DRAFT

Final paper and slide release during DEF CON 23 at mcgrewsecurity.com and available on defcon.org shortly afterwards. Trust me, you'll want the final copy.

I Hunt Penetration Testers: More Weaknesses in Tools and Procedures

Wesley McGrew, Ph.D.

wesley@mcgrewsecurity.com

ABSTRACT

When we lack the capability to understand our tools, we operate at the mercy of those that do. Penetration testers make excellent targets for bad actors, as the average tester's awareness and understanding of the potential risks and vulnerabilities in their tools and processes is low, and the value of the information they gather and gain access to among their client base is very high. As demonstrated by Wesley's DEF CON 21 talk on vulnerabilities in penetration testing devices, and last year's compromise of WiFi Pineapple devices, the tools of offensive security professionals often represent a soft target. In this talk, operational security issues facing penetration testers will be discussed, including communication and data security (not just "bugs"), which impact both testers and clients. A classification system for illustrating the risks of various tools is presented, and vulnerabilities in specific hardware and software use cases are presented. Recommendations are made for improving penetration testing practices and training. This talk is intended to be valuable to penetration testers wanting to protect themselves and their clients, and for those who are interesting in profiling weaknesses of opposing forces that may use similar tools and techniques.

INTRODUCTION

Motivation

It is this author's viewpoint that penetration testers, or "attackers" in general, are simultaneously:

- Very attractive targets, for the information and access that they naturally carry and are associated with
- Highly vulnerable, due to the usage of tools and procedures that are not themselves resistant to attack

It may seem counter-intuitive, but a professional engaged in offense is not necessarily an expert on their own defense, and may lack the knowledge or experience needed to identify their own risks and take measures to prevent compromise. By describing the operational security concerns specific to penetration testers, identifying vulnerabilities in tools and procedures, and classifying tools by the degree of care that must be taken in using them, it should be possible to raise awareness among offensive security professionals. Ultimately, tests may become more secure, raising the security of penetration testers and their clients, and more secure tools and techniques may arise. Training and instructional material for penetration testers may adapt as well.

The same information may also be useful to those engaged in "active defense"/counter-attacks against threats. Many of the same concerns for security that penetration testers face are also faced by malicious attackers. For those who hunt the hunters, you may be able to apply this information to mapping out potential weaknesses…

Terminology

In this paper and the associated talk, the potential exists to confuse the two types of attackers:

- Those who are conducting attacks on a target organization (penetration testers, for most of our examples)
- Those who are attacking the first category

I will not be distinguishing who is the "good guy" or "bad guy" in this work. The first category might be attackers that want to cause harm to the target organization, and the second might be a group working to stop them. It's a matter of situation and perspective.

To make things easier to follow, the term **penetration tester**, or **pen tester**, will be used to describe those who are conducting attacks on target **organizations**. This is a simplification, as those terms imply the attacks are authorized, when that is not necessarily the case for all useful interpretations of this work. The term **attacker** will only be used in the context of those who are attacking the penetration testers.

Assumptions About Attacker Capabilities

In this work, we have to make some assumptions about how an attacker might be able to compromise a penetration tester. While any potential attacks will be discussed in the context of the pre-requisites needed for that attack to be successful, it's important that we establish what can be considered realistic. This section describes the assumptions that were used in determining the vulnerabilities that penetration testers face.

