Software Engineering Institute | Carnegie Mellon University

Penetration Testing Tools

Ken van Wyk

January 2007

ABSTRACT: This article provides a primer on the most commonly used tools for traditional penetration testing. (A related article provides an overview of penetration testing practices.) Although some tools are listed by name, these are merely intended to serve as examples of particular types of tools. The list is in no way intended to be comprehensive and should not be interpreted as an endorsement of the tools listed.

That said, we start by looking at the most common tool types, port scanners and vulnerability scanners. Examples in the open source and commercial communities are provided for each, where appropriate.

Next, we delve into the state of the commercial practice with regards to tool usage and how penetration testing services are provided. We then make a series of recommendations for selecting the right toolkit for the job and for training one's testers in penetration testing and the tools used.

PORT SCANNERS

Port scanning tools are used to gather information about a test target from a remote network location. Specifically, port scanners attempt to locate which network services are available for connection on each target host. They do this by probing each of the designated (or default) network ports or services on the target system.

Most port scanners are able to scan both TCP as well as UDP ports. Most can also target a specified list of ports and can be configured for the speed and port sequence that they scan. Additionally, most port scanners are able to perform a range of different varieties of port probes. These can include a standard SYN-->SYN-->ACK-->ACK sequence for TCP ports, as well as "half scans". Lastly, another common feature of port scanners is their ability to deduce the operating system type—and often times the version number—based on watching the empirical behavior that it exhibits when probed with variations of TCP flag settings. They can do this because many TCP/IP implementations vary in their specific responses to probes that aren't explicitly addressed by Internet convention.

The rationale for all of the configuration flexibility with port scanners is so that the tester can employ a great deal of agility in testing for different port configu-

Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-2612

Phone: 412-268-5800
Toll-free: 1-888-201-4479

www.sei.cmu.edu

rations, as well as to attempt to hide from network intrusion detection mechanisms and the like. Although this can be particularly useful for testing production or near-production network environments, its usefulness is at best diminished for the purposes laid out here.

Perhaps the best known and most popular of port scanners is Nmap, available for free from http://insecure.org/nmap/.

VULNERABILITY SCANNERS

The primary distinction between a port scanner and a network-based vulnerability scanner is that vulnerability scanners attempt to exercise (known) vulnerabilities on their targeted systems, whereas port scanners only produce an inventory of available services. That said, the distinguishing factors between port and vulnerability scanners are often times blurred.

Apart from that, a good vulnerability scanner is a vital tool to a traditional penetration tester. They provide an essential means of meticulously probing each and every available network service on the targeted hosts. Vulnerability scanners work from a database of documented network service security defects, exercising each defect on each available service of the target range of hosts.

This enables the tester to rapidly and quite exhaustively look for common configuration weaknesses in the targeted systems as well as for unpatched network server software.

Traditional vulnerability scanners are generally able to scan only target operating systems and network infrastructure components, as well as any other TCP/IP device on a network, for operating system level weaknesses. They are not able to probe general purpose applications, as they lack any sort of knowledge base of how an unknown application functions.

Some vulnerability scanners are able to attempt to exploit network trust relationships by recursively scanning the targeted network on each compromisable host. This capability is particularly useful to a CIO audience, as it enables the test team to demonstrate how an attacker might be able to enter a corporate network by taking iterative steps towards a target. Again, however, it is of little relevance to the sorts of penetration testing that matter the most in a software development context, except (arguably) to demonstrate to the development team how a single weakness might lead to greater compromises if exploited.

Host-based vulnerability scanners are also readily available, both commercially as well as within the open source community. They scan a host operating system for known weaknesses and unpatched software, as well as for such configuration problems as file access control and user permission management defects. Although they do not analyze application software directly, they are useful at finding mistakes made in access control, configuration management, and other configuration attributes, even at an application layer. Therefore, they are useful aids in a development driven penetration test, if only to spot human errors in configurations.

Although both host- and network-based vulnerability scanners do little to help an application-level penetration test, they are necessary fundamental tools for penetration testers. A popular free (but not open source) vulnerability scanner is Nessus. Good examples of commercial vulnerability scanners include Core Impact, Qualys's QualysGuard family, and ISS's Internet Scanner.

APPLICATION SCANNERS

Taking the concept of a network-based vulnerability scanner one step further, application scanners began appearing several years ago. These attempt to do probing of general purpose web-based applications by attempting a variety of common and known attacks on each targeted application and page of each application.

Most application scanners can observe the normative functional behavior of an application and then attempt a sequence of common attacks against the application. The attacks include buffer overruns, cookie manipulation, SQL insertion, cross-site scripting (also referred to as "XSS"), and the like.

Although this feature set sounds as though it might be of significant value to a test team that is evaluating a web-based application, the chief shortcoming of the technology is that the tools only test for a relatively small and simplistic set of attack profiles—for example, putting a few hundred "A" characters into a string variable to look for a buffer overrun situation. Further, since the testing is still performed in an entirely black box manner, the utility of such tools is greatly diminished to any serious testing process.

That is, although failing any of the tests is demonstrably a bad situation, passing all of the tests can only provide, at best, a misplaced sense of security.

Popular commercial application scanners include Watchfire's Appscan and SPI Dynamics's WebInspect.

