HORNECYBER

PROTECTING PENETRATION TESTS: RECOMMENDATIONS FOR IMPROVING ENGAGEMENT SECURITY

Dr. Wesley McGrew
Director of Cyber Operations
HORNE Cyber
@McGrewSecurity
wesley.mcgrew@hornecyber.com

This paper, and its associated presentation, represents a capstone to previous years' work by the author on the subject of vulnerabilities that are present in penetration testing tools, procedures, and learning materials. These vulnerabilities and common practices have been shown to unnecessarily put client systems and data at risk. Systems and infrastructure used by penetration testing teams are also at risk of compromise, in the style of "ihuntpineapples" or worse: quietly and over a long period of time.

In this work, the author presents a comprehensive set of recommendations that can be used to build secure penetration testing operations. This includes technical recommendations, policies, procedures, as well as information on how to communicate and work with client organizations about the risks and mitigations. This paper addresses the balance of making testing capabilities more professionally sound, while avoiding a negative impact on the speed, agility, and creativity that good testers are able to apply to engagements with current practices.

MOTIVATION

PRIOR WORK

AUTHOR'S PREVIOUS PRESENTATION

Vulnerabilities that jeopardize the security of the penetration testing process have been explored in previous talks on the subject by the author at DEF CON and Black Hat USA [1,2,3]. Two of these talks have demonstrated technical vulnerabilities in offense-oriented tools, and described how a malicious third-party attacker would be able to compromise the security of the tester, the client organization, or both. A third describes the operational security issues that arise due to vulnerable practices being taught in common penetration testing learning material. This section serves to describe some of this prior work by the author, and to cover a selection of work by other vulnerability analysts in this area.

The presentation Analyzing and Counter-Attacking Attacker-Implanted Devices at DEF CON 21, in 2013 by the author explored forensic and vulnerability analysis of small hacking "drop boxes". Such "embedded",

"low power", or otherwise portable and concealable systems are frequently used by penetration testers to leverage physical access to a client network into a continuously accessible backdoor/attack-platform [1]. The whitepaper and demonstration for this talk used a case study of the Pwnie Express Pwn Plug concealable platform to illustrate how a maliciously-installed implant could be analyzed and counter-attacked. The focus of this presentation was on the technical aspects of analyzing the device, with brief mention to the implications of security in the penetration testing process.

At DEF CON 23, in 2015, the author presented I Hunt Penetration Testers: More Weaknesses in Tools and Procedures [2]. In this presentation, the concept was expanded and applied to the problem of operational security in penetration testing. Scenarios in which a testing team or client organization might be attacked successfully by a third party were discussed, along with a case study of vulnerabilities identified in the popular Hak5 Wi-Fi Pineapple device. A brief study was also conducted of a subset of tools in the Kali Linux distribution with a focus on the confidentiality of client data and communications. Since this presentation, many improvements have been made to the security of the Wi-Fi Pineapple platform.

The following year, at DEF CON 24 and Black Hat USA 2016, Secure Penetration Testing Operations: Demonstrated Weaknesses in Learning Materials and Tools, the author explored a fundamental problem in offense-oriented training material [3]. A study identified that an overwhelming majority of books, classes, and other learning material used to develop talent in the field of penetration testing were found to contain examples and instruction that lead to poor operational and communications security during penetration tests. It was established that a properly designed toolchain that had the capability of being secure can be used in insecure ways through a lack of training and awareness.

In previous presentations, recommendations have been made to improve the situation, though only briefly as part of concluding remarks. The focus has been on defining and illustrating the problem though examples of vulnerabilities in commonly-used attack tools. Now that the problem has been clearly presented, the goal of this work is to go beyond brief recommendations and provide a meaningful discussion of actionable characteristics that define a more secure penetration testing practice.

OTHER RESEARCHERS

Justin Steven has recently disclosed a set of directory traversal vulnerabilities in the Metasploit Framework [4]. The vulnerability allows an attacker with a modified version of Meterpreter to write arbitrary files to the Metasploit operator's file system. Attackers who are targeting penetration testers could use this vulnerability to gain code execution on the testers' workstations, or their deployed appliances. Justin Steven is also responsible for identifying vulnerabilities in the Metasploit Web UI, that, when combined, would allow a third-party attacker on a penetration test to gain access to the penetration tester's system that is hosting the Metasploit instance [5].

Justin's stated motivation for analysis and disclosure is directly in line with this author's concerns and motivations for presenting the current and previous years' work in secure penetration testing. In Justin's words [4]:

Users of software, including security assessment and exploitation tools, should take steps to mitigate against unknown bugs. These tools provide network services and directly interface with untrusted hosts. Run your tools in isolated environments. Compartmentalize your work. Don't run software as root if it can be avoided. Don't let your security, and the security of your clients, depend only on the correctness of the code you run.

