Owning your own your stuff

Jailbreak

Who's device is it? Just because you bought it doesn't mean you can look inside. But the Internet has plenty of resources for jailbreakers. By Kurt Seifried

First, a personal admission: Embedded Linux generally worries me. Most embedded Linux systems are not very security conscious. Most device makers care more about making stuff work than making stuff secure. To make matters worse, vendors have a tendency to abandon their products, leaving you without security updates, and this is especially true for consumer-level devices.

Why It Matters

In the past, embedded devices weren't really a security problem because they were attached to networks. The only way to fiddle with one or attack it was to get physically close to it, take it apart, and have some fun with it. But now everything is attached to networks – your printer, your scanner, your thermostat – pretty much anything with a processor in it, which is almost everything these days. An attacker who gets access to a device will be able to launch man-in-the-middle attacks against any client systems on the same network.

An Example of Embedded Security

One example rises above all others to demonstrate why embedded devices are such a pain: default passwords. The Default Password list [1], last updated in late 2010, has almost 1,300 default passwords, most of them in hardware devices (Figure 1). Companies including 3Com, HP, IBM, and Cisco are all well represented on the list. Other security issues pop up all the time. For example, one line of network-capable scanners apparently has a flaw that lets them be activated remotely, allowing a remote intruder to see what is being scanned.

So what to do? I don't want to buy a new wireless router or cell phone as soon as the vendor stops supporting it (i.e., every six to 12 months). And, what if I want to use a device I purchased (like a PlayStation) for something other than its intended purpose (like building a supercomputer)? I'm going to load custom firmware on it, of course.

Loading Custom Firmware

Many devices support updates, which means you can install custom firmware on the device. For these devices, an open source firmware alternative often means better support, longer support, and more features. A perfect example is consumer-level wireless routers, most of which are intentionally crippled but which can be turned into VPN servers, web servers, even SIP servers with the simple loading of custom firmware. (I covered this topic in a previous column [2].) In fewer than 15 minutes, you can turn an old wireless router into something that will serve you well for a few more years.

But, not all vendors are so progressive as to let you load custom firmware on a device that you own. Cell phone vendors are perhaps the worst offenders. They don't want you anywhere near the firmware. Other vendors, like Sony, are also convinced you shouldn't be allowed to do what you want with your hardware. The PlayStation initially supported Linux when it was released back in 2000 and continued to support it officially for almost a decade. However, in 2010, Sony decided to discontinue Linux support on the PlayStation and even did their best to prevent it from working at all. So, what do you do if you want to build a supercomputer out of PlayStations, as the US Air Force did [3]?

Jailbreaking

Device vendors are increasingly locking up their devices – in effect, placing the user in a jail – with heavy restrictions on what they can and cannot do with the devices they own. "Jailbreaking" is the art of liberating a device from vendor restrictions imposed on the software. An important note: The legality of

KURT SEIFRIED

Kurt Seifried is an Information Security Consultant specializing in Linux and networks since 1996. He often wonders how it is that technology works on a large scale but often fails on a small scale.

jailbreaking varies enormously depending on where you live and what kind of device you have. For example, the US Copyright Office decided that jailbreaking your cell phone is not illegal. However, also in the US (as far as I know), it *is* still illegal to jailbreak a console system (the argument being that jailbreaking a console system encourages software copyright violations).

Assuming that jailbreaking is legal in your jurisdiction, the first step is to find (one hopes) a software-based method for breaking in. The good news is that vendors are mostly terrible at security. Software-based jailbreaks often use problems like buffer overflows in the loading of saved games, weak encryption schemes to sign updates, and PDF rendering flaws. Some of the jailbreaking software groups produce very professional and easy-to-use software that allows you to take control of your device (especially true for iPhones, and increasingly for Android-based phones).

But, what if the vendor has actually locked down the device to prevent jailbreaking? What are your options?