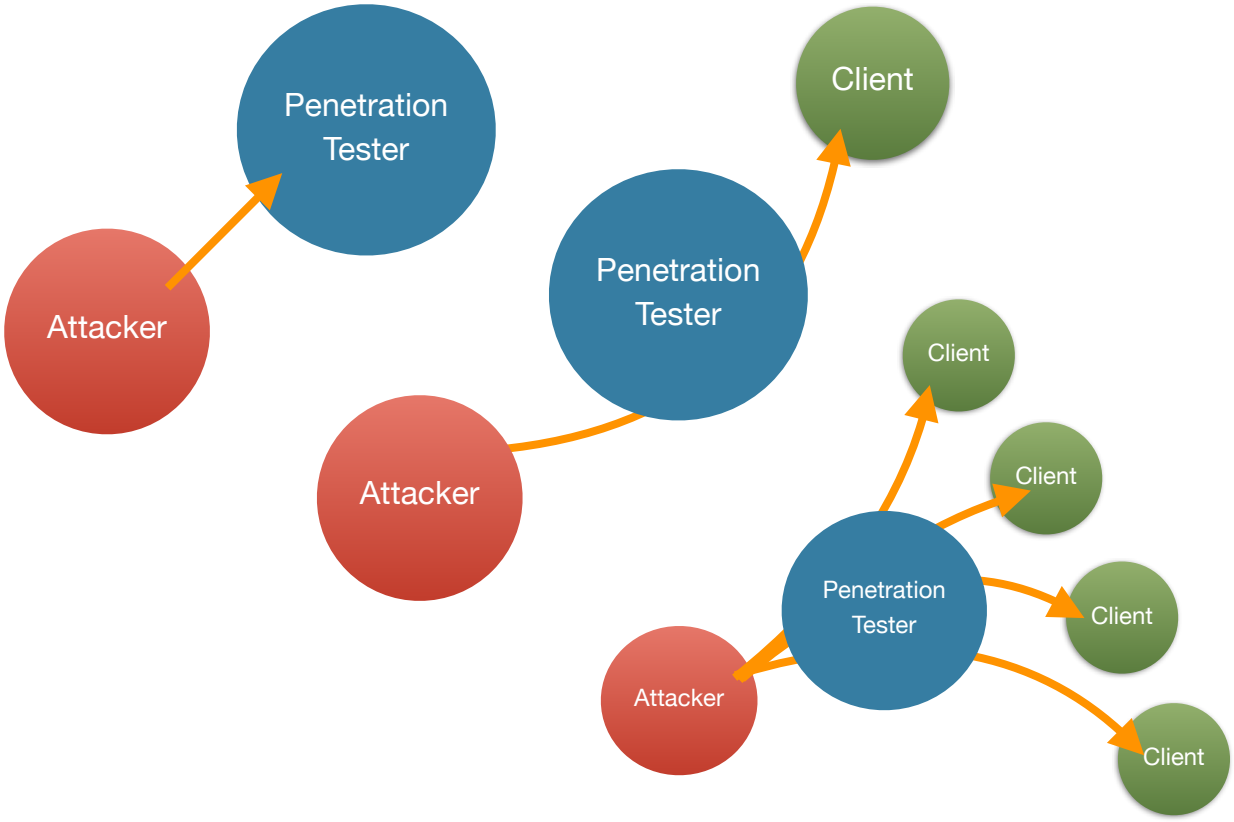
This work largely assumes that an attacker operates with sophistication, skill, and resources that exceeds that of the targeted penetration tester, even though that penetration tester has experience with offensive tools. While it is hoped that this work will help penetration testers repair the imbalance, it is fair to assume at the current time, given the value of the compromised information, that an attacker motivated to compromise a penetration tester is potentially well-funded.

VALUE OF TARGETING PENETRATION TESTERS

Victimology

When an attacker launches an attack against a penetration tester, the ultimate target may vary (see Figure 1). It is possible that the pen tester is the primary victim, in that the attack has been carried out to damage the pen tester in some direct way. Imagine scenarios where the pen tester has personal, business, or consultancy information that is of interest to an attacker that might want to commit some form of fraud. A goal might be to sabotage the operations of a pen tester, or leak their private communications, in a way that embarrasses them among their peers or potential clients. Consider the

Figure 1 - Victims of attackers compromising penetration testers include testers and their clients.



targeting of security professionals by Zero for 0wned [1] or mass remote disabling of Hak5 WiFi Pineapple devices on-site at DEF CON 22 [2].

More valuable and more sinister, however, is the concept of the penetration tester's client base as a target for the attacker. The victim might be a single client, or a persistent compromise of a pen tester might be leveraged into compromising all of the clients that the pen tester engages with over a period of time. If client data is not stored securely by the tester, an attacker might be able to compromise clients for whom the test occurred prior to the compromise of the pen tester.

Goals

Typically, penetration testers of an organization exist outside the normal employee/ account structure, and their access to the organization's network is extensive and, by the nature of a pen test, not constrained to policy. Indeed, it's a pen tester's *job* to explore the possibilities of elevating access where technical and policy measures are not currently adequate. One measure of a "good" penetration test is its fidelity in simulating an realistic attack on a target organization. A compromised penetration tester might accomplish, for a specific or large number of targets, the goals of an attacker that is "riding along".