Rapid7 addressed the vulnerabilities disclosed by Justin, and recommended the use of Meterpreter's "Paranoid Mode" functionality, which layers stronger authentication onto the payload. This functionality was identified in this author's work presented last year as a strong mitigation against demonstrated hijacking of command-and-control during penetration tests. Justin's findings support the importance of software security and sound procedure to the operational aspects of the security testing process, and more practitioners are becoming aware of the issues as a result.

Fuzzing protocol dissectors in packet sniffing software has proven to be wildly successful. Historically, many vulnerabilities have been identified in Wireshark's dissectors [11], including some used during DEF CON CTF [10]. More recently, however, a large number of vulnerabilities have been identified by fuzzing the tcpdump sniffer. A recent security advisory puts this at 41 distinct issues [12], including many identified by Brian Carpenter [13]. Many security tools make use of complex parsers, likely subject to the same types of problems.

FEEDBACK

Feedback received from previous presentations on the topic from penetration testing practitioners has been positive, with an attitude favorable to the stated need for change. Many had briefly considered the issue before, and perhaps had noticed vulnerabilities on their own, but had not considered the full scope of the problem. Some had made use of such vulnerabilities before. Of these, the author expected to hear from individuals who had taken advantage of weaknesses in penetration testing tools during "player vs. player" capture-the-flag exercises. More often, however, it was mischievous blue team members stretching the rules of engagement of a red teaming exercise.

Stakeholders in organizations that have contracted for penetration testing engagements have responded to the previous work with concern about the quality of software development and training applied to the practice of penetration testing. Some have described incidents involving "backdoors" inadvertently left by penetration testers as artifacts of testing. These stories mirror incidents observed by the author's teams on engagements.

Unsurprisingly, while previous talks presented by the author were seen by the audience as entertaining (with live demonstrations of vulnerabilities identified) and informative in content, the most common question of all across them is the question that all "breaker" talks are, or should be, asked:

How do we fix it?

This paper and its associated presentation attempt to address this question.

TERMS

The following terms are defined for the purpose of consistently presenting this paper's recommendations. The field of penetration testing is saturated with the use of non-standardized terminology. The scope of this section covers terms used in this paper, and does not represent an attempt at addressing the larger problem of vocabulary confusion.

The phrase "**offense-oriented services**" is used as a blanket term for the spectrum of engagements that are often described as **penetration testing** and **red teaming**. Put in very broad and simple terms, penetration testing involves the identification of as many vulnerabilities as possible in the organization's attack surface, while red teaming engagements test security measures and incident detection/response through the slower and more deliberate exploitation of a smaller set of vulnerabilities. In the author's experience conducting tests, many penetration tests are scoped to include elements of red teaming, to provide additional value on top of a more comprehensive test. For this work, recommendations for secure penetration testing are also applicable to red team services. The term "penetration testing" is used most frequently in this work, but for the purposes of this research, the recommendations should be interpreted to represent a broad spectrum of service types.

When papers about attackers that attack attackers are presented, roles can get confused. As in previous papers and presentations by the author, a **third-party attacker** is an attacker that, outside the scope of the engagement, seeks to maliciously compromise the penetration tester or target organization. A **malicious attacker** in the more general sense will include those that are attacking organizations through conventional means. In this work, penetration testers are attempting to identify the vulnerabilities that a might otherwise be found by malicious attackers. An **attacker**, without additional qualification, is used to identify an actor operating an attack tool which may, in a given situation, be a penetration tester or a malicious attacker.

Post-exploitation **command and control** of an exploited system will be referred to as **C2**, as it is frequently abbreviated in the field of malware analysis. This includes more sophisticated session-based post-exploitation, such as that provided by the Metasploit framework in the Meterpreter payload, as well as simple shellcode.

RECOMMENDATIONS
QUALIFIED TEAM

Much has been written, within the penetration testing community, about the qualifications required to perform penetration testing. The term "script kiddie" has, for decades, been a term the larger hacker community has used to describe individuals that run attack tools, expecting results without a deeper understanding of the underlying technology. Certifications are known among practitioners as being an insufficient measure of a tester's capability, though those certification programs are still widely used in hiring decisions and sought by individuals who wish to get a "foot in the door".

It should become apparent to the reader, as this paper's recommendations are presented, that the design, implementation, and execution of a secure penetration testing operation relies heavily on the individuals that are part of the testing team. Performing a penetration test while ensuring the security of team operations and

the client network requires more training and knowledge than performing a test in the traditional way. While a lack of foundational knowledge is known to limit the ability of minimally-trained and qualified penetration testers to find many of the vulnerabilities that skilled malicious attackers might find, the same tester should now be considered to be an even more clear and present danger to the client network--a vulnerable extension of the network's attack surface during and potentially after the engagement.

It is important to build a team that has an understanding of the underlying technology that their tools and their targets operate on. A team that is built upon training that is specific to the operation of vulnerability scanning and exploitation software may be able to go through the motions of a penetration test, but will not understand their own attack surface (nor anything else that is not explicitly covered in their training). They may have no awareness of whether their C2 is suitably protected, and no background to apply to the evaluation of new tools and exploits.

