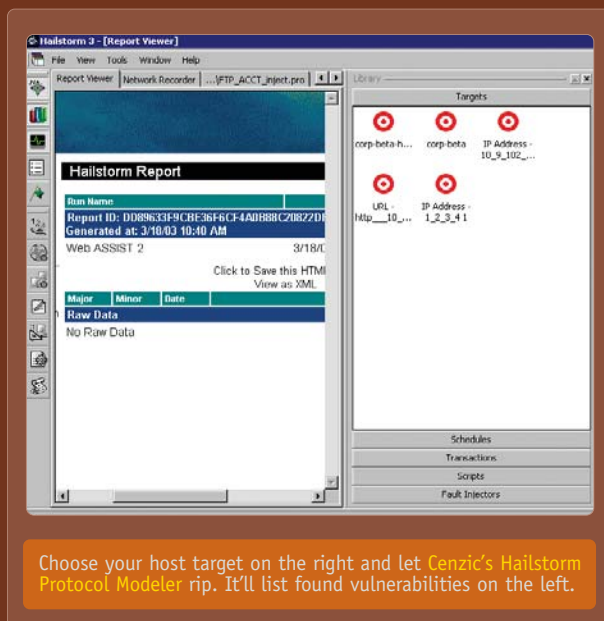




ARMING YOUR TOP SECURITY GUNS

By Patrick Mueller

Cenzic's Hailstorm Protocol Modeler isn't for everyone, but if you can swing the cost and know an RFC violation from a SQL disclosure vulnerability, it's the tool for you



Ever had a theory on how to break a network-enabled application but never quite got around to testing it? Maybe you didn't feel like digging out that book on socket calls and coding something from scratch in C. Perhaps you got halfway there when you found some obscure limitation in the PERL::Net libraries. Even if you did find the perfect open source packet-generation toolkit, your project may have been interrupted by real work—completing host-level security audits of your company's public Web servers, for instance.

If you've found yourself in one of these situations—and using raw tools to generate network security testing traffic seems perfectly normal to you—there's a good chance you could have cranked out your testing tool quickly using Hailstorm Protocol Modeler, the flagship product from Cenzic, a company co-founded by famous hacker and security expert Greg Hoglund. (We use the term *hacker* here in the proper sense: an extremely clever programmer.)

Cenzic's Protocol Modeler helps uncover bugs in network-enabled software, whether that software is running on a popular computing platform, like Intel or Sun Solaris, or on a specialized hardware platform, such as a router, load-balancer or VPN concentrator. Any

device or system with a listening IP is a potential target, just like in the real world.

Protocol Modeler lets you quickly develop network-layer attacks by using fault-injection components. The attacks are aimed at uncovering vulnerabilities in underlying software, whether it be closed-source “blackbox” testing of a commercial product—say your perimeter firewall’s IP-based administrative interfaces—or testing of an in-house Web application under development. Protocol Modeler is capable of virtually any

Executive Summary

Cenzic Hailstorm Protocol Modeler

Clearly, Cenzic Hailstorm Protocol Modeler’s \$25,000 price will put it out of reach of most small to midsize organizations. But if you have the cash, gather potential users, show them this article, and observe their reactions. If they gush about specific tests they’d love to run against an application and marvel at how much time such a tool could save them, ask for a demo. If their eyes glaze over or they want to know how many step-by-step wizards come bundled, keep moving.

Say you decide you want to buy. Making a case to the business office could be tricky. Driving the tool will require a high-level analyst with a deep understanding of networking protocols, attack methodologies and secure programming practices. This security ninja will have the chance to execute these precise and deadly moves only if tasked with application and/or network-device testing. In many organizations, such testing is left to third-party security consulting companies specializing in application blackboxing, code auditing and network-device fault testing. However, if you have the time, the mandate and the skills to do this type of work in-house, Protocol Modeler will save you money quickly compared with the cost of contracting security houses each time you need a test performed.

type of test because, even if the prebuilt fault injectors don’t address your testing needs, it’s possible to create your own fault injectors—albeit with much work.

You can manipulate simpler types of network traffic using the GUI tools and undertake more complex tasks using a PERL-like API, though once you head down that road, you’ve lost most of the speed and elegance of test creation that make Protocol Modeler attractive.

