19-601, Spring 2002
Homework 2

Daniel M. Vogel

February 17, 2002

1 Denial of service attacks

1.1 Definitions

It is said that:

- "Stacheldraht shares TFN's features of distributed network denial of service by way of ICMP flood, SYN flood, UDP flood, and 'Smurf' style attacks.
- Unlike the original TFN and TFN2K, the stacheldraht code does not contain the "on demand" root shell bound to a TCP port (it may be based on earlier TFN code than was made public by Mixter in mid-1999).
- Unlike trinoo, which uses UDP for communication between handlers and agents, or the original TFN, which uses ICMP for communication between the handler and agents, stacheldraht uses TCP and ICMP

Can you make sense of these different statements and explain to what extent Stacheldraht is an "improvement" or one step up in the art of designing DDOS tools. (Be as precise technically as possible).

- Stacheldraht provides the common denominator of denial of service attacks. These are, in general, attacks which succeed only by utilizing as much bandwidth as available, for the purpose of resource starvation.
- Stacheldraht is not a direct derivitive of TFN, although it may have been instantiated from an early code base. This may indicate some collaboration between the authors of both systems.
- Stacheldraht is distinct from trinoo and TFN in that it establishes stateful connections between handlers and agents. This may make the agent-handler relationship more complex as well as more reliable, as the denial of service attacks are less likely to disrupt established TCP sessions.

  As well, Stacheldraht introduced encrypted communication between the attacker and handlers. This is a significant boost from previous wide-open traffic in earlier systems.

1.2 Kaiten improvements

Kaiten's main differences were that it came as payload within a worm, exploited a specific vulnerability in ideal zombie hosts (Microsoft SQL servers: big, on fast networks), and used IRC as its control mechanism. Kaiten also was installed through BackOrifice. Stacheldraht generally required manual installation (and unix machines).

1.3 Defense against DOS attacks on DNS

DNS is a poor approach to the problem. DNSSEC may provide some better options. But heirarchical, centrally managed systems are natural targets.

2 Defense against Buffer Overflow and Intrusion Detection

2.1 StackGuard, StackShield, and bypasses

2.1.1 How do StackGuard and StackShield work?

They monitor and preserve the integrity of function return addresses. Modification of function return addresses through overflowing buffers is the most common route of buffer overflow exploits. StackGuard provides a mechanism for detecting return address modification. StackShield provides a mechanism for removing the return address from being in deterministic, overflowable position.

2.1.2 How is it possible to still "beat them"

Both of the Stack* utilities guard against modification to the return pointers. They do not (as of the article) guard against arbitrary buffer modification. As a result, longjmp, function pointers, various other branching memory calls may still be modified, opening up exploitable options. As well, modification of the Global Offset Table makes any system vulnerable. If you can change blue to red, what does it matter if you ask for a legitimate paint?

2.1.3 Is there no definitive way to protect against BOs?

As long as users are permitted to address memory directly, there will be the potential for an exploitable program being written. The article does not eliminate the functionality of the Stack* utilities. It demonstrates that poorly audited code is still vulnerable to attack. Buffer Overflows can be avoided with or without the Stack* utilities – and the Stack* utilities reduce the exposure of less secure code.

So no, while users are permitted to address memory directly, there is no definitive way to protect against BOs. Because operating systems will continue to be reliant on datastructures for preserving important information, and when any code can be run at a root level, this is at risk (see the various Phrack 58 articles on kernel hacking... there are no limits that the operating system can impose on root).

2.2 Network Layer IDS problems

The suite of problems against IDS can be summarized by a lack of perfect knowledge of a network topology. This includes explicit information on the distance and various magic names of each node, as well as a detailed on the behavior of the node. Otherwise, a discrepancy may be opened between the information which is passed between the attacker and the host versus the attack and the IDS. There are exploitation frameworks (insertion, evasion) which take advantage of either situation.

For forensic or monitoring use, a NIDS is only good if it is WYSIWYG. And WYSIWYG, by the techniques detailed here, is only possible with perfect knowledge of the network graph and its component nodes. This may or may not be possible, and is certainly infeasible for networks of reasonable size.

Phrack 56-11 offers a comparable rundown on NIDS', as well as documenting why passive network monitoring IDS' will be irrelevant in the near future.