Undetectable Interception of Network Traffic on LAN Technologies

Roman Zharinov, Dmitriy Virovlyanskiy, Yuri Shvedov
Saint Petersburg State University of Aerospace Instrumentation
Saint-Petersburg, Russia
roman@vu.spb.ru, dima_virov@mail.ru, yurashvedov@yahoo.com

***Abstract*—Nowadays it's impossible to imagine a company that doesn't use Internet and Intranet (LAN) to move confidential data around. Safety measures can be both: hardware and software information security solutions. The cases of development and usage of custom, their own systems of automation of information processing (electronic document management, accounting, backup, etc.) are not rare, any more. In that case information security department assess the sustainability of computer networks or systems to attacks by the software, forgetting the data that being transmitted over the Ethernet (except, perhaps, Wi-Fi channel, but they only checking for cryptographically strong algorithm and a password). In current article, we propose a concept of the device for not-detectable interception (and modification) of the transmitted information through the Ethernet network.**

***Keywords*—interception LAN traffic, security, hardware.**

I. Introduction

In most of the companies, big part of all Ethernet traffic is, at least, confidential. For outgoing (through Internet) information there is special information security software being used (e.g. VPN), but in companies Intranet most of the data being transferred is unencrypted. It is believed that a potential attacker can't get any access to the network hardware, even by using social engineering tricks. So, often it is true: the servers, switches, routers are normally at least locked-up inside the server room and workstations usually have people working on them. On Wi-Fi hardware routers and access points usually connections that use cryptographically strong, complicated password-protected algorithms [1].

Widespread of laptops and net-books minimizes the chances of successful installation of hardware keyloggers (requires USB or PS/2 connected keyboard) and flash devices with exploits (potentially dangerous software). But, it is easier to physically steal.

But technology and progress does not stand still. And in this article, we want to introduce a concept of the device for not-detectable interception (and modification) of the transmitted data through the Ethernet network. With the device we can safely connect to active Ethernet cable, fully transparent, without being spotted. Then passively intercept all the data that being transmitted over the Ethernet and save if to internal storage (e.g. SD-memory card) or even upload it (through companies Internet connection or our devices wireless interface) to our own server or cloud file storage.

II. Tasks and purposes

A. Requirements

Preferred features and requirements:

- Invisibility of device. This concept includes the presence of such characteristics as small size, the use of low-key casing, the lack of indication, quietness (usage of passive cooling system).
- As low as possible power consumption. Power supply unit should feature the ability of supply from the rechargeable batteries, mains (AC socket) or PoE.
- Remote access, control and information management. It is necessary to have one or more wireless interface (e.g. Bluetooth, Wi-Fi or 3G) in order to securely control the device over SSH and transfer successfully captured data to our own file storage server or cloud file storage for further analysis.
- Failsafe (backup/default) mode. Ability to restore default connection wiring, in order to keep connection operational during power supply unit failures (e.g. Battery is low).
- Fast installation and connection. Usage of handy connectors (crocodile/alligator clips) and tools to speed-up Ethernet cable connection procedure.
- Usability. Ability to install and stably use *nix operating system (OS). Preferably Debian-based ARM Linux distribution. Vast majority of software tools that needed in order to successfully intercept network traffic have already been precompiled for ARM Debian Linux, also Linux OS has wide range of supported hardware.

B. Possible attacks

In order to preform attacks we have to break in to the line. For that purpose we have designed hardware switch. The main reason why we need this device is that there is no way we can strip and connect cable by hand fast enough. To prevent workstations OS “connection lost” notification, is need to reduce time when there is no physical contact, so we had to use relays. There are two stages of switching passive and active. Passive mode allows us to check accuracy of connection and executing sniffing operation. After checking accuracy and direction of traffic(by connecting to the device via SSH an checking packet sniffer's “source” and “destination” fields) we can setup software bridge and only after that we can safely cut the line (physically, with scissors).

In active mode we can cut the line (electronically with relays) and connect two separate cables to devices Ethernet ports simultaneously. Using virtual bridge interface device has an access to the network resources and Internet, scan network for vulnerable machines with popular tools for analysis. Also device can execute some common attacks. The most interesting of them are:

- Person-in-the-Middle (MitM);
- ARP poisoning;
- DoS;
- Blocking traffic.

