A Look Back at "Security Problems in the TCP/IP Protocol Suite"

Steven M. Bellovin
AT&T Labs—Research
bellovin@acm.org

Abstract

About fifteen years ago, I wrote a paper on security problems in the TCP/IP protocol suite, In particular, I focused on protocol-level issues, rather than implementation flaws. It is instructive to look back at that paper, to see where my focus and my predictions were accurate, where I was wrong, and where dangers have yet to happen. This is a reprint of the original paper, with added commentary.

1. Introduction

The paper "Security Problems in the TCP/IP Protocol Suite" was originally published in *Computer Communication Review*, Vol. 19, No. 2, in April, 1989. It was a protocol-level analysis; I intentionally did not consider implementation or operational issues. I felt—and still feel—that that was the right approach. Bugs come and go, and everyone's operational environment is different. But it's very hard to fix protocol-level problems, especially if you want to maintain compatibility with the installed base.

This paper is a retrospective on my original work. New commentary is shown indented, in a sans serif font. The original text is otherwise unchanged, except for possible errors introduced when converting it from *troff* to LaTeX. I've left the references intact, too, even if there are better versions today. The reference numbers and pagination are, of course, different; the section numbers remain the same, except for a new "Conclusions" section. As a general rule, the commentary *follows* the section it's discussing.

It helps to understand where this paper came from. When I started work at Bell Labs Murray Hill in 1982, I assumed ownership of $1\frac{1}{2}$ of the first three pieces of Ethernet cable in all of AT&T, then a giant monopoly telephone company. My lab had one cable, another lab had a second, and a "backbone" linked the two labs. That backbone grew, as other labs connected to it. Eventually, we scrounged funds to set up links to other Bell Labs locations; we called the resuling network the "Bell Labs Internet" or the "R&D Internet", the neologism "Intranet" not having been invented.

Dedicated routers were rare then; we generally stuck a second Ethernet board into a VAX or a Sun and used it to do the routing. This mean that the routing software (we used Berkeley's *routed*) was accessible to system administrators. And when things broke, we often discovered that it was a routing problem: someone had misconfigured their machine. Once, we found that someone had plugged a new workstation into the Murray Hill backbone, rather than into his department's network; worse yet, the (proprietary) address assignment software on his machine didn't see any (proprietary) address assignment servers on that network, so it allocated .1—the gatewayy router—to itself. These two situations worried me; it was clear that anything that could happen by accident could be done deliberately, possibly with serious consequences.

Several other things focused my attention on security even more. One was Robert Morris' discovery of sequence number guessing attacks; these are discussed extensively below. Another was the "Shadow Hawk" incident—a teenager broke into various AT&T computers [2]. He was detected when he tried to use *uucp* to grab password files from various Research machines; a number of us had installed detectors for exactly that sort of activity, and noticed the un-

usual behavior. At that, we were lucky—most of the connectivity within AT&T was via the proprietary Datakit network, which he didn't know how to exploit.

By the time of the Internet worm of 1988, this paper was already in substantially its current form. The result was an analysis (to the best of my ability; I was relatively new to security at the time) of protocol-level problems in TCP/IP.

The original paper was criticized in [54]. Some of the criticisms were valid; some, in my opinion, were not. At the time, I chose not to publish a detailed rebuttal; I will not do so here. I have, where appropriate, noted where my analysis was especially incorrect. I did and do feel that my conclusions were substantially correct.

The TCP/IP protocol suite [41, 21] which is very widely used today, was developed under the sponsorship of the Department of Defense. Despite that, there are a number of serious security flaws inherent in the protocols. Some of these flaws exist because hosts rely on IP source address for authentication; the Berkeley "*r*-utilities" [22] are a notable example. Others exist because network control mechanisms, and in particular routing protocols, have minimal or nonexistent authentication.

When describing such attacks, our basic assumption is that the attacker has more or less complete control over some machine connected to the Internet. This may be due to flaws in that machine's own protection mechanisms, or it may be because that machine is a microcomputer, and inherently unprotected. Indeed, the attacker may even be a rogue system administrator.