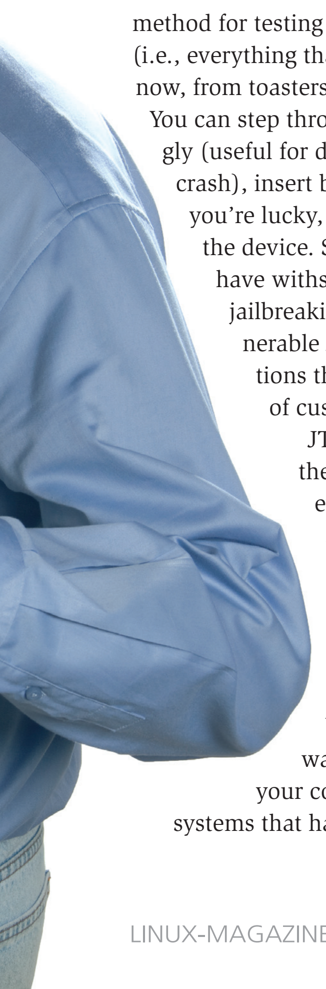
JTAG to the Rescue

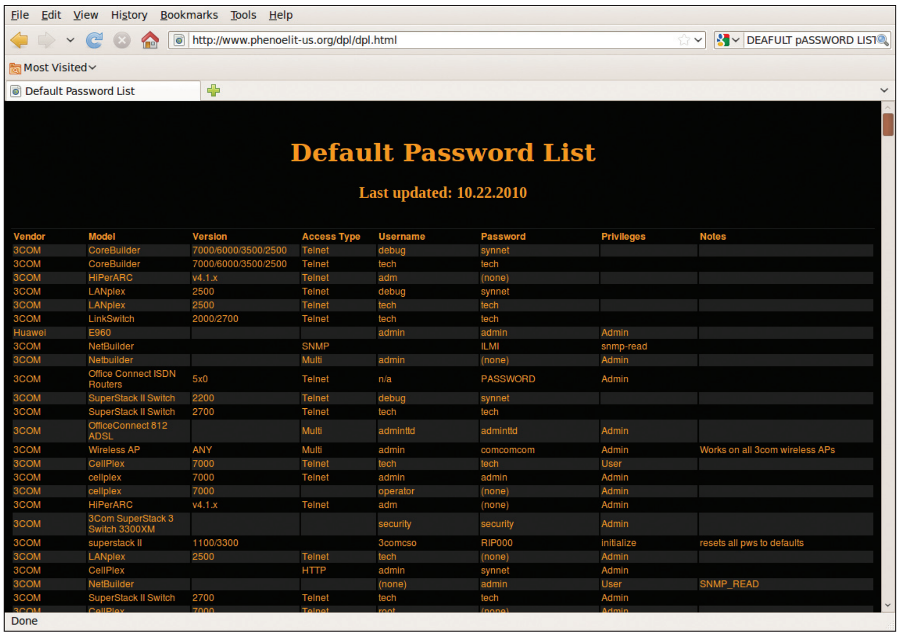
JTAG (Joint Test Action Group) is a direct result of the proliferation of embedded devices. JTAG basically specifies a method for testing printed circuit boards (i.e., everything that uses electricity now, from toasters to routers).

You can step through instructions singly (useful for debugging a persistent crash), insert breakpoints, and, if you're lucky, load firmware onto the device. Several devices that have withstood software-based jailbreaking methods have vulnerable JTAG implementations that allow the loading of custom firmware.

JTAG hacking is not for the faint of heart, however. You will need some hardware skills. The days of having to build your own JTAG-reading hardware are over, though; you can now buy USB-based hardware devices that will let your computer talk to many systems that have JTAG ports. The best news for JTAG is that, if you're lucky, you will not only be able to load custom hardware, but you might be able to gain bus master access, thus allowing you to modify system memory directly while the device is running (meaning you can control what the device does and modify the security systems it uses to keep people from loading custom firmware).

For general information on JTAG hacking, check out the "Hack a Day" blog [4]. For information on finding the JTAG pinout for a specific device, try the JTAG Finder [5] website, and for a good example, see the Qi Hardware site [6].