Software vulnerabilities are frequently the result of the developer's misconceptions about a system's layers of abstraction. Put plainly, the developer has their perception of how the system works and develops for that perception. When the perception does not map to reality of the underlying system, errors are made that can only be discovered and exploited by those who deeply understand the lower levels of abstraction. For example, many C++ programmers believe that there is some form of watchdog or protection that prevents unauthorized access to private member variables of objects. The reality is that this is typically a syntactic convenience for the developer and checked during compilation only, with nothing protecting those variables in memory at runtime.

To conduct a secure penetration test, team members must be able to understand vulnerability information that is published about penetration testing tools, and evaluate the software that they use to determine the amount of trust that can be put into it. Team members should also have the capability to understand vulnerability information and evaluate the potential impact of an exploit on the target system. The skill set should include proficiency in:

- **PROGRAMMING**
  The ability to read, understand, and identify potential vulnerabilities in a large number of programming languages is as crucial to identifying vulnerabilities in penetration testing and exploitation tools as it is to the analysis of mainstream IT software.

- **PLATFORMS**
  The ability to understand the software's execution environment is key to understanding that software's vulnerabilities, and to evaluate the impact that attack tools will have on systems. This includes everything from the frameworks an application might be running on top of, down to the operating system it's hosted on.

- **NETWORKING**
  A fluency in networking is required by penetration testers to monitor client organizations' networks. It is difficult to imagine a penetration tester, even one reliant on automated tools, that would be able to operate without this knowledge. This same knowledge can be applied to checking penetration testing procedures' attack surface and potential for information exposure (unencrypted protocols, or traversal of untrusted networks).

Taking this into account seriously, care should be taken to recruit and build talent that has a depth of knowledge and experience far beyond brief training courses. A university program in computer science or engineering, absorbed in earnest by a student, is an excellent foundation, though it is certainly also possible to pick up the equivalent knowledge and experience independently or vocationally.

Building a secure penetration test requires a team with higher qualifications. Team members with a fluency in programming, platforms, and networking will have the fundamental skills needed to implement this set of recommendations.

VULNERABILITY ANALYSIS

The act of taking an adversarial look at the tools used to conduct penetration tests is a critical part of conducting secure tests for clients. Each of the previous talks by the author on this subject included the results of analyzing penetration testing software and practices for vulnerabilities that could be leveraged by third-party attackers. This recommendation should almost be a given, though it is important to point out that each team will need their own capabilities in this area.

Why will you need to perform some of your own vulnerability analysis? Consider a penetration tester that comes across a "custom" web application during the course of their testing, internally (or contractually) developed. Most testers see this as a gold mine: a body of code that has likely not seen the attention of mainstream security researchers, if it has even been peer reviewed at all. Vulnerabilities are often easy to identify in such systems. This same principle applies to penetration testing tools.

Interest in the vulnerability analysis of penetration software (or at least, in the public disclosure of vulnerability information) will never match that of widely used mainstream IT software. Penetration testing software is frequently developed in an ad-hoc fashion to demonstrate vulnerability, without the benefits of efforts towards secure development (and the reduced speed of development that comes with them). Most software in this field, outside of a few of the most popular projects, is not under continuous development and maintenance. At this time, if you expect to evaluate the trustworthiness of penetration testing software you wish to deploy and use, you must expect that there will be many vulnerabilities, and that your team will have to the heavy lifting of identifying those vulnerabilities. Much like your client organizations' custom code, your custom tools will also never see the benefits of vulnerability analysis unless your team is tasked with conducting it themselves. At a "meta" level, this also applies to the processes you use that combine your tools and manual testing into a coherent test.

At the smallest extreme of this recommendation comes the act of vetting new exploit code. Frequently, newly identified vulnerabilities have standalone proof-of-concept exploit code associated with them. You will likely want to be able to test your client organizations' networks for these vulnerabilities. Can your team read the advisory and exploit code and determine the suitability and safety of those exploits for their task? Does the exploit contain a backdoor? Would launching it against a host under third-party attacker control allow that attacker to gain code execution on the penetration tester's workstation?

Intensive vulnerability analysis must be undertaken to identify vulnerabilities in the software and processes you use to conduct tests. New tools and exploits that are to be used on engagements should be adequately vetted.

IMPACT ESTIMATION

Related closely to (and possibly overlapping) vulnerability analysis, impact estimation involves penetration test members performing analysis of their tools, exploits, and procedures as it relates to the state of the systems being tested. For confidentiality, integrity, and availability of the target systems, the following questions should be asked:

- Does the exploitation process leave the system being attacked more vulnerable immediately? For example, are security features being disabled on a system-wide scale?
- Does the continuous process of post-exploitation C2 with the tool leave the targeted system, or the data flowing across the C2, vulnerable? Is there authentication and encryption associated with the C2? Does it "listen" for connections, or does it connect back to the tester?
- Can changes that impact the security of the target be reverted by the tester after the system is no longer being actively examined and leveraged by the tester? Will it require intervention from the client organization's IT staff?