1.1. Exclusions

We are not concerned with flaws in particular implementations of the protocols, such as those used by the Internet "worm" [95, 90, 38] Rather, we discuss generic problems with the protocols themselves. As will be seen, careful implementation techniques can alleviate or prevent some of these problems. Some of the protocols we discuss are derived from Berkeley's version of the UNIX system; others are generic Internet protocols.

We are also not concerned with classic network attacks, such as physical eavesdropping, or altered or injected messages. We discuss such problems only in so far as they are facilitated or possible because of protocol problems.

For the most part, there is no discussion here of vendor-specific protocols. We do discuss some problems with Berkeley's protocols, since these have become de facto standards for many vendors, and not just for UNIX systems.

One of the criticisms in [54]) was that I had lumped Berkeley-specific protocols together with standardized protocols described in RFCs. It's quite clear from the preceeding paragraph that I understood the difference. However, the use of address-based authentication—a major flaw that I criticize throughout the paper—was peculiar to Berkeley's software; I did not make that distinction clear. It is indeed a bad way to do authentication, but it was not blessed by any official standard.

2. TCP Sequence Number Prediction

One of the more fascinating security holes was first described by Morris [70]. Briefly, he used TCP sequence number prediction to construct a TCP packet sequence without ever receiving any responses from the server. This allowed him to spoof a trusted host on a local network.

The normal TCP connection establishment sequence involves a 3-way handshake. The client selects and transmits an initial sequence number ISN_C, the server acknowledges it and sends its own sequence number ISN_S, and the client acknowledges that. Following those three messages, data transmission may take place. The exchange may be shown schematically as follows:

$$
\begin{array}{lll}
 & C \rightarrow S: & \mathit{SYN}(\text{ISN}_C) \\
 & S \rightarrow C: & \mathit{SYN}(\text{ISN}_S), \mathit{ACK}(ISN_C) \\
 & C \rightarrow S: & \mathit{ACK}(\text{ISN}_S) \\
 & C \rightarrow S: & \text{data} \\
\text{and/or} & & \\
 & S \rightarrow C: & \text{data}
\end{array}
$$

That is, for a conversation to take place, C must first hear ISN_S, a more or less random number.

Suppose, though, that there was a way for an intruder X to *predict* ISN_S. In that case, it could send the following sequence to impersonate trusted host T:

$$
\begin{array}{ll}
X \rightarrow S: & \mathit{SYN}(\text{ISN}_X), \mathit{SRC} = T \\
S \rightarrow T: & \mathit{SYN}(\text{ISN}_S), \mathit{ACK}(\text{ISN}_X) \\
X \rightarrow S: & \mathit{ACK}(\text{ISN}_S), \mathit{SRC} = T \\
X \rightarrow S: & \mathit{ACK}(\text{ISN}_S), \mathit{SRC} = T, nasty - data
\end{array}
$$

Even though the message $S \rightarrow T$ does not go to X, X was able to know its contents, and hence could send data. If X were to perform this attack on a connection that allows command execution (i.e., the Berkeley *rsh* server), malicious commands could be executed.

How, then, to predict the random ISN? In Berkeley systems, the initial sequence number variable is incremented by a constant amount once per second, and by half that amount each time a connection is initiated. Thus, if one initiates a legitimate connection and observes the ISN_S used, one can calculate, with a high degree of confidence, ISN'_S used on the next connection attempt.

Morris points out that the reply message

$$S \rightarrow T : SYN(\text{ISN}_S), ACK(\text{ISN}_X)$$

does not in fact vanish down a black hole; rather, the real host T will receive it and attempt to reset the connection. This is not a serious obstacle. Morris found that by impersonating a server port on T, and by flooding that port with apparent connection requests, he could generate queue overflows that would make it likely that the $S \rightarrow T$ message would be lost. Alternatively, one could wait until T was down for routine maintenance or a reboot.

> I mischaracterized Morris' paper on this point. While flooding can work—without explicitly stating it, I anticipated the denial of service attacks that started occurring in 1996—Morris in fact exploited an implementation error in the Berkeley kernel to accomplish his goal with many fewer packets. That flaw (described in [10] as well as in Morris' paper) received very little attention at the time, and was not fixed until many years later.
>
> For that matter, sequence number attacks received little attention outside of my paper, until Kevin Mitnick reimplemented Morris' idea and used it to attack Tsutomu Shimomura [93]. Shimomura then proceeded to track down Mitnick.