The attacker may also be seeking to steal tools and techniques from the penetration tester. While most penetration testers are most likely not in possession of zero-day vulnerability information for popular software products, some percentage might subscribe to private exploit feeds or commercial tools from which an attacker might derive value. An attacker who has thoroughly compromised a pen tester's operations might even be able to intentionally modify the results of the pen tester's scans and exploits to "hide" a bug from the pen tester. This would allow an attacker to gain access where the pen tester failed, or to maintain exclusive access to a system they are already on. In this way, an attacker can prevent vulnerabilities from being reported, avoid examination on some systems, and maintain persistence in the organization even after remediation steps are taken post-report.

A penetration tester can make an excellent smoke screen as well. Many penetration testing tools are "noisy", and depending on how a test is scoped, the organization's IT security staff are likely to either know a test is taking place or will be expected to find out. An attacker's activities attacking and setting up persistence are more likely to go unchallenged amongst the traffic generated by the penetration testers, especially if those activities appear to be coming from the same source.

OPERATIONAL SECURITY ISSUES

The techniques and procedures penetration testers use might expose them to attackers as much or more than specific vulnerabilities in the tools that they use.

Standalone Exploits' Payloads

The most popular free collection of exploits used by penetration testers is included with the Metasploit exploit development framework. Metasploit abstracts the payload away from the exploit code for the purposes of modularity, also giving us a set of payloads we can "trust" and plug into a wide variety of exploits. Unfortunately, not all publicly available exploits are in Metasploit, and there is a large body of "standalone" exploits available in various scripting languages. Sites like Offensive Security's Exploit Database[4] collect these exploits, though a penetration tester might find one on a mailing list or less well-established site in the course of researching a particular target's vulnerabilities.

These exploits typically come with a binary payload encoded directly into a string in the exploit itself. These payloads are rarely annotated with regards to the opcodes they represent, and occasionally don't even label the primary goal of the payload as a whole ("*maybe* this launches /bin/sh?"). Many exploits, including some written to ultimately use external payloads (such as exploits distributed for use with Metasploit), contain long encoded binary strings provided as input to the remote service as part of the exploitation process.

Each encoded string and payload represents a part of the exploit's code that the penetration tester must either fully understand or place trust in by association with its source. Non-payload strings could be examined, with knowledge of their impact on the target software. As for the payload, one choice is to disassemble the provided payload in order to verify its expected operation. Another choice would be to replace the payload with a trusted one (perhaps one from the Metasploit Framework), making any adjustments necessary for size and filtering. If a penetration tester explicitly trusts the source from which the exploit was obtained, he or she might forgo these checks.

In reality, this trust decision is often forced by the lack of training and skill in programming, vulnerability analysis, and exploit development among penetration penetration testers. Many testers do not have the necessary experience with disassemblers and reading low-level assembly language that is necessary to understand un-annotated listings of payloads. The details of what makes software vulnerable and how an exploit works to set up the desired state of the target software is considered to be an advanced topic for penetration testers. Some penetration testers lack the versatility needed to move between the many programming languages needed to understand targeted software, exploits, and their payloads.

This is a result of how penetration testers are taught. As a "sexy" profession, with dreams of being a paid "hacker" without fear of prosecution, many are drawn to books and training programs that require little in the way of prerequisite knowledge. It's relatively simple to prop someone up with "just enough" skill to go through the motions of a test, using tools that embody knowledge far beyond that required to launch them. A penetration tester can be safe without this advanced knowledge if they stay well within the boundaries of using code vetted by others, however in cases where an external source has an exploit that may make the difference between a successful and a failed test, one can predict what many pen testers will do.

Attack scenarios for this are not difficult to imagine. A "watering hole" style attack may place backdoored exploits on the public Internet for the target penetration tester (or a wide range of testers) to find and use. A non-functional "exploit" for a version of target software not known to be vulnerable would likely be successful in drawing interest. Most penetration testers have experience with exploit code that simply does not work, and therefore would not necessarily be suspicious after it failed. A working exploit that also introduces persistence for the attacker on either the penetration tester or the target organization would be even more successful. **Many websites where exploits are distributed, ~~including the popular Exploit Database[4]~~, operate over plaintext HTTP, which would allow an attacker in the right position to man-in-the-middle rewrite or replace exploit code being downloaded by penetration testers**. This is no longer true of Exploit Database, as of a recent change in the site!