Analysis and laboratory testing of the tools and processes used by the testing team in the context of these questions will help illustrate what operational changes need to be made to minimize the impact of the testing on the security of the target systems.

Team members should have the capability to assess and test the impact of exploitation tools and procedures on the security of the target systems before executing them against client organization systems. This will help the team determine what changes need to be made to the operational use of tools on engagements.

LAYERING PROTECTION

Like the implementation of security on a traditional IT network, the security of penetration testing operations benefits from layering security measures--"defense in depth". An exploitation tool's built-in C2 channel may be determined to be unsafe to operate over a hostile network, such as the public Internet or an open Wi-Fi access point. In these situations, it might be possible to continue the test and carry out the attack more safely by tunneling it through another protocol. By layering defensive measures, more trust can be placed in processes and tools.

Many common penetration testing tools, tactics, and procedures should be considered as being unsafe to conduct over publicly accessible networks, as demonstrated in previous works. Many penetration testers already make use of "drop box" devices to gain and maintain access to the internal network of a client organization remotely. Such devices, if configured and used correctly, can be used to change the "source" of a penetration tester's attack to a network location that is more closely controlled and monitored by the client

organization's IT staff. If the number of network "hops" that a potentially insecure tool operates over can be minimized, the security of the procedure is improved.

In the author's experience, such devices are best configured with the ability to securely connect back to penetration tester's systems via a VPN. The use of encryption and certificates will reduce the potential for the connection to be monitored or hijacked, and the versatility of the VPN will allow a variety of tools and processes to operate over that secured connection transparently. The VPN should be monitored for third-party attacker abuse.

Many of the other requirements presented in this paper revolve around the layering of security. In fact, implementation of a set of best practices will, by the nature of the recommendations, result in measures that overlap in protection. The ability to analyze one's own toolchain enhances the host security of the penetration testing workstation, which, in turn, enables the penetration testing team to make better decisions about what attack activities can be considered "safe". Ultimately, human oversight and client communication help the soundness of all the recommendations that have been put into practice.

Security of the penetration testing process, like most aspects of information security, benefits from defense in depth. Security measures should be taken that enhance the effectiveness of other measures. Fail safe.

CLIENT INVOLVEMENT

Penetration tests illustrate risk, a combination of threat, vulnerability, and impact, to the client organization. It is equally important to have a discussion with the client about the risks associated with the act of testing. In current best practices, this conversation revolves around the availability and stability issues that penetration testing can introduce into an organization.

While an experienced team that knows how to test the equipment they are facing is less likely to "knock over" production systems on a test, it can be difficult to avoid in some cases. Every penetration testing team can tell you stories about devices that fail when port scanned. One of the author's teams recently tested a network containing a door access controller that would completely freeze upon receiving an HTTP request for the root page of its embedded web server (the developers never imagined that there would ever be a request that did not go straight from their application to the controller's web API URLs). When availability incidents are unavoidable, an open line of communication with client stakeholders and IT staff allow the incidents to be identified and resolved quickly.

This line of communication is of critical importance to the process of secure penetration testing as well. In the early stages of negotiating an engagement, the client should be made aware of the current state of the art in secure penetration testing, and the steps that the team takes to mitigate the risks. This serves to educate the client and to make them aware that the team does take the security of the organization's network seriously. The same channel that is used to communicate availability and stability issues should be used to communicate other security concerns related to the test.

This communication should allow the client and penetration testing team's representatives to communicate and share information securely. This may be accomplished through end-to-end encrypted emails or a secure "drop" for files. It should also involve a more immediate form of communication that does not rely on the availability of the client or penetration testing teams' networks. This may take the form of exchanging cell phone numbers.

Communication should be perceived by both parties to be open and will likely be frequent. It should be made clear to the client that signs of an attack during the period of the engagement should be verified as being part of the test with the penetration team's representation. This includes events identified by security monitoring, changes observed in systems behavior and performance, and phishing attempts that have been reported. Most of these will be verified quickly as being associated with the test, but, in the author's experience, some anomalies that a client reports with the caveat of "this is probably your team, but…" wind up being attributed to malicious attackers. It is important for the security of the client organization's network that the test not serve as a "smoke screen" for the malicious activities of real attackers.

Many penetration testers hesitate to provide information to the client on vulnerabilities identified until they deliver the report at the end of the engagement. Testers want to avoid having the client IT staff "chase" them around the network, closing avenues of attack (and C2) until reportable information about the target has been extracted and further pivot-able vulnerabilities have been identified. This aids a testing team in determining the impact of the vulnerabilities, an important factor in measuring and communicating risk.