You Go Test

When we first fired up Protocol Modeler in our Chicago Neohapsis partner labs, we used the product’s wizards to walk us through common security-testing activities. One wizard crawls a Web site looking for SQL-disclosure vulnerabilities, cross-site scripting bugs and opportunities for successful command-injection attacks. Unfortunately, even on the small test Web site we pointed this script at—a beta version of our corporate intranet server—Protocol Modeler crashed. Cenzic blamed the crash on bugs that had infiltrated the latest released version of the code (more on this later). A smaller site with less user interaction proved a more digestible target.

Another wizard tests firewalls. Basically, it turns Protocol Modeler into a glorified port scanner—a rather uninteresting use for such a flexible tool.

The wizards quickly show their limitations, though they do serve the highly useful purpose of introducing the

REPORT CARD

	Cenzic Hailstorm Protocol Modeler
DOCUMENTATION (20%)	3
PERFORMANCE/STABILITY (20%)	2
PRICE (20%)	2
PROTOCOL BREADTH SUPPORT (APPLICATIONS) (15%)	4
REPORTING (15%)	3
INSTALLATION AND CONFIGURATION (10%)	5
TOTAL SCORE (100%)	2.95
A≥4.3, B≥3.5, C≥2.5, D≥1.5, F<1.5 A-C GRADES INCLUDE + OR - IN THEIR RANGES. TOTAL SCORES AND WEIGHTED SCORES ARE BASED ON A SCALE OF 0-5.	
C	

Customize the results of this report card using the Interactive Report Card®, a Java applet at www.nwc.com.

user to the application's components. Without this sort of guide, you'd be left to muddle through the complex interface, a frustrating and potentially fruitless experience.

Protocol Modeler's Network Recorder provides a more focused testing method. Hit the record button, then prompt the application under test to execute the net-

For Web applications, Protocol Modeler investigates HTTP-based vulnerabilities.

work transaction you'd like to investigate. It can be a well-known standards-based application, such as SMTP or IMAP, in which case the packets will be decoded into proper fields. Or, it can be a proprietary application, where the captured traffic will not have an automatically overlaid format. In this case, the user must scroll through the transactions and attempt to decipher what's happening. Finally, it could be a Web application, in which case Protocol Modeler provides additional support for investigating HTTP-based vulnerabilities as well as a browser interface for walking through an application

and gathering data for the recorder. Once data is captured, it can be viewed, manipulated, loaded up with fault injectors and played back to test for vulnerabilities.

One of Protocol Modeler's most powerful features is its variety of fault injectors, which can be dragged into the desired components of a transaction and are critical to the program's process. For example, in a Web application providing user input that appears to be fed to a back-end database query, attempting a SQL Disclosure fault injector would be a good choice. After you hit play, the transaction plays back repeatedly using different permutations in an attempt to fool the Web server into passing SQL commands to the back-end database server directly. Protocol Modeler watches for the application to "fail" in this manner. Continuing with this example, if it sees an apparent SQL error coming back on the HTTP return, Protocol Modeler notes this and reports it to the user.

Protocol Modeler uses several strategies to monitor for faults and signs of vulnerabilities. One, previously noted, is to watch for typical return values of a fault; another compares normal return data, as from a normal-size input, to deviant return data, as from a successful buffer overflow applied to the same input.

The fault injectors can be applied in two ways. You

How We Tested Protocol Modeler

The requirements are pretty basic: a beefy, but not killer, Windows machine with at least one NIC. We recommend using standard, well-supported hardware (including that NIC) in case you run into problems, as we did. This lets you rule out hardware problems quickly.

You'll need to create a segmented test network, if you don't already have one, before you let Protocol Modeler hurl packets. Don't even think about deploying this product on a production network. Critical applications crash at the mere men-

tion of the program being connected to their segments. Not only are most of Protocol Modeler's tests designed to produce fatal run-time faults in OSs, network devices and applications, but sometimes a test will do something slightly different from what the operator intended.

Fortunately, at our Neohapsis labs we have more test networks than production networks. We plugged the Protocol Modeler host (housed on a Compaq 1850R Pentium III machine) into a Cisco Systems 2900 series switch. Several Linux and Microsoft Windows 2000 hosts, all running var-

ious services including POP, SMTP, DNS and HTTP, provided target applications. One advantage of running on a full-featured managed switch is the option of configuring a span port to monitor traffic. The other option is to connect Protocol Modeler to a hub.

