



**Secure64 DNS Software
Performance and Attack Resistance Test Results
March 7, 2007**

Executive Summary

We undertook an assignment surrounding a series of two tests, one measuring performance, and an attack test/identity regimen, both poised against two applications on an HP server supplied by Secure64 Software Corporation. The tests were performed between January 27th, 2007 and March 7th, 2007. The HP server (HP rx2660 server) served as the host for an operating system and application based on Secure64's DNS (Domain Name System) software and hardware and alternately as a host for the Linux operating system and the 'BIND' DNS application. The results, when comparing the Secure64 DNS versus the Linux/BIND combination, showed that the Secure64 performs several times faster than Linux/BIND (even when we used best practices to optimize the Linux/BIND combination), with a secure/attack-resistant profile.

The Secure64 application and BIND applications are designed to be used in networks to provide answers to DNS queries. The Secure64 application serves as an *authoritative DNS server*¹. An authoritative DNS server varies slightly in practice and use from traditional DNS servers. Results were compared with an identical test methodology applied against the same server hardware running the Linux operating system (which had been modified specifically to suit only the DNS services that the Secure64 software supports) and the BIND version 9.3.4 program.

The attack resistance test results we measured indicate that the Secure64 DNS software appears to be able to sustain a variety of common attack profiles while continuing to function. The Secure64 software was subjected to a number of attacks known to be disruptive to servers, and ignored the attacks, delivering information as requested up to the saturation point of the connections used. The Secure64 DNS software also evaded operating system detection, remaining anonymous. Additionally, we used two applications commonly used to identify/fingerprint the Secure64 DNS software, to no avail. The results are laudable.

Overall, the Secure64 software appeared to be highly reliable in terms of penetrability, ignored attacks, and an admirable performer. We found several characteristics that we feel could be improved, but these flaws were minor in nature.

¹ Authoritative DNS servers delivers the results of queries asked of it after having stored and updated DNS information. What is different is that there are no forward-referenced queries, and there is no caching of requests unlike the Linux/BIND server combination.

Part One: Query Performance Test Plan and Execution

Test Plan and Execution

We tested both the Secure64 DNS software and the Linux/BIND combination to gauge performance in terms of queries per second. We used an open source tool known as dnsp perf to query each server under identical circumstances and hardware and test conditions to gauge the sustained queries in any given second over a period of 100 seconds.

Two different databases were used: one supplied by Secure64; and another that we randomly generated ourselves. Our test results revealed that the database size and choice made no difference in outcome/results. A peak number of entries, regardless of their type, was found in the hardware combination tested, of approximately 11 million entries.

The dnsp erf app loads and runs successive queries against a DNS server. The maximum amount of successful queries was obtained using two different querying client machines, and five instances of dnsp erf against the Secure64 DNS server. We recorded a maximum 109K satisfied queries per second. Sustained rates of over 100K queries per second are possible and normal. We found that query results were largely unaffected by whether a large zone query, small zone query, large database, small database, large query or small query was used.

By contrast, the Linux/BIND combination produced at best, 29K queries per second, less than one-third of the rate of the maximum number under the same conditions. Average query results were 26K queries per second. The results of the performance testing are outstanding for Secure64's DNS server software.

Test Conditions

We used a GBE managed switch network, and mirrored traffic from the HP rx2660 server to a Fluke OptiView III protocol analyzer. We additionally substituted a Network General Sniffer initially to corroborate the Fluke analyzer's ability to capture traffic. Both devices report the same results, and we continued using the Fluke for all tests.

We populated the DNS servers database with identical configurations; the configuration did not appear to matter. Configuration A utilized an ExtremeLabs-generated random database, while Configuration B utilized a sample database provided by Secure64 Software. Results didn't vary notably between the two DNS database population sets.

We subjected the DNS servers sequentially to DNS-specific queries of various types, using first one client query instance of dnsp erf, then a second. We then added an additional client machine, then a third and fourth instance (now two per machine). A peak was achieved using two machines, one with two instances, and the other with three instances, for a total of five instances of dnsquery instances against the Secure64 DNS server, and then the Linux/BIND combination. The Linux/BIND combination was

stripped of all services, daemons, and other elements not essential to DNS services. IPTables was used on the Linux/BIND DNS server in a generic configuration that exposed only telnet, ssh, and DNS ports.

We launched dnperf using discrete instances, and ran them for 100 seconds. The best second from the 100 seconds is used to quote the peak query rate.