While this approach is normally appropriate, there are instances in which it's more important to provide information prior to reporting, soon after discovery, through secure channels:

- …when the nature of the vulnerability identified in the client organization's network is such that it is at risk of being exploited by an external attacker at any time. This may include systems on the external attack surface of the network that require no authentication (or manufacturer's defaults), or that were exploited using widely available, publicly disclosed vulnerability information

- …when the presence of a third-party attacker is identified by penetration testers in clients' systems

- …when a third-party attacker is identified as attacking the penetration testing team's processes, tools, or infrastructure (reported so that information can be corroborated with the client's network security monitoring, and as an impetus for the client to begin incident response if it occurred on the internal network)

- …when the exploitation of a vulnerability has left a client organization's system in a less secure state than when it was first encountered (for example, by disabling a security feature or leaving a "back door" that cannot be removed by the tester without IT staff intervention)

- …when evidence is found of weaknesses introduced and left behind during previous penetration testing teams' engagements (the author's teams have found persistent Meterpreter shells on client systems that were later attributed to previous firms' engagements)

Open communication should flow both ways between the client organization and the penetration testing team leader. Anomalies identified by the client should be verified with the team, and risks of high and immediate concern should be communicated back to the client in timely fashion. A solid understanding of risk and trust in the penetration testing team will allow the team to more effectively monitor and protect the security of the ongoing engagement.

DATA PROTECTION

Client data should be protected in transit and at rest to prevent disclosure as a result of testing procedures. The penetration testing firm's proprietary information (tools, vulnerability information, processes) requires the same level of care. Testing procedures should be chosen that protect data, and adversarial self-testing should focus on ensuring that inadvertent disclosure is not an issue.

Many penetration testing tools were identified in previous work by the author that provide C2 of compromised systems without sufficient encryption. While functional, such C2 can expose sensitive data being viewed or exfiltrated by the penetration testing team. The nature of the targeted application or system may be such that it is infeasible to encrypt the C2. If this is the case, it is important that best practices should be used to eliminate or minimize the risk (see the recommendation for "Layering Protection"), and that the duration of time that the C2 is active should be minimized as well. The sensitivity of the information that is being targeted for exfiltration should factor into a decision to proceed.

Client data should be protected at rest as well. This data includes information targeted for exfiltration, as well as the products of testing procedures. Testing produces scan results, tool output, logs, notes, and, ultimately, reports. If stolen, this accumulated data may directly serve as a roadmap for third-party attackers. Maintaining host security (especially access control and audit trails) of the systems that contain this data is important. The footprint of this material should be reduced as much as possible. For example, sensitive data should be moved off of implanted devices, laptops, and mobile devices to the penetration testing team's local storage as soon as possible to minimize the impact of a deployed device being compromised.

As described in the previous section, client communication should be conducted over as secure of a channel as possible. Report delivery and distribution should be treated with care, as it is the product of a large amount of sensitive client and penetration tester information. Frequently, reports contain more sensitive information than any one employee in the client organization should have been trusted with (for purposes of separating privilege and encouraging checks and balances).

The penetration testing firm should be prepared to, post-engagement, securely delete information that is sensitive to the client organization. An overall policy of data retention should address the need to keep work products, such as reports, along with procedures for processing sensitive data to extract what is needed for continual improvement of the penetration testing process (notes about vulnerabilities and tools that are independent of specific client environments), while securely removing the rest. Data that is retained, such as any information about a target network that is kept as part of an ongoing client relationship, should be kept in an encrypted state between engagements. These policies should be clearly communicated to the client to

provide them assurance, and to ensure that the policies are compatible with the client organization's data protection needs.

Sensitive data about the client and the penetration testing team's activities should be encrypted in transit and at rest. Communication between representatives of the team and the client should be secure. Data retention policies should be clearly communicated to the client organization.

SCOPING CONCESSIONS

Penetration testing and red teaming services provide value by identifying vulnerabilities that are not easily identified as part of an organization's normal IT processes. While the IT staff may be attempting to follow best practices, those practices may not cover the full gamut of misconfiguration, and staff may not have the appropriate training to identify weaknesses in applications in network protocols. A "fresh set of eyes", taking an adversarial approach, will find things that have been overlooked.

An important characteristic of this type of testing is **threat emulation**. Penetration testing should use, at some level, the tools, tactics, and procedures employed by real attackers. By taking an approach to penetration testing that is similar to the approach used in real-world breaches, a test is more likely to cover the set of vulnerabilities identified and used in a real breach. Tests that make use of automated tools unlikely to be used by real attackers provide less coverage.

Scoping a penetration test with a client involves a discussion of how closely the testing team will emulate tools, tactics and procedures of real attackers. Generally, it is considered best to do so with as high of a fidelity as is possible. Clients wish to get an attacker's viewpoint on the network, and the penetration testing team wants the freedom to explore the attack surface of the organization without scoping limitations.