C. Software

For device’s software we have several important requirements:

- Minimal use of system resources (not GUI, minimal requirements for CPU, etc.)
- ARM support
- Free license
- Almost free applications (app)

TABLE I. OPERATION SYSTEM’S COMPARISON

	Windows	Debian	Ubuntu
RAM	1Gb	64Mb	128Mb
HDD	16Gb	500Mb	500Mb
CPU	1GHz	300MHz	300MHz
Console mode	-	+	+
Free license	-	+	+
Support ARM processors	±	+	+
Free App	±	+	+
Fast boot	-	+	+
Remote control	+	+	+
Flexible configuration	-	+	+

Thus, the most suitable operating system for our system is Debian with support ARM architecture.

The most popular free network packet sniffers: WireShark, SniffPass, Microsoft Network Monitor (MNM), tcpdump and Capsa [2]. Their comparisons are presented at Table II.

TABLE II. SNIFFER’S COMPARISON

	WireShark	SniffPass	MNM	tcpdump	Capsa
GUI	+	+	+	-	+
Console mode	+	-	-	+	-
Supports ARM	+	N/A	-	+	N/A
Used memory	209 Mb	80Mb	60Mb	1 Kb	40Mb
Plugins	+	-	-	-	-
Language	C, Lua, Pyton	C	C	C	N/A
OS	Windows, Linux	Windows	Windows	Linux	Windows

For our system we have chosen tcpdump, because it already has working ARM-architecture port, uses low amount of RAM and has flexible console mode (which is very good for SSH remote control).

III. EXISTING SOLUTIONS

We have analyzed existing hardware and software solutions for interception and modifications information over Ethernet channel. Most of the device have no official name, so in our paper, we have given them the name of «Device N».

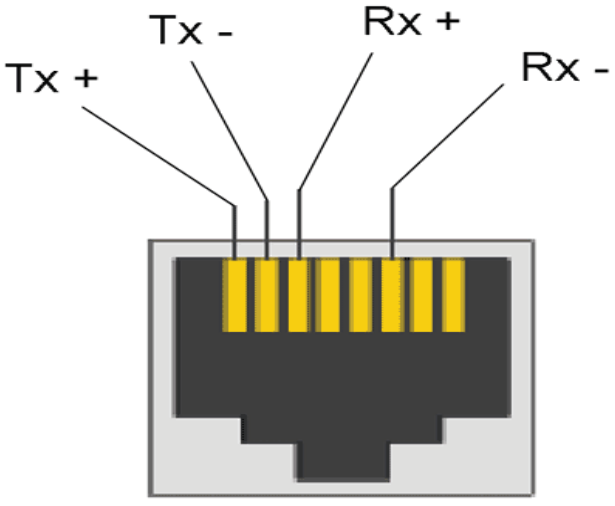

Fig.1 Signals transfer in 10/100Base-T

Device 1 [3]. Positioned as a standalone sniffer with ability to save dump to disk and controlled by Wi-Fi.

To sniff (see Fig. 1) the information need to connect destination pair (Rx for retrieve information and Tx for send information) to sniffer's host. As the device is used Wi-Fi router with the ability to change the firmware and slot for USB devices. Firmware may be DD-WRT [4] or OpenWRT [5]. Use one of these firmwares necessary to install Linux software for intercepting or analyzing network packets. USB flash is needed to store the intercepted information.

Device 2. "Throwing Star LAN Tap" [6]. The board looks like a star. It has four Ethernet ports and two network interfaces. Connect physical integrated circuits together through a Field-Programmable Gate Array is presented on Fig. 3.

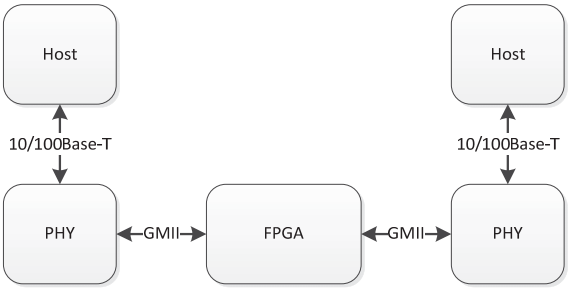

Fig. 2 Specialized Ethernet switch platform

Device 3. "Pwn Plug Elite" [7]. One of the first commercial products for penetration testing. It runs the ARM build of Ubuntu Linux and includes almost open source security tools for network analysis such as nmap, Aircrack-ng, Ettercap, etc. This device also has a few wireless interfaces, such as Wifi, GSM, USB, and Bluetooth for managing. Unfortunately, it require AC socket.