Data in Transit

A penetration tester will, in the course of their test, interact over networks with the target organization's systems. This will include all phases of the test, but we are especially concerned here with exploitation and post-exploitation. In post-exploitation, we consider the command and control of target systems and the exfiltration of target information. In the exploitation phase of a test, a penetration tester might not have the choice of encrypting communications with the target service, leaving the details of exploitation and at least the probable success or failure of the exploit open to interception.

Upon successful exploitation by a penetration tester, the communications between the target system and the penetration tester are sensitive. Typically included in this traffic are commands and responses as the pen tester interactively (or though some automation) uses the target system, as well as data observed on or wholesale-exfiltrated from the target system. The establishment of this communication and its contents represent a desirable target for attackers.

Post-exploitation communications are usually either handled within the payload, with the most straightforward (and insecure) example being a shell served over a plaintext TCP session, configured by the payload (such as an added OS user), or configured by the penetration tester interactively through a payload. The most versatile payload in the Metasploit Framework, Meterpreter, has used encryption for its communications since 2009. Unfortunately, among free exploits and the payloads contained with them, there are many exploits not in Metasploit, and most of those bring their own payloads which typically do not encrypt traffic. Even when trained in exploit development, penetration testers may not be using payloads that securely communicate.

Extending Networks

Many devices that penetration testers implant for the purposes of remote access allow for command and control via out-of-band communications in order to avoid detection by the target organization. Are these implanted devices also (temporarily) opening up the network for attackers? Are the communications channels (rogue WiFi, cellular data, text) secure?

Data at Rest

Exfiltrated data must be stored by penetration testers for some time while it awaits analysis and reporting. There are a number of questions that should be asked about the security of this data:

- Where is the storage located? Is it physically controlled by the penetration tester?
- If data is on an implanted device (such as a Pwn Plug or WiFi Pineapple), is it physically secure within the target organization? (For this question, if the implant was placed surreptitiously, the answer is likely “no”.)
- Is the data encrypted? If it is encrypted by disk/volume based encryption, how much time does it spend unlocked?
- Where are the keys? Who has access?

- What client data is kept? Is it more than is needed for continuation of the test and reporting?
- Is client data securely deleted after it is no longer needed?

Additionally, the above questions need to be asked separately for past client reports (and any other data that might be stored about a client across engagements, such as notes taken by the individual pen testers). Even outside the scope of attacks launched against penetration tester tools and techniques during an engagement, if a penetration testing company is compromised in a conventional way (phishing, malware, etc.) as any business may be targeted, the compromise could reveal very sensitive client data.

Point of Contact Communications

A well-defined and scoped test will likely include a point-of-contact within the target organization for penetration testers to communicate with. These communications may include starting and ending times for tests, notification of inadvertent crashes, and clarifications on scope. Ultimately, a report must be delivered. Are these communications subject to eavesdropping, interception, or denial of service?

CLASSIFYING PENTESTING TOOL SAFETY

Classifying Penetration Testing Tool Safety

Later this work, we will look at a subset of the tools within Kali Linux as a case study in classifying tools in the system described in Figure 2. The category names on their own require some explanation, which will follow.