WEB APPLICATION ASSESSMENT PROXY

Although they only work on web applications, web application assessment proxies are perhaps the most useful of the vulnerability assessment tools listed here. Assessment proxies work by interposing themselves between the tester's web browser and the target web server. Further, they allow the tester to view and manipulate any and all data content flowing between the two. This gives the tester a great deal of flexibility in trying different "tricks" to exercise application weaknesses in the application's user interface and associated components. This level of flexibility is why assessment proxies are considered essential tools for all black box testing of web applications.

For example, the tester can view all cookies, hidden HTML fields, and other data in use by a web application and attempt to manipulate their values to trick the application into allowing access where the tester should not be able to get to. Changing cookie values such as "customerID" can have startling results on poorly developed applications.

Popular web application proxy tools include Paros Proxy and OWASP's WebScarab, available from http://www.parosproxy.org/ and http://www.owasp.org/, respectively.

STATE OF THE COMMERCIAL PRACTICE

Few penetration testing organizations today do white box penetration testing like that described here. Most simply do the sort of black box testing described above. Some organizations offer hybrid approaches in which they do traditional penetration testing along with some level of source code analysis of the application's code base. As the software development community continues to awaken to rigorous software security practices, there's little doubt that the penetration testing community will follow suit. For the time being, though, this appears to be the status quo.

In the area of testing tools, most commercial penetration testing organizations adopt a very pragmatic view, using a hodgepodge assortment of commercial, open source, and home grown software tools. As pointed out above, this approach has eventually led to a push for standardization of reporting formats, with XML data encoding becoming increasingly pervasive. The need for maintaining a wide selection of tools was driven by two principal forces: testing engineers who demand maximum flexibility and customer organizations who want to leverage their own tool licenses and/or specify what tools may be used on their infrastructures.

The primary purposes for penetration testing continue to be largely twofold: detection of unpatched or improperly configured systems in existing network infrastructures, or testing new application environments prior to putting them into production.

Another relative newcomer in penetration testing, as mentioned above, has been the so-called “fire and forget” sorts of services. These services aim to conduct fully automated scanning of entire networks, looking primarily for changes that would indicate new systems being installed or existing systems being unpatched. As one might expect, these services are of little use to software developers but can be highly useful to a CIO audience.

SELECTING THE RIGHT TOOLKIT

Since, as we’ve seen here, most of the tools available today are primarily meant for traditional penetration testing practices, it is vital that software development penetration testers choose their toolkits wisely. Many of the features that would appeal the most to a traditional tester are likely to be nearly or utterly useless in a software development context. For example, network vulnerability scanners that try to evade detection by IDS and IPS devices would normally not be useful for software development.

The following list of features should be the sorts of things that a software development penetration testing team would find most useful among the available tools:

- Visibility. Perhaps the most important feature for any penetration testing tool is that the tests that it does and the reporting that it provides need to be entirely visible to the test team. Specifically avoid commercial tools that veil their probing as proprietary or otherwise not viewable to the test team.
- Extensibility. Testing application software requires tools that can be highly customized. Look specifically for scripting languages or plug-in capabilities and the like that can be used to construct customized probes.
- Configurability. Most of the testing tools are highly configurable in terms of which tests they do. This is vital to the test team.
- Documentation. To maximize the usefulness of a testing tool for software developers, the probes that it performs should be well documented and explained.
- License flexibility. Some commercial tools “lock” the tool so that it can only test a particular IP number or range. They do this for licensing purposes, as well as to protect the user from inadvertently scanning a network without authorization. In most cases, this sort of locking does nothing of value to a

software development test team. Avoid this “feature” if at all possible.

With these things in mind, the test team should be able to assemble a toolkit that works best in their context.

THE NECESSITY OF TRAINING PERSONNEL

As a final note regarding penetration testing tools, it is vital that the test team has a thorough understanding of the capabilities in each of the selected tools. If vendor training on the tools is available, then it should be carefully considered. If none is available, then the test team should be strongly encouraged to thoroughly and rigorously practice using the tool in a safe, isolated network environment. Particular focus should be paid to learning every feature and capability of each tool. In cases where a tool has a scripting language or other plug-in mechanism, these features too should be thoroughly explored.

Additionally, penetration testing processes tend to be rather knowledge intensive in a way that extends well beyond even a thorough understanding of the tools being used. The best penetration testers have extensive experience in their craft; they intimately understand each and every attack used by their automated testing tools; they intuitively and explicitly know what to look for when assessing the results of their tests; they understand how complex software systems work.

There exists a multitude of training programs and so-called “boot camp” training sessions that immerse the students in the various tools and how they can best be used. The best ones include a significant amount of hands-on labs so students can put to practice the concepts presented in the lecture sessions.

Although these training programs can be an excellent source of training to get training personnel a quick exposure to penetration testing, they are still not the same as “time in cockpit” experience. It remains necessary for testers to gain experience in a well focused and controlled manner. For this, there is no substitute for a positive mentoring environment that pairs up junior- and senior-level personnel.

The reason for this is that the testers need to thoroughly understand not just the mechanical workings of the tools but the rationale and judgment decisions that go into using them effectively. Even the best training programs are pressed to teach rationale and judgment effectively.

For this reason, a combination of training and mentoring through experience is generally considered to be the best means of building a penetration testing team.