Fig. 3 Algorithm of work Pwn Plug

At Fig. 3 is present main algorithm of work [8]:

- Device must be placed in-line between an Ethernet (client PC) and a wall jack or switch.
- Using a modified layer 2 bridging module, the Pwn Plug transparently passes authentication packets between the client PC and the switch.
- The first outbound port 80 packets to leave the client PC provides the Pwn Plug with the PC's MAC/IP address and default gateway.
- To avoid tripping the switch's port security, the Pwn Plug then establishes a reverse SSH connection using the MAC and IP address of the already authenticated client PC.
- Once connected to the plug's SSH console, you will have access to any internal subnets accessible by the client PC.

The final comparison is shown in Table III.

TABLE III DEVICES'S COMPARISON

	Device 1	Device 2	Device 3
Transparency	-	-	-
Availability MITM	N/A	-	N/A
OS	dd-wrt	-	Debian
Power supply	+	-	±
Remote control (SSH)	+	-	+
Additional tools	±	-	+
RAM usage	8 mb	-	N/A

In addition, there are routers which can analyze the traffic. In this article, we will not consider them. Based on the comparison of devices, none fully meets the requirements and objectives imposed on the developed system.

IV. DEVELOPING SYSTEM

A. Overview:

The developed device is a single-board computer with an extended form as a switch. The main objective of the switch is "transparent" interception of a line. If necessary, is possible to modify network traffic.

Unlike existing solutions, one of the main problems for the system - is the autonomy.

There are 3 main ways to power our device:

- AC mains socket – simple, but not always available, unlimited uptime.
- Battery power – complicated, requires DC-DC step-up converter (e.g. 3,7V to 5V) and a battery with charger, hours of uptime (can be used as an addition to other options as UPS).
- Power over Ethernet (PoE) [9] – a simple solution to share both data and power over a network. Various standards – complicated, requires wide input voltage range DC-DC step-down converter and voltmeter (tester), unlimited uptime. But there

is one problem – PoE reduces the actual data speeds of the network (except in some cases with the high end switches) down to 10/100.

Since our device is only a prototype, right now we have it powered by lab power supply.

We plan to use multiple wireless interfaces, such as GSM, Wi-Fi or Bluetooth. Wireless interfaces allow remote management of the device using SSH. Presents of GSM module allows to install additional software and manage the device even in the absence of the Internet.

The system is developed in such a way as to appear to the ordinary administrator is transparent and difficult to detect. The term "transparency", we mean impossibility to detect the device using standard operating system tools such as ping or traceroute / tracert.

The default mode is shown on the circuit diagram showed on Fig. 4.

The relays are simply duplicating the lines, so we can safely connect, check and cut the line. Once done, we can flick the switch and engage active mode. But it isn't practical doing it on the spot, so we are planning to develop simple AVR (atmega8)-based upgrade for this system. Atmega's job here is to listen it's UART for a commands from device's main (ARM) board, and activate/deactivate relays, as soon as it receives one.

Another advantage of this circuit is - if power supply unit goes down, the relays will automatically return to backup (default) mode. Crocodile clips and scissors are not an optimal and fast kind of connector; also it's kind of conspicuous. So we are thinking about designing special fast cable-stripper-connector combo device.

Current setup:

- For demo we have three stand-along workstations with different operating systems installed. As “device” OS we using special Linux distribution - backtrack 5.
- On “device” we install two Ethernet cards for hubs imitation.