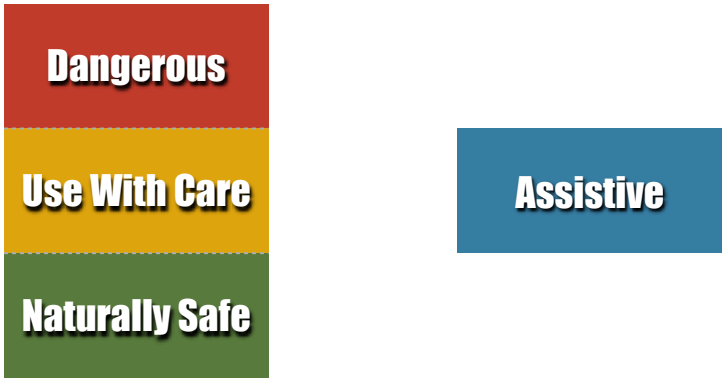


Figure 2 - Classification of Penetration Tool Impact on Security

- Tools classified as **Dangerous** may cause a penetration tester to be particularly vulnerable to attackers. Known vulnerabilities in the tool would contribute to this, as well as communications that are clearly subject to eavesdropping or man-in-the-middle attacks.
- **Use With Care** tools have defaults or common use cases that may lead to situations that would classify them as Dangerous, but can be configured or used in a way that mitigates the risk by someone mindful of the issues laid out in this work.
- **Naturally Safe** tools default to secure communications and are generally safe to use in normal use cases.
- **Assistive** tools are those that are not penetration testing attack or communication tools, but can be utilized to help with the operational concerns described earlier, particularly communication and data-at-rest issues.

Note that these are simply at-a-glance classifications meant to draw penetration tester attention to where it may be needed. The details are often more complex. A tool considered **Dangerous** may be perfectly secure if measure are taken to mitigate its

vulnerabilities and/or attack surfaces (such as being wrapped/tunneled in a secure channel), or if it is used in situations that avoid attackers (such as operating closer network-wise to the target). A **Naturally Safe** tool may in fact be quite dangerous if used outside of its normal use cases or configuration by a penetration tester with improper awareness.

It is also worth noting that so few penetration testing tools have built-in capabilities for encrypting the saved results of their operation, even in cases where that output is designed to be stored for later analysis, that this is not considered in this work's classifications. In all cases, it is recommended that penetration testers implement their own measures for securely storing the results of penetration testing tools executed against client machines.

Case Study: Kali Linux Tools

Offensive Security's Kali Linux is clearly the most popular distribution of Linux for desktop and laptop computers among penetration testers. While the individual tools that comprise Kali are not exclusive to it, the effort and time required to install and correctly configure them all is significant, compared to the ease of deployment and use of the Live CD, installation, or virtual machines. Offensive Security's own training programs make use of Kali Linux, and many other training programs and books do as well.

The following table contains a subset of the Kali tools, color coded with the classification system described in the previous section:

Tool	Classification	Rationale
BeEF	Dangerous	Default pen tester interface is HTTP listening for connections from anywhere, with a default username and password. Recommend at least configuring/firewalling it to only listen on the localhost (or specific remote ones), changing passwords in the config file. Hooked clients communicate with the server via unencrypted HTTP, which may be unavoidable. This is incredibly useful software, though, just be *very* careful with where it's deployed and where the hooked clients are.
sqlninja	Use With Care	Interacts with the target database over a vulnerable web application, so communications-wise you're at the mercy of the target application being accessible over HTTPS. Be mindful of where you launch this from when targeting HTTP-only apps.
dirbuster	Use With Care	This classification could be valid for nearly any scanning software. If pointed at unencrypted services (in this case, HTTP), then your findings are essentially shared with anyone listening in.
searchploit	Assistive	By providing a mechanism for searching a local copy of the Offensive Security Exploit Database acquired as a secure package that would otherwise be accessed through the non-HTTPS exploit-db.com, this tool provides a set of standalone exploits that have gone through at least some vetting.
Metasploit exploitation with Meterpreter payload	Use With Care	Metasploit has a lot of functionality, but specifically for launching an exploit and deploying a meterpreter payload, the communication channel is fairly safe. An attacker may be able to observe and conduct the same attack, though.
SET with Meterpreter payload	Use With Care	Similar rationale as Metasploit. The resulting channel is safe, unless you are hijacked on the way there.
cymotha	Dangerous	None of the provided injectable backdoors offer encryption. Could potentially modify this to include some more robust backdoors, or use the "script execution" backdoor to configure an encrypted channel.
nc	Dangerous	Good old vanilla netcat, like your favorite book/trainer taught you, gives you nothing for communications security.