Either way, you'll want to hook up your favorite sniffer—we used Ethereal and its CLI-based sibling, Tethereal—for out-of-band monitoring of Protocol Modeler. The runtime logging is insufficient not only for monitoring progress but also for gaining an understanding of the transactions being run.

can drop one directly into a job, and Protocol Modeler will calculate all the appropriate places to apply it automatically. This blanket method often results in extremely long test times, depending on the fault injectors used. Alternatively, fault injectors can be applied to specific parts of a transaction, for example, an HTTP *post* field.

Other examples of fault injectors are Bitwalk, which can be applied to any binary payload by applying a large range of values to each byte within the field; Buffer Overflow, which applies incrementally larger values to target fields and monitors for signs of a successful buffer overflow; and Cross-Site Scripting, which attempts to post executable content throughout the site and then goes back to check for its availability to the user (see features chart, page 108, for additional details).

Busy Windows

The Protocol Modeler GUI is decent, though a few quirks prevent it from presenting a highly polished front end. All component windows are contained

within the master window and can be moved around and docked and undocked, and many can be closed and later reopened. You'll want as large a monitor as possible because of the myriad subwindows and data sprawling across the screen. The transaction-editor pane hasn't been updated since we last examined the tool (more than a year ago) and still presents a clunky interface that requires the user to scroll horizontally for miles while searching for the desired field. The company promises better integration with the transaction editor in future versions. Even with a large display, working in the GUI can be a bit cramped because some of the subwindows cannot be closed, and resizing widgets can be tricky.

The fact that this hard-core network-vulnerability testing tool is housed on a Microsoft Windows platform may strike some as a bit strange. After all, cobbling together network vulnerability testing tools offering the same functionality typically means spending time with custom packet-generation libraries and a

Where Does Protocol Modeler Fit?

That you've never heard of a tool quite like (or maybe anything like) Hailstorm Protocol Modeler is a testament to its uniqueness. From a business standpoint, that's both a strength and a weakness. On one hand, Cenxic has no commercial competition to contend with. But existing in a vacuum makes the product somewhat of a black sheep. The security tools market is already crowded, creating some stiff competition for security analysts' mind share. To help you put Protocol Modeler in perspective, here's a look at some security tools with relevant similarities. For more on Cenxic finding its target customer, see www.nwc.com/1408/1408rd1.html.

» **Vulnerability-Assessment Scanners:** VA scanners run through databases of known attack types, probing a host or network device for known security vulnerabilities. When a new vulnerability is discovered—for example, Microsoft IIS is found to be susceptible to some new type of script disclosure—a check is developed and added to the list of signatures. Each is run in turn on the targeted host, perhaps after being narrowed down by host-type identification, “only run the IIS checks on IIS servers, not on Apache.”

Like most signature-driven anti-virus software, with VA you are protected only against known attacks. Protocol Modeler doesn't offer conventional VA services but rather enables the user to proactively probe an application for unknown but suspected security vulnerabilities. This bug hunting is more time-consuming and technically demanding than running a VA scanner. Because these activities have different goals—identifying known vulnerabilities versus finding undiscovered flaws—comparing them is useful only to distinguish the two types of tools.

Examples of VA tools include Internet Security Systems' Internet Scanner and Nessus.

» **Web Application Scanners:** Web application scanners have properties of conventional VA scanners as well as similarities to the Protocol Modeler testing platform. Generally using a proxy-based architecture, they can crawl automatically or be driven by a user “test case” through a Web application. By watching for typical insecure Web programming practices and running checks against suspect components, the tools can discover and evaluate these vulnerabilities. Application

tampering, including malicious cookie tampering and hidden value attacks, can be detected and identified by these tools. Protocol Modeler can pinpoint some of these Web application bugs, though the feature sets and strengths relative to Protocol Modeler will depend on the product in question.

» **Toolkits and Libraries:** The final product space is largely noncommercial. Loads of open-source programming libraries, APIs and toolkits are available to help automate the process of network vulnerability testing. Using them will require deep technical knowledge of the protocols involved as well as programming and Unix expertise. These are not for the faint of heart. Fuzzers—tools that feed pathologically formed input to a program (in this context, a network-enabled application) to produce a fault—fall in this category. Security consultancy @Stake produces one such tool, called Spike. Protocol Modeler can undertake many of these tasks. In fact, the real value proposition of this product is that it provides a faster and more effective way to create many tests that would otherwise have to be assembled using these different network security toolkits.

compiler on your preferred BSD or Linux platform. But it turns out that Windows is a logical choice for this tool. In terms of ease of use and familiarity, Windows GUI widgets create a familiar home for most users. Some religiously Unix-oriented network-security geeks may bristle at anything coming out of Redmond, but you can't please everyone all the time. Any NIC supported under Windows will work fine.