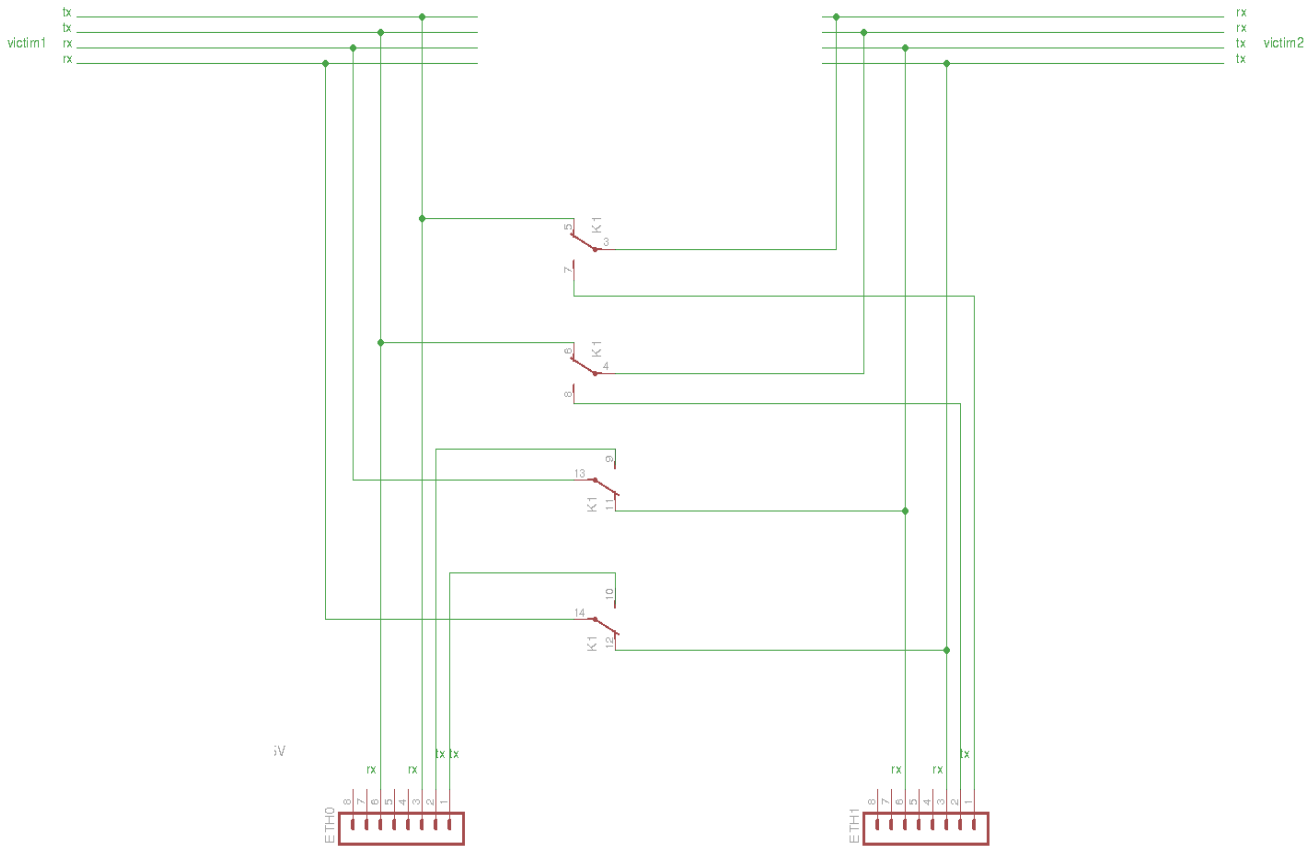

Fig. 4. Default mode

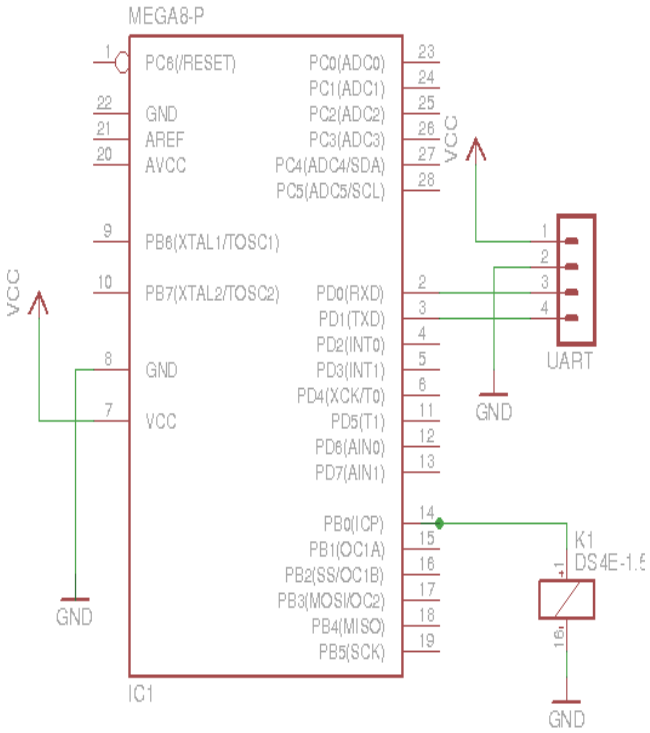

Fig. 5. AVR circuit

- Ethernet cables (about 3m of UTP-cable) with the RJ-45 connectors: six RJ45 male connectors and four RJ45 female connectors.
- Also we have four flash drivers, which allow us to save and restore the images of the systems (if it became corrupt during a test).
- One prototype of hardware switch.
- 5v DC power supply.
- Linux software network bridge.

So, at this point we have done:

- Hardware switch prototype (relay based, manually controlled). Based on AVR atmega8 - general purpose 8-bit micro-controller.
- Linux bridge interface setup (works fine for bidirectional transparent sniffing and Internet/network resources access). Network Bridge in our device consists of two network interfaces combined with each other using the utility brctl. Thus, we get a network bridge, which consists of three interfaces: two physical (eth0, eth1) and one virtual (br0).

B. Future research and development:

- Try out ARP-proxy. The main application is the ability to simulate the presence of a node from one network segment to another. This kind of technology can be used in our project, in order to hide our device, without violating the integrity and structure of the network.
- Atmega8-based automatic hardware switch, at some point we are going to develop custom PCB design.
- Active attacks, such as spoofing, MitM and others.
- Porting to ARM architecture.
- Fast cable-stripper-connector development.
- ARM board Li-ion/Li-pol power supply unit (approx. 6 hours of active uptime), also mains
- AC power supply (where available), unlimited uptime
- Try out a solid state relays. For faster switching, also they are completely silent.
- Self-destruction function, automatically preform storage memory erase, or even burn-out memory chip or SD-card.
- Camouflaging case.