Tool	Classification	Rationale
ncat	Naturally Safe	Netcat, but with SSL support that one can use. You'll need to set up certificates for it.
dbd/sbd	Use With Care	Another netcat clone with encryption. Easier to set up than ncat for encryption, but relies on a shared passphrase that you'll have to be careful about setting on either end.
gpg	Assistive	Provides the capability to encrypt data at rest and prepare sensitive data for transit
truecrypt	Assistive	Provides the capability to encrypt data at rest and prepare sensitive data for transit

Overall, Kali Linux itself has to be considered as **Use With Care**, both as a combination of tools of varying classifications, and with the operating system itself being configured primarily to support its set of tools, rather than a secure computing environment. For example, most Kali users operate as the root user for the majority of the time.

SECURITY OF IMPLANTABLE DEVICES

Pwnie Express Pwn Plug

The author of this work previously presented, at DEF CON 21, some initial work studying vulnerabilities in penetration testing devices. This work focused on vulnerabilities in the web interface of the commercial firmware for the original Pwn Plug device[6]. This device is meant to be an implantable device, easily mistaken as a power supply for a device such as a printer, and provide a penetration tester with remote access to an internal network. This was the first of the Pwnie Express "sensor" devices, and is currently sold as an "Academic Edition" device [5].

The work performed by this author included a procedure for acquiring a forensic image of the device for the purposes of extracting information about its operator. At the time, version 1.1.2 of the commercial firmware was vulnerable to command injection via a combination of XSS and CSRF in the "plugui" web interface. To exploit the vulnerability, crafted packets were sent to the device's sniffer in order to make the user interface display crafted web requests that included the XSS/CSRF/command-injection payload. A script was then uploaded to the device for persistent access and continuous exfiltration of new information logged by the device.

Pwnie Express has since expanded their offerings into a series of new "plugs", as well as phones and tablets configured for penetration tester use.

Hak5 WiFi Pineapple Mark V

The vulnerabilities that a WiFi Pineapple introduces into a penetration test for the client and the tester are difficult to avoid. This has been a popular device for years, and it's just recently becoming safe to use under certain circumstances. The Pineapple is sold as an all-purpose penetration testing appliance, able to perform a variety of wireless attacks and interceptions, as well as act as an implantable remote access solution, through which a penetration tester can launch further attacks on a target organization. Testing wireless security involves a lot of risk if it is assumed that there are other bad actors in the area eavesdropping.

Versions of the WiFi Pineapple firmware released before DEF CON 22 in 2014 (versions prior to 2.0.0) were vulnerable to a bypass of the authentication on the web-based interface. Authentication was being performed in the footer of the PHP code, simply not displaying the rendered page if authentication of the user did not check out. This made it possible to blindly inject commands into a variety of interfaces within the web-based administration panel. This vulnerability was demonstrated by the author at DEF CON 23[2]. This vulnerability was clearly and directly exploitable in an automated fashion, giving an attacker assured access to a penetration tester's device, within range.

In the months that have passed since DEF CON 22, an effort by the Hak5 developers has been put into making the Mark V a more secure device for its operator. The latest version of the firmware, 2.2.0, includes a separate wireless interface specifically designed for the administration of the device (on older versions, wireless administration was accomplished on the same unencrypted interface as the one that was opened for victims to connect). Authentication is now checked in the `header.php` prior to any other action, and anti-CSRF code has been inserted into the header as well. For all actions, other than those that are authentication-related, a CSRF token must be set that matches the SHA1 hash of the PHP session ID.

The only code of use not protected from cross-site request forgery is the authentication code. An attacker able to draw an operator's web browser into submitting a GET request to the Pineapple interface would be able to log the operator out of the interface. This requires a currently-interactive operator and the ability of an attacker to draw the operator's interest to a non-Pineapple-interface site. While this scenario isn't out of the question, the payoff for an attacker is very limited.

While much of the process of checking for and obtaining upgrades from within the Pineapple interface is performed over HTTPS to wifipineapple.com, the download of the actual update file is performed over plain HTTP in /pineapple/components/system/info/includes/files/downloader. The corresponding installer script does not check any kind of signature, simply installing whatever image is located at /tmp/upgrade.

For an upgrade instantiated over the web interface, the size and MD5 hash of the upgrade is acquired over an SSL connection, and resubmitted by the operator's browser back to the web interface to be checked before the installation begins. This makes this upgrade process only as secure as the XSS/CSRF protection of the interface, rather than on the strength of a hash. The manual upgrade process is more secure. The firmware download page is HTTPS by default, and the manual upgrade instructions specifies that the MD5 hash should be verified by the operator.

