this metric value automatically based on network interface link speed. For example, a Windows host is connected to a 100 Mbit/s network interface and to a faster 1 Gbit/s network interface, the metric of the slower 100 Mbit/s network interface will be higher, therefore less preferred, then then the 1 Gbit/s network interface. If two network interfaces have the same metric value assigned, for example two 1 Gbit/s network interfaces, then their priority is based on the network interface binding order [10]. Therefore, the USB Armory will still be preferred over any other internal network interface because it is later installed then the other network interface(s).

However, this statement is only valid if the internal interfaces are not greater than or equal to 2 Gbit/s [9]. Since 2 Gbit/s NICs are relatively expensive and not popular among consumer devices, it is very likely that the USB Armory network interface card will be the preferred network interface. In order to guarantee the preference of the USB Armory network interface over the other interfaces present, the RNDIS network interface emulation driver should be tempered to lower the metric value to '5'. If that is possible, however, remains to be investigated.

5.2.2 Ubuntu 14.04

The Linux kernel routing table has an option associated with a metric for each route entry. This metric value is usually expressed in number of hops. However, according to the main page of the `route` utility, this field is not used by modern kernels, but it is still there since it can be used by certain routing daemons [11]. Linux does not have a preferred connection, but it rather uses the most specific routes entries in the table to direct the traffic via the corresponding interface.

5.3 DHCP

The next step in the attack process is to deploy a DHCP server to lease the victim machine with an IP address on the interface that connects it with the USB Armory. This step is generic for both Windows and Linux since the DHCP protocol resides in the 7th layer of the OSI model.
Under normal circumstances, USB Armory reserves IP-address `10.0.0.1` for its own side of the connection and defines `10.0.0.2` as its default gateway. From the side of the host, however, interface `usb0` comes up but has no IP address by default. That means that the user of the USB Armory normally has to statically assign an IP in the `10.0.0.0/24` network to be able to communicate with the device. This is shown in Figure 4.

Using a DHCP server under normal usage means that the USB Armory will be able to connect and interact with any host over the network with no user interaction. That is a clear attack vector, since a DHCP server not only assigns IP addresses to its clients but it can also pass a series of other networking information such as the DNS server that the client shall use as well manipulate the routing table of the machine by pushing static routes (DHCP option 121 for Classless Static Routes) [12].

For the implementation of our Proof of Concept we used `dnsmasq`, an open source and lightweight DHCP and DNS server aimed to be used in systems with minimal resources [13].

For our needs we configured to the `dnsmasq` service to run on boot and assign an IP within the range of the `10.0.0.0/24` subnet on the `usb0` interface on the victim machine. The real attack is executed

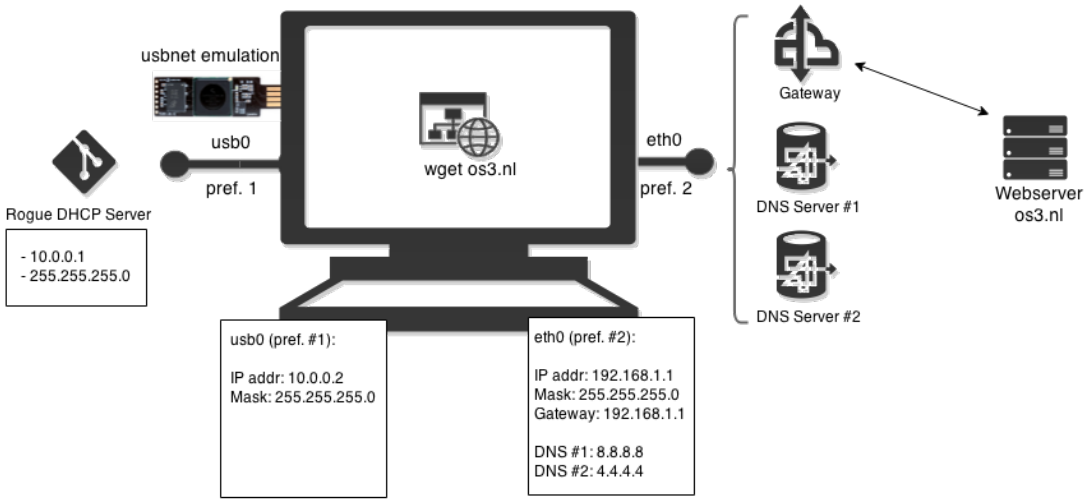


Figure 4: DHCP Server server assigning an IP-address to the USB Armory

by dictating a DNS server of our choosing, and pushing certain static routes to the host machine.