- Check existing sniffer (such as sniffdet - Remote Sniffer Detection Tool/Library [10]) detectors on operation verification.

V. Conclusion

Loads of people and companies trust wired connections, way more than wireless. You should never underestimate a guy, dressed like electrician, with a strange box of wires and tools. To prevent data leakages, we must use such technologies as SSL, SSH, VPN, and other for privacy. Even with developed prototype hardware, somebody are able to remotely sniff confidential information, scan network for vulnerable machines with popular tools for analysis, do some basic DoS attacks and just use unsecured (not protected by password) network resources (SAMBA, FTP, proxy) and, of course Internet access (upload successfully captured data).

References

[1] T. Cross, “Exploiting Lawful Intercept to Wiretap the Internet”, unpublished.
[2] A. Barth et al, “Secure Content Sniffing for Web Browsers, or How to Stop Papers from Reviewing Themselves”, In Proc. of the 30th IEEE Symposium on Security and Privacy (Oakland 2009)
[3] Habrahabr official website, Sniffer vitoy pari iz Wi-Fi routera, Web: http://habrahabr.ru/post/90678/
[4] DD-WRT official website, Free Linux-based firmware, Web: http://www.dd-wrt.com/site/index
[5] OpenWrt official website, a Linux distribution for embedded devices., Web: http://openwrt.org/
[6] hackaday.com, Sniff Ethernet with a throwing star, Web: http://www.hackaday.com/2011/02/18/sniff-ethernet-with-a-throwing-star

[7] Pwnie Express official website, Pwn Plug Elite, Web: http://www.pwnieexpress.com/collections/premium-pentesting-products/products/pwnplug-elite
[8] Pwnie Express, Pwn Plug Release 1.1. User Manual, Web: http://cdn.shopify.com/s/files/1/0159/6468/files/PwnPlugRelease 1.1UserManual.pdf?2
[9] IEEE , 802.3at Amendment 3: Data Terminal Equipment (DTE) Power via the Media Dependent Interface (MDI) Enhancements, September 11, 2009
[10] Remote Sniffer Detection Tool/Library official website, Web:http://sniffdet.sourceforge.net/