Part Two: Attack Resistance and Profiling Tests

Test Plan and Execution

Two varieties of tests were executed, Performance Under Attack and Identity Exposure/Fingerprinting. ExtremeLabs used a Spirent 2700 Avalanche appliance as a load generator and assault device, along with several industry known-and-accepted test and performance measurement components. The same attacks were subjected to the Secure64 software as well as the Linux/BIND combination.

The Spirent 2700 was connected to the Secure64 DNS software via a DLink managed GBE switch, and a port connection to the Secure64 DNS software was reflected to another port where a Fluke OptiView III analyzer was connected, to allow direct connectivity and trace. We attacked the software, observed its functionality, and audited the results.

Performance Under Attack

Eight different attack simulations were used to judge whether the Secure64 DNS software could both withstand the distraction of the attack as well as continue to deliver the results to service requests.

The attack test methodology builds an increasing (ramped) number of queries focused at the server's Ethernet connection ports, until the number of queries actually saturates the connection, forcing out legitimate queries from occurring altogether. Some servers, faithfully responding to attacks, tie up internal resources, and rapidly become unavailable to legitimate requests, thus denying service. This methodology through each of the attack sequences emulates 'bot' attack methods that try to achieve and hold 'illegitimate' connections, thus tying up the server from being able to perform its service function, in this case, the results of DNS queries and legitimate zone-transfer requests.

Attacks were sequentially generated by the Avalanche 2700 via modified attack scripts generated by the Spirent Avalanche Commander software. The attack tests were ARPFlood, EvasiveUDP, Ping of Death, Ping Sweep (flood), TCP SynFlood, UDPFlood, Unreachable Host Attack, and XmasTree Attack. These attack sequences are annotated in the Appendix.

During the attacks, the Unix (Darwin-BSD) dig command was used to query the Secure64 software to verify the software was still operational and showed no latency of results until the point where the attacks saturated the Ethernet connection to the Secure64 software.

We usually saw no slowdown in the software up and until the near attack saturation point of the software (the point where *no* server/software survives). The Secure64 software appeared to ignore the attacks, and delivered until the wire was full of attack queries. Illegitimate requests appeared to be totally ignored. There was no latency seen during the attacks until the connection queries nearly saturated the connection during each of the test sequences. Each test was run through three successive iterations, and the results compared.

The Linux/BIND combination fared poorly by comparison. While the Linux/BIND server did not crash, we found, the Linux/BIND combination became unavailable for longer periods of time, and more quickly than the Secure64 server. A test-by-test results listing is located in the Appendix. Additionally, the EvasiveUDP attack caused the server to begin to respond unconditionally, expanding server apparent downtime.

In summary, the Secure64 DNS software we tested showed excellent immunity to a suite of common attacks, when compared to other servers used to provide DNS services. Its availability in comparison to the Linux/BIND combination strongly favors the Secure64 in terms of availability and resistance to attack.

Identity Masking/Fingerprinting Tests

We used two commonly accepted tests, NMAP and a routine of the NESSUS package to test the Secure64 software to reaction to probing.

NMAP is a port probe used to find open computer server Ethernet ports/sockets and establish or determine OS identity through the results returned by various probing methods. By revealing either open ports, or the base operating system type/version of a server, attacks can be selected that probe and select specific tactics based on the identity of the server that's been identified. This common tactic relies on the ability to identify the host operating system and services running on the operating system so that they can be manipulated and cracked, allowing the systems to be 'taken over' and controlled for various nefarious purposes. These 'fingerprints', once identified, are subsequently used to attack a server, generally by 'bots', to take over and capture the server.

We found only two ports open on the Secure64 software, corresponding to the SSH protocol and to DNS. This is an uncommonly small attack surface. We probed these ports, and attempted to shove malformed packets, as well as deformed packets, and these actions produced no results in response from the Secure64 software.

The NMAP program was unable to identify the operating system used in the Secure64 software, but because of the relationship between the DNS software and the operating

system published by Secure64 (SourceT²), it's reasonable to guess that is the operating system running inside the Secure64 DNS software. It was possible to guess the SSH version supported by the Secure64 software, but we could not make this version react negatively or cause it to stop or cease functioning when the port was configured as a console port.

We additionally used NESSUS software to probe the DNS services within the Secure64 software, and it was unable to guess ('fingerprint') the OS running at the IP addresses associated with the Secure64 software under test.