File Edit View History Bookmarks Tools Help
http://www.phenoelit-us.org/dpl/dpl.html
DEAFULT pASSWORD LIST
Most Visited
Default Password List
Done

Default Password List

Last updated: 10.22.2010

Vendor	Model	Version	Access Type	Username	Password	Privileges	Notes
3COM	CoreBuilder	7000/6000/3500/2500	Telnet	debug	synnet		
3COM	CoreBuilder	7000/6000/3500/2500	Telnet	tech	tech		
3COM	HiPerARC	v4.1.x	Telnet	adm	(none)		
3COM	LANplex	2500	Telnet	debug	synnet		
3COM	LANplex	2500	Telnet	tech	tech		
3COM	LinkSwitch	2000/2700	Telnet	tech	tech		
Huawei	E960			admin	admin	Admin	
3COM	NetBuilder		SNMP		ILMI	snmp-read	
3COM	Netbuilder		Multi	admin	(none)	Admin	
3COM	Office Connect ISDN Routers	5x0	Telnet	n/a	PASSWORD	Admin	
3COM	SuperStack II Switch	2200	Telnet	debug	synnet		
3COM	SuperStack II Switch	2700	Telnet	tech	tech		
3COM	OfficeConnect 812 ADSL		Multi	adminttd	adminttd	Admin	
3COM	Wireless AP	ANY	Multi	admin	comcomcom	Admin	Works on all 3com wireless APs
3COM	CellPlex	7000	Telnet	tech	tech	User	
3COM	cellplex	7000	Telnet	admin	admin	Admin	
3COM	cellplex	7000		operator	(none)	Admin	
3COM	HiPerARC	v4.1.x	Telnet	adm	(none)	Admin	
3COM	3Com SuperStack 3 Switch 3300XM			security	security	Admin	
3COM	superstack II	1100/3300		3comcso	RIP000	initialize	resets all pws to defaults
3COM	LANplex	2500	Telnet	tech	(none)	Admin	
3COM	CellPlex		HTTP	admin	synnet	Admin	
3COM	NetBuilder			(none)	admin	User	SNMP_READ
3COM	SuperStack II Switch	2700	Telnet	tech	tech	Admin	
3COM	CellPlex	7000	Telnet	root	(none)	Admin	

Figure 1: The Default Password list shows the default passwords for more than 1,300 devices.

A NOTE ON THE GPLV3

The problem of getting access is exactly why the GPLv3 matters so much. More and more embedded devices are running Linux, which is great because that means we get the source code to the underlying system and can customize it as much as we want to, right? Wrong. Many hardware vendors are doing an end run around the GPL by using Digital Rights Management (DRM) to prevent users from loading the software on their devices.

To make matters worse, the US Digital Millennium Copyright Act (DMCA) makes bypassing these DRM measures illegal, and convictions carry fines and jail time as punishment.

GPLv3 in Section 3 states that you can write DRM software that is GPLv3 licensed, but the developer must waive all rights to forbid the circumvention of the DRM (i.e., the DMCA can't be enforced). This sounds harsh until you remember that virtually everything protected by DRM is also protected by copyright or patents; DRM is just an attempt to use a technological enforcement for a social problem.

Conclusion

I believe in the doctrine of first sale; that is, once you buy something, you own it. If you want to use your PlayStation 3 as a door stop, turn it into Linux computer, or simply paint it pink and call it modern art, I believe you should be allowed to do so. So, when buying your next device, make sure you get something you can actually own and control. ■■■

INFO

[1] Default Password list: *http://www.phenoelit-us.org/dpl/dpl.html*

[2] "Linux WAP" by Kurt Seifried, *Linux Magazine*, October 2010, p48

[3] Air Force Playstation Supercomputer: *http://www.informationweek.com/news/government/enterprise-architecture/221900487*

[4] HACK A DAY: *http://hackaday.com/tag/jtag/*

[5] JTAG Finder: *http://www.c3a.de/wiki/index.php/JTAG_Finder*

[6] Qi Hardware: *http://downloads.qi-hardware.com/people/werner/ubb/vga/web/*