Concessions are made in scoping, however. Sensitive or otherwise fragile systems are tested during hours when downtime is acceptable, or when emergency contacts are available. Availability attacks may be avoided. Engagements including social engineering should not violate the privacy of employee's personal social networking accounts. For a variety of reasons, such as meeting client requirements and addressing legal restrictions, an offense-oriented service engagement will be limited in some way compared to the actions that a malicious attacker might take.

Protecting a penetration test will also require concessions to realism. The most frequent concession comes when defining the boundary between tests of the external and internal attack surfaces. When conducting a test of the external attack surface of an organization, vulnerabilities may be exploited across the public Internet. Depending on the details of the vulnerability identified and exploit used, the post-exploitation C2 may not be secure. While verifying exploitability of vulnerabilities is an important part of a penetration test, extensive post-exploitation and pivoting in the external engagement might be better left to be explored in the internal engagement, if a secure C2 cannot be established.

The breadth, depth, and rules of engagement for a penetration test should consider the security of performing the test on the client organization's systems.

STANDARD OPERATING PROCEDURES

Penetration testing, when conducted effectively, is a process that defies rote procedure in favor of dynamically and creatively finding weaknesses in the client organization's systems. A set of overly rigid standard operating procedures will have a negative impact on the agility of the test, and therefore its effectiveness (as discussed in the section Challenges: Agility). A penetration test that is executed solely "by the book" is likely to be no better than one that primarily relies on automation.

While naturally abhorring strict order, it is necessary for a team to follow some standard procedures for exploring a target network and enumerating vulnerabilities. In addition to the guidelines of the Penetration Testing Execution Standard (PTES) [6], most teams should already have some procedures in place for:

- Division of labor
- Collecting findings
- Reporting

Despite being dynamic and responsive in their approach, there will be standard tests that will be applied to each device on a target network. A team member would not, for example, examine an Internet-of-things device identified on the network without consulting documentation or notes for the manufacturer's default password. These procedures are put into place to ensure that the penetration test is comprehensive and consistent across the network, across team members, and across multiple engagements.

In the same way, documented standard operating procedures can be used to ensure that the principles of secure penetration testing are applied consistently by all team members on all engagements. This allows for security to be addressed by habit, rather than when the team "thinks" about it. Common situations that lead to vulnerabilities for clients and penetration testing teams themselves can be addressed as a matter of policy.

For example, consider the management of credentials created during an engagement. Many vulnerabilities and post-exploitation activities involve the creation of accounts by penetration testers (often with a high degree of privilege on the target). Without adequate documentation of these accounts as they are hastily created in the process of gaining access, it may be difficult to confidently produce a comprehensive list of them for "cleanup" at the end of the engagement. If accounts are left in place post-engagement, or unnecessarily vulnerable during the engagement, third-party attackers may take advantage of them. In the author's experience, such accounts have been identified and exploited on engagements that were later found to have been created during previous penetration tests by other firms.

Some of the concerns that need to be addressed for this specific example would include:

- Creation of secure username and password pairs. It is dangerous to use simple "admin/admin" style credentials, or to re-use the same password across multiple parts of the engagement, though it can be tempting (for expediency, knowing that it's temporary as well).

- Documentation and storage of the credentials in a secure way that allows fellow team members access to the account.

- If exploitation involves changing a legitimate user's credentials, the penetration testing team leader must be made aware, in order to coordinate with the client's IT staff to minimize disruption.

- Defining a portion of the "cleanup" phase that involves attempts to delete the accounts created by team members during the engagement.

- It's not always technically feasible to delete your own accounts, so the remaining ones must be provided to the client's IT staff for immediate deletion.

Similar standard operating procedures should be developed for defining acceptable tools and techniques for use in different types of engagements and network environments. As a simple example, unencrypted C2, such as the netcat shells identified as problematic in most training and learning material in the author's previous year's presentation, should not be allowed across the public Internet, or other situations in which it cannot be wrapped in a more secure protocol or more private networking environment. The purpose of these standard operating procedures is not to "tie the hands" of testers. Deviation from the safety procedure might be determined an acceptable risk in pursuit of the goal of identifying serious vulnerabilities, but that deviation should only come after oversight, discussion, and minimization of the risks.

Standard operating procedures should be developed to ensure that safety-critical processes are being followed consistently by all team members on every engagement. These defined procedures should have a mechanism for having peers and team leaders review situationally-appropriate exceptions.

CONTINUOUS IMPROVEMENT AND SELF TESTING

After the report's delivery, penetration testing teams should meet in order to discuss the engagement. This meeting, in the author's experience, typically involves:

- Briefing the team leader (where needed) on findings--vulnerabilities and recommendations--in preparation for meetings with the client

- Postmortem on any technical issues with equipment and software, with assignments to team members for research and development needed to improve reliability

- Discussion of vulnerabilities identified that have not been seen on previous engagements

- Discussion of process: Where could the engagement have been conducted more effectively, more quickly, or with a reduction in duplicated effort? Can a software tool or a documented procedure help on future engagements?