The Linux/BIND combination was subjected to the same tests. We removed all but DNS and ssh services in the server, and used NMap and NESSUS to probe the server. NMap test results were identical to the Secure64 software. However, we were able to use NESSUS to 'guess' the identity (correctly) of the Linux/BIND combination server, despite numerous best-practices steps we'd used to conceal the identity of the Linux host. We believe that the NESSUS routine uses a combination of ssh query and the way that the network card driver reacts to 'finger' the identity of the Linux/BIND combination server.

The Secure64 DNS service measured comparatively high availability under attack stress that we placed on the unit with the Spirent Avalanche 2700 routines that we used. The server also was unable to be fingerprinted, or have its identity or its components discerned by the probing applications we used. This apparent immunity from attacks and fingerprinting is unusually good, compared to many installations we've seen and researched. Linux/BIND did not fare as well.

The Secure64 DNS software has a very low attack surface, we found, and otherwise ignores bad/malicious environmental behavior. As an authoritative DNS server, it's likely to become a successful product.

Attack and Identity Test Conditions

We used a Spirent Avalanche 2700 appliance that contains four GBE ports, connected to a crossbar-Ethernet switch to the ports of the Secure64 software to perform these tests. These tests, listed below, attack a target device under test (in our configuration, the targeted device was the Secure64 DNS software) using differing methods designed to overwhelm or otherwise tie up the ability of the target device under test to respond.

Each tests starts slowly then builds independent user (IP/MAC pairs), emulating tens, then hundreds then thousands then tens of thousands of users until the server can no longer be reached. During the test, a Unix (compiled under gcc4, running on a Macintosh Xserve server containing twin Core-Duo Intel processors running at 3.0Ghz with twin onboard-GBE connections to the switch) application known as dig, queried the Secure64

² A registered trademark; also cracks for this operating system are unknown to us, as it's a stripped operating system, and its source code isn't published.

DNS software, until dig was unsuccessful. Failures occurred and were noted at wire saturation as monitored by the Fluke OptiView III software.

The results of these tests demonstrated that the Secure64 software ignored the attacks, and continued to respond to dig requests with no latency changes until the attacks became so strong that the Secure64 DNS software could no longer be heard from, as attacks overwhelmed the circuit. When the attacks ceased, the Secure64 DNS software appeared to be in the same state as before the attacks started, and behaved as expected.

Appendix of Attack Tests and Results

Attack and identity tests were performed on a switched GBE isolated network. The Spirent Avalanche 2700 was programmed to build a profile of increasing user counts (and IP/MAC pairs where applicable) poised at the IP address of the server under test. The server's configuration was unmodified for all Secure64 DNS server tests, and was highly modified for the Linux/BIND test. We modified the Linux/BIND server, removing all non-essential services, daemons, and processes, and optimized virtualization/paging/memory management, as well as using IPTables techniques to make the server more 'stealthy'.

Performance was audited by packet capturing on a Fluke OptiView III, which concurrently ran OptiView software to search for errors or odd behavior. The dig program was used, making 100 queries at a time, in a looped script. We failed a server after three successive dig queries failed in a row. We found that the behavior of both servers was to cease at once during these failure. This method uses a user-perspective establishment that the server has become unavailable, usually but not always at saturation, as noted and excepted in the individual test results below.

ARPFlood is a test that floods a network segment with inaccurate arp protocol-based information that forces devices to update their own arp tables, and forcing some devices to reply in retort. Because of the pseudo-random addresses sent, some servers decay rapidly in performance, while others can actually become increasingly or totally unavailable or crash.

Testing Method: IP/MAC address pairs are sent to the server under test until the server stops responding to dig queries. Queries are ramped until the server doesn't respond to dig queries.

Secure64 DNS Results: 7.02 million flood packets saturated the wire and caused the server to stop responding. The data rate at saturation was approximately 828mb/s. The server responded to queries after the test load became quiescent essentially immediately.

Linux/BIND Server Results: 181 thousand flood packets caused the server to stop responding, although the wire did not saturate; the data rate at saturation 241mb/s. The server began to respond to queries approximately 7s after the test load became quiescent.

Evasive UDP attacks uses a method that makes a server believe that a session already exists, but at ports that haven't been actually established or authenticated. The behavior of the server is to respond to the UDP requests. Servers under Evasive UDP attack typically allocate session overhead to these attack, then become overwhelmed by the increasing number of responses required.

Secure64 DNS Results: The server saturated at 6.28 million packets were sent at a data rate of approximately 805mb/s. The server responded to queries after the test load became quiescent essentially immediately.