From this point on we have several paths that we can choose depending on the targeted platform.

5.4 DNS Hijacking

One can easily assign the DNS server for a system via DHCP option 6 "Domain Server". In our case we choose to use Google's Public DNS (`8.8.8.8`).

Unfortunately, according to our tests, while this option works fine in Windows hosts, it does not perform as expected in Linux systems. Windows machines will use the dictated DNS server as soon as the network configuration is in place. Linux, on the other, hand will place the newly assigned DNS server at the bottom of the `/etc/resolv.conf` file and keep using the old entries as preferred DNS server.

Luckily, this is where DHCP option 121 "Classless Static Route" comes in handy. We assume that we know the real DNS server for the targeted network. This is relatively easy since, usually, networks announce their DNS servers publicly. Then we push a static host route to the primary server via the USB Armory. On the attacking side we add an IP alias on the `usb0` interface of the USB Armory with the IP address of the targeted DNS server.

Once the USB Armory is attached to the Victim machine and the network connection between the device and the host has been configured the USB Armory will start answering all DNS queries. For that we need to add a hosts file with a few DNS entries so `dnsmasq` will be able to resolve certain addresses. All addresses that cannot be resolved by the fake DNS server will fall back to the secondary DNS server of the network. It is usual practice that networks have two DNS servers for redundancy, thus we will be impersonating only the primary server so all other operations appear normal to the user.
From that point on we can redirect web traffic to any server of our choice. It must be noted here that only DNS traffic is routed towards the USB Armory. All other traffic must continue to flow through the default interface for the victim machine, to be able to continue to find the outside world. This phase is shown in Figure 5.

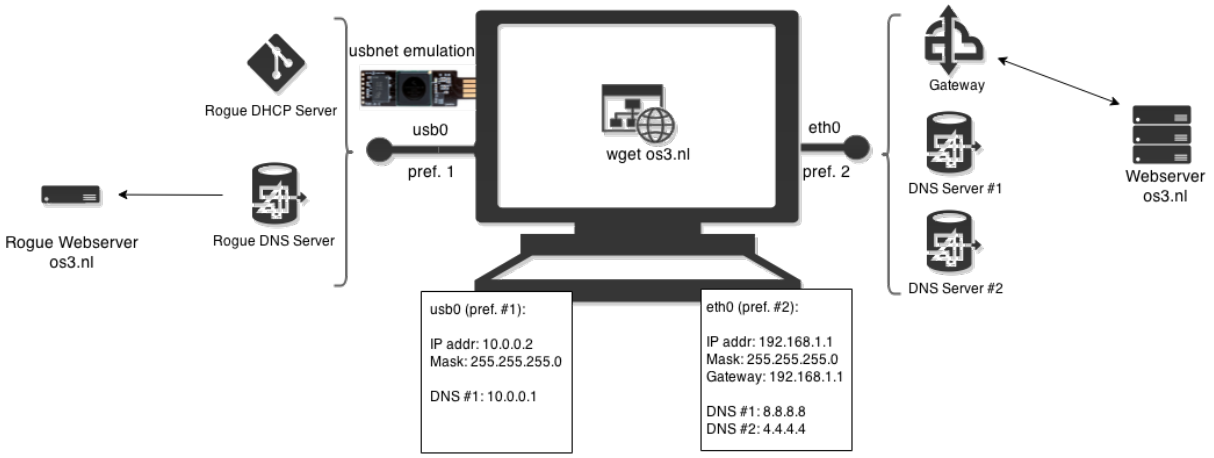


Figure 5: Divert traffic to a rogue web-server

5.5 Diverting traffic

Analogous to the aforementioned scenario, we can use DHCP option 121 to push static routes towards any host we wish. That can have two uses. First of all, we can push fake routes and bring the victim machine offline, as a form of DoS attack. Secondly, we can push the appropriate routes to basically route the whole Internet traffic through our own infrastructure.