In addition to the points above, and others designed to improve efficiency, a team should make a concerted effort to discuss the security of their testing environment and processes. This may include:

- A brief discussion by the team member(s) that reviewed the team's logs for signs of intrusion
- A review of "exceptions" made to standard operating procedures during the engagement. Where were insecure tools and practices required to be used, with exceptional care, without a more secure alternative? Would efficiency and security be better served as a result of some research and development on new tools?
- A discussion of potential vulnerabilities in procedure or tools that should be the subject of vulnerability analysis in the team's lab environment.

Self-testing may be carried out by penetration testing team members by replicating real-world penetration testing situations in their controlled lab environment, and attempting to impact the confidentiality, integrity, and availability of their own tools and processes. Qualified penetration testers, especially those with application security and development experience, are well-equipped to take an adversarial look at the third-party tools that they are using. For internally-developed tools and practices, peer testing should be used within the firm to gain the benefit of "fresh eyes".

Engaging team members on the development of secure best practices will help them retain ownership of their tools, tactics, and procedures. Continuous improvement and self-testing are necessary to keep "ahead of the curve" in potential risks posed by testing.

CHALLENGES
THE NATURE OF EXPLOITATION

The exploitation of vulnerabilities is often a "destructive" process. Bratus describes exploits, in the context of the theory of computation, as additional, unintended, and often unrestricted functionality within a system that was not part of the original developer's intentions [7]. There is nothing in the design of systems and software that guarantees that an exploit and its payload will, in the process of exercising a weakness in the code, leave the system stable or otherwise secure. More frequently, the exact opposite is true.

It will not always be possible to ensure that the exploitation process, as carried out by a penetration tester's tool, is safe in the potential presence of third-party attackers. While the establishment and maintenance of C2 post-exploitation has more flexibility, establishing control of the target program in the first place provides the penetration tester with fewer options. In most situations, a third-party attacker that can observe, on the network, the compromise of a client organization system by a penetration tester, then simply replicate the attack for their own gain.

This problem does not mean that the situation cannot be improved by changing penetration testing processes. By addressing vulnerabilities in penetration testing tools, C2 channels, and implementing best practices for

host and data security, the attack surface presented by an ongoing test can be reduced. If the only remaining way for a third party attacker to compromise the client is through a vulnerability that already existed on the system prior to the test, and identified during the test, that is a significant improvement. Careful monitoring of the network may reveal the success of such an attack, and post-engagement remediation can remove the vulnerability, after-the-fact.

The nature of exploitation, however, will always result in the development of testing tools and practices that are a challenge to deploy in a secure way.

DENSITY OF VULNERABILITIES

Schneier [8] and Geer [9] have explored the question of whether vulnerabilities in software are dense or sparse. In other words: do concerted efforts to find vulnerabilities in software find enough vulnerabilities to make a "dent" in the overall number of those vulnerabilities? This is a difficult--and largely insufficiently answered--question in the context of all computer software, and has a direct impact on the effectiveness of hunting for vulnerabilities in the tools used by penetration testers.

Can a determined effort by your team (or even a larger community of penetration testers/vulnerability analysts) identify and remediate enough weaknesses to make a difference? For procedural issues and problems involving communications security (lack of encryption, authentication), it is intuitive that the attention will be rewarded. The question becomes more complex when we talk about the "devil in the details", subtle errors in programming that give rise to exploitable vulnerabilities. Of course, this is the same problem being encountered head-on in mainstream software vulnerability.

It is this author's assessment that enough vulnerabilities can be identified and patched (or otherwise worked around or defended in layers) in penetration testing tools that the cost of finding new and reasonably exploitable bugs may rise above the potential attacker's reasonable budget. Essentially, while vulnerabilities might still be present, the risk can be reduced by "raising the bar". Close examination of many tools may result in the abandonment of those tools, in favor of those developed from the ground up on more secure principles (such as complex parsers written in low-level languages with little thought put into the integrity of data structures in memory, which may be better off re-implemented in higher-level interpreted languages).

AGILITY

Penetration testers are loath to change anything that might slow down or restrict their ability to do whatever it takes to identify and exploit vulnerabilities in their target networks. Attempts at creating a more secure penetration testing process must not degrade the value of creatively identifying vulnerabilities in the test.

The goal of implementing these recommendations is to build a penetration testing process in which the testers have a good understanding, before an engagement begins, of tools and procedures that are trustworthy. With

good standard operating procedures, a test operates at an acceptable pace, though clearly slower than one in which these considerations are not being made. Ultimately, the cost of exposing a system to third-party attack, or leaving a system in an insecure state must be seen as higher than the benefits of a more rapid test.

The purpose of the recommendations made in this work is not to restrict the creative and unique actions of the testers. Rather, the goal is to define a set of criteria for identifying how to take a "new" or "different" action while incurring a minimal amount of risk. A common language and set of goals for discussion and evaluation is necessary.