As for reporting, by storing the data in a SQL-based repository of the user's choice, custom reports based on specific requirements can be created outside Protocol Modeler. The built-in reports are sufficient for most testing and even include (where appropriate) rollup graphs and narrative explanations.

Crash-Test Dummies

Unfortunately, we faced some insurmountable glitches with version 3.06 (shipped to us for testing) that were especially painful given Protocol Modeler's hefty \$25,000 price. The QA cycle that let this version out the door leaves us less than impressed. According to Cenzic, the changes made in 3.06 were all performance related, but the developers seem to have outsmarted themselves: Protocol Modeler frequently ran out of file handles and crashed spectacularly. Any intensive test fell prey to these faults. The company issued patches and the engineering team scrambled, but Cenzic could not fix the problems during our testing window.

The Web-crawling wizard experience detailed earlier left us with Microsoft Visual C++ exception windows covering the screen and a crashed version of Protocol Modeler. We encountered similar disappointments on other long-running tests. For example, we tried running a SQL Disclosure attack on a single user-supplied input on a Web page being posted back to a Web server.

Features

	Cenzic Hailstorm Protocol Modeler
OSI Layers supported	2-7
Basic Layer 7 support (baselines)	FTP, HTTP, ICMP, IIS ISAPI, IMAP, POP3, SMTP
Extended Layer 7 support	HTTP
Fault injection Multilayer	BitWalk, buffer overflow, format string, relative path, Windows command injection, Unix command injection, Generic SQL, Microsoft SQL, MySQL, Postgres, SQL invalidation
HTTP Layer 3-4	Cross site, delimiter IP fragmentation, TCP segmentation
Platform	Microsoft Windows 2000 (Pro or Server), Windows XP
Wizards	Web application, network device, IDS signature, firewall test, unauthorized access audit, orphan audit, login-policy test, password-policy test
Price	\$25,000

WebLinks

» "Proxies Add a Protective Shield" (NETWORK COMPUTING, March 5, 2003) www.nwc.com/1404/1404f4.html

» "Secure to the Core" (NETWORK COMPUTING, Jan. 23, 2003) www.nwc.com/1401/1401f1.html

» "Tipping the Scales" (NETWORK COMPUTING, Sept. 30, 2002) www.nwc.com/1320/1320sp1.html

We watched the logs on the HTTP server (the user will often find him or herself monitoring closer to the target application) and got a glimpse of the types of attempted queries. We observed 8,600 attack queries before Protocol Modeler finally crashed (Cenzic said the fault injector was probably close to finishing, given that number). Unfortunately the product doesn't do any checkpointing, so we couldn't find out if any vulnerabilities were discovered in those queries.

A more critical feature gap is the lack of any indication of the approximate and relative run-times of the fault injectors, some of which can run for hours or days, depending on the size of the test. Even nicer would be a: "This test requires about 7 minutes per iteration, times 60 loops = 7 hours" message.

Art Meets Science

The Protocol Modeler experience is, in many ways, like staring at a blank 6-foot canvas with a full palette in your hands. Using the tool successfully takes creativity. We don't claim to be the Picasso of the Protocol Modeler world—frankly, much of our work was painting by numbers with the wizards, though we did begin to devise some interesting tests as we became more comfortable with the Protocol Modeler environment. Make no mistake: The product is difficult to use. Allow at least a full week to ramp up on the tool and assemble a preliminary test network. This assumes you have advanced knowledge of both IP networking protocols (at every level) and advanced knowledge of security vulnerability theory and practice. If not, allow much more time. According to Cenzic, a few days of on-site training come with the purchase price. We recommend taking full advantage of this.

At press time Cenzic had launched a new product, Hailstorm Web. The core engine technology is the same as Hailstorm Protocol Modeler but also includes extensive workflow management for structured use within an organization. Security expert and QA analyst roles let application security testing tasks be distributed logically. Predictably, the focus is on HTTP-based applications. [NWC](#)

Hailstorm Protocol Modeler 3.06, \$25,000. Cenzic, (408) 626-9004. www.cenzic.com

PATRICK MUELLER is a senior security analyst for Chicago-based security consultancy Neohapsis. Write to him at pmueller@neohapsis.com. Post a comment or question on this story at www.nwc.com/go/ask.html.