By the classification system defined in this paper, the current version (2.2.0) of the WiFi Pineapple Mark V firmware is considered classified as **Use With Caution** as an attack platform, primarily due to the exposure to tampering it faces in its natural unattended use cases. It is important to note that this classification is with the caveat that if Pineapple features are used to set up open rogue access points, then it naturally exposes the users of those access points to eavesdropping by third-party attackers. This is, however, inherent to the nature of the device and what it tests, so it may be ultimately unavoidable. This should be discussed with the client when scoping the test, and care should be taken to limit how and for what duration this feature is used.

Comparatively, the 2.0.0 version and prior (those that were vulnerable to @ihuntpineapples-style attacks), along with any firmware version of previous hardware WiFi Pineapples (Mark IVs and older), should be classified as **Dangerous** for use in penetration testing. Any "clone" devices home-configured or sold with older WiFi Pineapple software (such as the "WiFi R00tabaga" in Pineapple mode, or other miniature routers described on the net) should also be classified as **Dangerous**.

While penetration testing devices are attractive to inexperienced penetration testers, safe operation of any current implantable penetration testing device is going to depend very heavily on preparation and the skill of the penetration tester that deploys the device. If such devices are used on a real penetration test, planning should include a discussion of how each issue raised in the operational security issues above will be addressed.

CONCLUSIONS AND RECOMMENDATIONS

Penetration testers are valuable targets and frequently and uniquely vulnerable as a result of their tools, techniques, and training. This vulnerability has serious consequences for the penetration tester being targeted by an attacker, as well as the body of clients that that penetration tester serves. It is hoped that an awareness of the

issues raised in this work, and a system for classifying tools, will help improve the security of penetration testers.

The following *specific* recommendations may also help:

Operational

- Test tools and exploits before deployment into a real test, as much as possible. Never launch a tool or exploit that you don't fully understand and/or trust.
- Be aware of what information is exposed in exploitation and post-exploitation connections you make to the client. Know which ones of your tools, exploits, and payloads encrypt for you.
- Be aware of the network environment between you and the client, and if the information exposure cannot be suitably mitigated, attempt to reduce the network distance between your tools and the client. For example, launch an insecure tool from a beachhead within the client network, then encrypt and transfer the results.
- Take care when "extending" a client network with an access point or covert channel. Are you opening that network up for another attacker?
- Keep client data in an encrypted state unless you are analyzing it or writing the report. Having it on your whole-disk encrypted computer that never turns off is not good enough.
- Securely delete any client data not needed between engagements. Encrypt the rest, including reports.
- Communicate results to the client in person, or over a secure medium.

Training and Instructional

- Discuss the role of secure communications and handling of client data.

- Where possible, teach a penetration testing exploitation and post-exploitation process that focuses on establishing a secure channel before exfiltration.
- Do not treat penetration testing as something that can be undertaken without an understanding of programming, vulnerability analysis, exploit development, and basic operational security. **When we lack the capability to understand our tools, we operate at the mercy of those who do**.

REFERENCES

[1] zf0 ezine, http://web.textfiles.com/ezines/ZF0/

[2] Ms. Smith, *Hacker Hunts and pwns WiFi Pineapple with zero-day at Def Con*, http://www.networkworld.com/article/2462478/microsoft-subnet/hacker-hunts-and-pwns-wifi-pineapples-with-0-day-at-def-con.html

[3] Carlos Perez, *Meterpreter Stealthier than Ever,* http://www.darkoperator.com/blog/2009/7/14/meterpreter-stealthier-than-ever.html

[4] Offensive Security, *Exploit Database,* http://exploit-db.com

[5] Pwnie Express, *Pwn Plug Academic Edition*, https://www.pwnieexpress.com/product/pwn-plug-academic-edition/

[6] Wesley McGrew, Pwn The Pwn Plug - Analyzing and Counter-Attacking Attacker-Implanted Devices, https://www.defcon.org/images/defcon-21/dc-21-presentations/McGrew/DEFCON-21-McGrew-Pwn-The-Pwn-Plug-WP.pdf