STANDARDS

Recommendations, such as these and those that will undoubtedly evolve from them, would be useful as elements of a larger standard for penetration testing practices. The state of the art of penetration testing standards is not currently well-defined, but existing standards and guidelines at least serve as a baseline for comparing services and setting expectations. The author of this paper strongly prefers using the Penetration Testing Execution Standard as a baseline for describing penetration testing services [6].

CONCLUSIONS

This paper, and its associated presentation, reviews previous work that establishes the need for better tools, practices, and training that address the need to conduct offense-oriented services, such as penetration tests, in a more secure fashion. A set of recommendations in nine categories are presented that attempt to define a set of actions and process changes that can be made by penetration testing teams to identify and reduce vulnerabilities that may arise as a result of their testing. While some challenges have been specifically identified in this work, the field of penetration testing can apply these recommendations to raise expectations about the maturity and security of services provided.

The author would like to acknowledge the audiences of talks he has given over the past few years on this topic for asking hard questions about how the problems being pointed out are meant to be fixed. The author would also like to acknowledge the penetration testing specialists that work on the teams he oversees in his role as Director of Cyber Operations at HORNE Cyber. Each team member has put significant effort into adopting new techniques and continuously improving the security and maturity of engagements they work on.

ABOUT THE AUTHOR

Wesley McGrew oversees and participates in penetration testing in his role of Director of Cyber Operations for HORNE Cyber Solutions. He has presented on topics of penetration testing, vulnerabilities, and malware analysis at DEF CON and Black Hat USA. He teaches a self-designed course on reverse engineering to students at Mississippi State University, using real-world, high-profile malware samples. Wesley graduated from Mississippi State University's Department of Computer Science and Engineering and previously worked at the Distributed Analytics and Security Institute. He holds a Ph.D. in computer science for his research in vulnerability analysis of SCADA HMI systems.

REFERENCES

1. McGrew, Wesley Analyzing and Counter-Attacking Attacker-Implanted Devices: Case Study: Pwn Plug, DEF CON 21, 2013, https://www.defcon.org/images/defcon-21/dc-21-presentations/McGrew/DEFCON-21-McGrew-Pwn-The-Pwn-Plug-WP.pdf

2. McGrew, Wesley I Hunt Penetration Testers: More Weaknesses in Tools and Procedures, DEF CON 23, 2015, https://media.defcon.org/DEF%20CON%2023/DEF%20CON%2023%20presentations/DEFCON-23-Wesley-McGrew-I-Hunt-Penetration-Testers-WP.pdf

3. McGrew, Wesley Secure Penetration Testing Operations: Demonstrated Weaknesses in Learning Material and Tools, DEF CON 24, 2016, http://hornecyber.com/media/1873/securepenetrationtestingoperationswp_final.pdf

4. Steven, Justin, 2017 Metasploit Meterpreter Dir Traversal Bugs, Advisory: https://github.com/justinsteven/advisories/blob/master/2017_metasploit_meterpreter_dir_traversal_bugs.md , also described in CVE entries: CVE-2017-5228, CVE-2017-5231, CVE-2017-5229

5. Steven, Justin, 2017 Metasploit RCE Static Key Deserialization, Advisory: https://github.com/justinsteven/advisories/blob/master/2016_metasploit_rce_static_key_deserialization.md , also described in CVE entries: CVE-2016-1000243, CVE-2016-1000244

6. Penetration Testing Execution Standard, http://www.pentest-standard.org/index.php/Main_Page

7. Bratus, Sergey, et al., Exploit Programming: From Buffer Overflows to "Weird Machines" and Theory of Computation, ;login: - The USENIX Magazine, December 2011, http://langsec.org/papers/Bratus.pdf

8. Schneier, Bruce, Should U.S. Hackers Fix Cybersecurity Holes or Exploit Them?, The Atlantic, May 19, 2014, https://www.theatlantic.com/technology/archive/2014/05/should-hackers-fix-cybersecurity-holes-or-exploit-them/371197/

9. Geer, Dan, For Good Measure: The Undiscovered, ;login: The USENIX Magazine, April 2015, http://geer.tinho.net/fgm/fgm.geer.1504.pdf

10. Samurai CTF, Reddit AMA, August 11, 2012, https://www.reddit.com/r/netsec/comments/y0nnu/we_are_samurai_ctf_and_we_won_defcon_ctf_this/c5reegh/

11. CVE listings associated with Wireshark (420 listed, as of February 14, 2017) http://www.cvedetails.com/vulnerability-list/vendor_id-4861/product_id-8292/Wireshark-Wireshark.html

12. Debian Security Advisory – DSA-3775-1 tcpdump – security update, January 29, 2017, https://www.debian.org/security/2017/dsa-3775

13. Carpenter, Brian (@geeknik), CVE fuzzing poc, https://github.com/geeknik/cve-fuzzing-poc