A typical routing table in a typical Linux host contains a default gateway entry and an entry with a route towards the default gateway. The following table shows the output of such a routing table:

Kernel IP routing table before

```
\$ route -n

Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         145.100.102.65  0.0.0.0         UG    0      0        0 eth0
145.100.102.64  0.0.0.0         255.255.255.224 U     1      0        0 eth0
```

While we could try to override the default gateway with one of our choosing, this will probably collide with the route already registered, which in most cases will be preferred. Instead we can include routes that are 1 bit more specific than the default route and basically reroute the whole Internet traffic with higher chances of success. For example a route towards the `0.0.0.0` network with a subnet mask of `128.0.0.0` routes half of the IP space while a route towards the `128.0.0.0` with the same subnet mask routes the other half of the IP space. Since the most specific route is always preferred, this routes will be always chosen before the default route. The following routing table shows the routes injected by the rogue DHCP server on interface `usb0`. As we can see three new routes are added to the routing table. Besides the route towards the `10.0.0.0/23` network (i.e. the USB Armory network) the host routes `/32` towards `8.8.8.8` (Google Public DNS server) and `145.100.96.11` (the workstation configured DNS server) are dictated via DHCP option 121 with malicious purposes. The USB Armory will now answer all traffic directed to either `8.8.8.8` or `145.100.96.11` by impersonating these two hosts since it has added IP aliases for both the IP addresses.

Kernel IP routing table after

```
\$ route -n

Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         145.100.102.65  0.0.0.0         UG    0      0        0 eth0
145.100.102.64  0.0.0.0         255.255.255.224 U     1      0        0 eth0
8.8.8.8         10.0.0.1        255.255.255.255 UGH   0      0        0 usb0
10.0.0.0        0.0.0.0         255.255.255.0   U     1      0        0 usb0
145.100.96.11   10.0.0.1        255.255.255.255 UGH   0      0        0 usb0
```

6 Conclusions

This report presents a way to turn the USB Armory into an attack platform. We can safely say that a rogue DHCP attack that affects Linux hosts with minimal prerequisites from the victim's side is feasible. This attack can be used as a basis for more attacks, such as DNS diversion, Man-in-The-Middle (MiTM) attacks by rerouting all Internet traffic, or even Denial of Service (DoS) attacks by directing all the traffic through a dead-end gateway.

We have also showcased that the attack can be successful executed on a Windows host machine, provided that the RNDIS driver is installed before the attack. In a different case, the driver starts being installed the moment the device is plugged and has booted. This of course greatly reduces the stealthiness of the attack, yet it shows that the attack can be successful in Windows hosts as well.

We can conclude that the USB Armory can be used as a robust attack platform, performing the scenarios we presented or even more networking attacks. The success factor of this scenario lies on the stealthiness as well as the computing capabilities of the USB Armory. The fact that there is a whole unknown IP network installed on a USB interface that connects the victim to a fully blown disguised computer opens the door to a whole lot of possibilities.

7 Discussion

DHCP is essentially an unauthenticated protocol [19]. That raises a series of security issues. RFC 3442 [14] warns about the severity of abusing the DHCP classless static route option. As demonstrated, this DHCP option is abused by injecting static routes to misdirect network traffic. This can result in a Denial of Service attack, where traffic is routed to an invalid IP-space, or even worse, have all the traffic flow through untrusted routers [14]. There is some mitigation effort against the issues above such as RFC 3118 [15] which, however, was not largely adopted due to scalability constraints [16]. Another protection mechanism is called DHCP snooping and it works by white-listing the MAC addresses that can access the IP network. However, this assumes a switched Layer 2 network with the DHCP server on the same domain. This would make this technique useless to the setup we described before since the USB Armory creates a totally different IP network between itself and the victim machine.

Part of the attack is based on the way that Linux routes its network traffic by selecting the most specific routing entry that matches the destination IP address in its kernel routing table. In our example we added a host route towards the DNS server that was configured in the system before the attack. In case there already is a static host route entry towards the DNS server, it is very likely that this route will be preferred. Again, that might differ from system configuration to system configuration and needs to be investigated.