Linux/BIND Server Results: The server saturated at 309K packets at a data rate of 541 mb/s and began responding essentially immediately after the test load was removed.

Ping of Death is a specially formed, illegally long packet designed to give the server targeted a case of indigestion; some servers respond to the oversized packet by causing application faults that allow an attacker to gain authentication to root resources of the server.

Test Method: Various size packets using the ICMP protocol are sent to the server, increasing in apparent size (width of packet and payload) and quantity (additional apparent packet origins by IP and MAC address pairing). Servers under attack usually either crash or misbehave when packet size (width) is increased beyond a duration point, or buffers overflow when many of them are received at once.

Secure64 DNS Results: the server appeared to ignore the attack packets; load increased until dig no longer responded at a data rate of approximately 654mb/sec. The server immediately responded to queries once the test load was removed.

Linux/BIND Server Results: the server appeared to ignore the attack packets; however the server stopped responding to queries at a very low data rate of 70mb/sec.

Ping Sweep (flood) attack floods ports with ICMP requests at different ports. Typical server responses to this attack are to tie up server resources by checking whether a service exists at the port, then responding with an appropriate 'service not available' message. Server resources are subsequently overwhelmed, resulting in denial of legitimate service requests.

Test Method: increase pings by port and independent IP/MAC pair to cause server to spawn an error message and service the request.

Secure64 DNS Results: The server responded to base messaging only. Queries were acknowledged up to wire saturation at a data rate of approximately 830mb/s.

Linux/BIND Server Results: The server responded to three ports. Queries were acknowledged until a data rate of 165mb/s had been achieved, then remained unavailable after the test load was removed for approximately 9s.

TCP SynFlood attacks use the ordered packet sequence needs of the TCP protocol to tie up server connections until the server becomes unavailable to legitimate service requests.

Test Method: IP/MAC pairs are generated, ramped up, then flooding activity begins until wire saturation is achieved/noted.

Secure64 DNS Results: server did not acknowledge the attack (little or no apparent reaction), and operated until data rate saturated the wire at approximately 830mb/s. When test load was removed, server responded essentially immediately to query requests.

Linux/BIND Server Results: server responded only mildly to the attack (had occasional very short pauses in response). Queries were serviced until a data rate of approximately 232mb/s was achieved, and server became available after the test load was removed essentially immediately.

UDP Flood sends a random pattern of UDP messages to a server. The server will check to see if a service exists on the port, and if it does not, will send an ICMP message to the sender with a 'service not available' type message.

Test Method: IP/MAC pairs are generated, and during the generation, a secondary activity of the IP/MAC pairs sends a sequence of pseudo-random port queries between ports 2 and 65K. Twenty queries per IP/MAC address pair are performed.

Secure64 DNS Results: A small impedance was seen if the IP address was on the same segment. Otherwise, the server responded until the wire was saturated at a data rate of approximately 791mb/s. The server became available essentially immediately after the test flood messaging load was removed.

Linux/BIND Server Results: Far worse performance was seen. The server responded to the attack, and the server stopped responding after only 41K UDP messages had been sent. The server stayed unavailable for 29 seconds after the test flood messaging load had been removed.

Unreachable Host attack simply sends messages to a server that a host was unreachable. The server tries to find what sent the message to send it to the service that ostensibly requested or needs the message. The server rapidly becomes unavailable to legitimate requests, distracted by the need to service the message.

Test Method: We trimmed the Spirent appliance to focus the messages specifically at port 53, the DNS port, to observe the behavior of the server under test, while continuing to pump legitimate queries to the server, even from the same IP/MAC pairs used by the attacking 'device' (the Spirent test appliance).

Secure64 DNS Results: Nominal impedance was seen to the point of wire saturation at approximately 810mb/s. When we removed the test load, queries were essentially processed immediately.

Linux/BIND Server Results: BIND's behavior was different. Server response decayed considerably after only 20K messages were sent although the server recovered quickly. The data rate was approximately 163mb/sec at saturation.

XmasTree Attack is used to send malformed packets, with all flags in the header of the packet set high, sent randomly to a server, to cause it to respond. By responding, the server ties up resources and may deny legitimate service requests.

Test Method: IP/MAC pairs are generated in the Spirent appliance, then used to probe the server under test. Any response to the packets is judged as a denial of service.

Secure64 DNS Results: The server did not respond to the attack.

Linux/BIND Server Results: The server did not respond to the attack.

© Copyright 2007 ExtremeLabs, Inc/Secure64 Software Corporation.
Entire Contents All Rights Reserved.