This attack can be prevented one level lower by creating a white-list of USB devices. However, USB devices do not always have a unique serial number and configuring this white-list is rather complicated for an average end-user [18]. Moreover, as we have seen before the USB controller IDs that each emulation mode of the USB Armory uses, are very generic and can be found in devices such as the Raspberry Pi or certain Android phones. In any case, black-listing all USB Armory devices just because they can be used as attack platforms is not the best solution since a USB Armory can be very useful if used within the intended way.

As in all Man-in-The-Middle attacks and spoofing scenarios, the attacks can be prevented by the use of secure protocols like DNSSEC and HTTPS. Indeed, since the integrity of the provided information would not be able to be verified the attack would most probably fail. However, this measure has general application and is not restricted to the attack platform we are examining in this report.

At this point we should mention that while the attack was successfully executed on the latest versions of Ubuntu 14.04 and Windows 8.1, OS X was found to be seemingly invulnerable to this sort of attack after a preliminary test we ran on a Macbook pro. That, however, remains to be further investigated before a definitive answer can be given.

8 Future Work

As mentioned before, the capabilities of the USB Armory are comparable to those of a regular computer. That means that there are a lot of possibilities to be explored in the domain of using the USB Armory as an attacking system.

First of all, we could explore ways to make the attack more sophisticated and increase its success probability. Parameters such as DHCP lease times or DNS record options could help make the attack more efficient.

Moreover, as we have seen in previous sections, the attack works on Windows machines with the restriction that the RNDIS drivers do not work by default. Moreover, if composite gadget emulation is enabled a whole new driver needs to be installed. It would be very useful to search for a way to lift this restriction and make the attack work on Windows without the need for additional actions from the side of the victim.

We have also seen that the attack was not successful on OS X hosts. Thus, part of the future work could be to explore and define the reasons why OS X is impervious to this attack scenario, and building on that try to form the attack in a way that OS X machines can be affected.

USB Armory is a complete computing platform that is capable of running full Operating Systems. This means that a large set of scanning and exploitation such as `nmap`, `yersinia`, `metasploit` tools can be installed on the USB Armory device with minimal effort. One could also create a full Kali Linux distribution for the USB Armory platform and use it to perform attacks. The very power of the device opens a lot of space for future work within the Offensive Technologies area.

References

[1] GitHub, "Inversepath UsbArmory", 2015, https://github.com/inversepath/usbArmory/wiki

[2] Inverse Path, "Open source flash-drive sized computer", 2015, http://www.inversepath.com/usbArmory.html

[3] Hakshop, "USB Rubber Ducky Deluxe", 2015, http://hakshop.myshopify.com/products/usb-rubber-ducky-deluxe?variant=353378649

[4] Impressum, "Turning USB peripherals into BadUSB", 2014, https://srlabs.de/blog/wp-content/uploads/2014/07/SRLabs-BadUSB-BlackHat-v1.pdf

[5] S. Vouteva, "Feasibility and Deployment of Bad USB", 8 Febuary 2015, https://homepages.staff.os3.nl/~delaat/rp/2014-2015/p49/report.pdf

[6] Adam Shostack, "Threat modeling: Designing for Security", 2014, http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118809998.html

[7] Microsoft, "Remote NDIS (RNDIS)", https://msdn.microsoft.com/en-us/library/windows/hardware/ff570660(v=vs.85).aspx

[8] David Brownell, "The GNU/Linux usbnet driver framework", 27 September 2005, http://www.linux-usb.org/usbnet/

[9] Microsoft, "An explanation of the Automatic Metric feature for Internet Protocol routes", https://support.microsoft.com/en-us/kb/299540

[10] Microsoft, "TCP/IP Routing Basics for Windows NT", https://support.microsoft.com/en-us/kb/140859

[11] FreeBSD, "Linux man page, route (8)", http://www.freebsd.org/cgi/man.cgi?query=route

[12] IANA, "Dynamic Host Configuration Protocol (DHCP) Parameters", 15 July 2014, http://www.iana.nl/assignments/bootp-dhcp-parameters/bootp-dhcp-parameters.xhtml

[13] Simon Kelley, "dnsmasq", http://www.thekelleys.org.uk/dnsmasq/doc.html

[14] IETF, Lemon, et. al, "RFC 3442: The Classless Static Route Option for Dynamic Host Configuration Protocol", 2002, https://tools.ietf.org/html/rfc3442

[15] IETF, Droms and Arbaugh, "Authentication for DHCP Messages", June 2001 https://tools.ietf.org/html/rfc3118

[16] IETF, T. Lemon, "Implementation of RFC 3118", http://www.ietf.org/mail-archive/web/dhcwg/current/msg00876.html

[17] IETF, Alexander and Droms, "DHCP Options and BOOTP Vendor Extensions", March 1997, https://tools.ietf.org/html/rfc2132

[18] Microsoft Technet, M. Farrugia, "How to use Windows 7 to lock down removable media and keep your computer safe", http://blogs.technet.com/b/mspfe/archive/2010/09/09/how_2d00_to_2d00_use_2d00_windows_2d00_7_2d00_to_2d00_lock_2d00_down_2d00_removable_2d00_media_2d00_and_2d00_keep_2d00_your_2d00_computer_2d00_safe.aspx

[19] IETF, Patric, "DHCP Relay Agent Information Option", January 2001 http://tools.ietf.org/html/rfc3046#section-5.0

[20] A. Barisani, "Forging the USB Armory", 26 March 2015, https://www.blackhat.com/docs/asia-15/materials/asia-15-Barisani-Forging-The-USB-Armory.pdf

Appendix

DHCP Server configuration

The following listing shows the configuration settings of `dnsmasq` DHCP server. These settings were used during the DNS spoofing attack.

dnsmasq configuration file

```
#define interface
interface=usb0

#server=8.8.8.8
no-hosts
expand-hosts
addn-hosts=/etc/dnsmasq.hosts


#DHCP
dhcp-host=usbarmory,10.0.0.1,10s
dhcp-range=10.0.0.1,10.0.0.20,255.255.255.0,10s


#static route injection
dhcp-option=3
dhcp-option=6,8.8.8.8

#add the primary DNS server of the target machine as DHCP option 121
dhcp-option=121,145.100.96.11/32,10.0.0.1, 8.8.8.8/32,10.0.0.1

#logging DNS traffic
log-queries
log-facility=/etc/dnsmasq.log
```

DNS Redirection Shell Script

The following script is used to delete the host entry after the user requests the website. Therefore, the user will be redirected to the legitimate web-site the second time the user tries to resolve that specific Web address.

script for disabling DNS redirection

```
#!/bin/bash
#Jeroen van Kessel
#Nikolaos Triantafyllidis

#variables
PATTERN="os3.nl"
LOGFILE="/etc/dnsmasq.log"
HOSTSFILE="/etc/dnsmasq.hosts"

#remove rogue website from hosts file
tail -fn0 "$LOGFILE" | \
while read line ; do echo "$line" | grep -a "$PATTERN"
        if [ $? = 0 ]; then
                sed -i '/'"$PATTERN"'/d' "$HOSTSFILE"
                echo "Pattern $PATTERN removed from the hosts file"
                exit 1
                break
        else
                echo "Pattern  $PATTERN not detected" > /dev/null 2>&1
                exit 1
                break
        fi
done
```

USB Emulation options

Shown below are the configuration options that need to be turned on in order to have USB emulation on the USB Armory device. We can choose mass storage emulation, ethernet emulation or both (composite gadget).

USB emulation configuration options

```
#/etc/modules
#mass storage emulation
g_mass_storage file=/path/to/disk.img
#ethernet emulation with static mac addresses
g_ether use_eem=0 dev_addr=aa:bb:cc:dd:ee:f1 host_addr=aa:bb:cc:dd:ee:f2
#composite gadget emulation
g_multi use_eem=0 dev_addr=aa:bb:cc:dd:ee:f1 host_addr=aa:bb:cc:dd:ee:f2 \
file=/path/to/disk.img
```