A variant on this TCP sequence number attack, not described by Morris, exploits the *netstat* [86] service. In this attack, the intruder impersonates a host that is down. If *netstat* is available on the target host, it may supply the necessary sequence number information on another port; this eliminates all need to guess.[1]

> The Berkeley implementation of *netstat* was dangerous, but not for the reasons that I gave here. It did list the open ports on the machine, as well as all current connections; both items are very valuable to would-be attackers. Indeed, discovering the former is a major piece of functionality of many attack tools. Fortunately, even in 1989 *netstat* was not available by default on any 4.2BSD or 4.3BSD systems. There were still TOPS-20 systems on the net at that time; those systems had a vulnerable *netstat*, a fact I refrained from mentioning for fear of pointing attackers at targets. Actual output is shown in Figure 1.
>
> There are several salient points here. The first, of course, which I stressed at the time, is that address-based authentication is very vulnerable to attack. I will return to this point later. A second point is a threat I mention later, but not in this context: if you know the sequence numbers of an active session, you can hijack it. This attack was implemented a few years later by Joncheray [53].
>
> A more important point (and this is one made in [54]) is that the *r*-utilities are implicitly relying on TCP sequence numbers—and hence on TCP session correctness—for security properties. However, TCP was never designed to be a secure protocol, nor were there ever any guarantees about the properties of the sequence number. The underlying issue is this: *what properties of a layer are "exported" to a higher layer?* Assuming too much at any higher later is a mistake; it can lead to correctness failurs as well as to security failures. For that matter, it is necessary to inquire even more closely, even of sequence numbers in a security protocol: what properties are they guaranteed to have? Are they simply packet sequence numbers, or can they be used as, say, the initialization vector for counter mode encryption [35]?
>
> Was there a security problem? Yes, there certainly was, as demonstrated graphically a few years later in the Mitnick vs. Shimomura incident. But the *architectural* flaw was the assumption that TCP sequence numbers had security properties which they did not. (Ironically, I have Heard that analyses of the security properties of sequence numbers were, in fact, done in the classified world—and they concluded that such attacks were not feasible. . .)
>
> The sequence number attack story isn't over. In 2004, Watson observed that TCP reset packets were honored if the *RST* bit was set on a packet whose initial sequence number was anywhere within the receive window (see US-CERT Technical Cyber Security Alert TA04-111A). On modern systems, the receive window is ofen 32K bytes or more,

1 The *netstat* protocol is obsolete, but is still present on some Internet hosts. Security concerns were not behind its elimination.

JCN	STATE	LPORT	FPORT	FGN-HOST	R-SEQUENCE	S-SEQUENCE	SENDW
0,-1	-3-.EST.OOPA---	15	2934	ATT.ARPA	333888001	760807425	4096
6,6	FIN.FIN.--P----	15	0	0,0,0,0 0	0	0	
6,5	FIN.FIN.--P----	79	0	0,0,0,0 0	0	0	
0,21	-3-.EST.O-PAV--	23	4119	26,1,0,16	2928942175	701235845	319
0,2	-3-.EST.O-PAV--	23	1792	192,33,33,115	739613342	660542923	4096

Figure 1. Output from a TOPS-20 *netstat* command. Note the "send" and "'receive" sequence numbers. The first line in the status display is the session I used to retrieve the data.

which means that it takes less than 2^{17} trials to generate such a packet via a blind attack. That sounds like a lot of packets, and it's only a denial of service attack, but for long-lived sessions (and in particular for BGP [84] sessions between routers), it's quite a feasible attack. Furthermore, tearing down a single BGP session has wide-spread effects on the global Internet routing tables.

Defenses

Obviously, the key to this attack is the relatively coarse rate of change of the initial sequence number variable on Berkeley systems. The TCP specification requires that this variable be incremented approximately 250,000 times per second; Berkeley is using a much slower rate. However, the critical factor is the granularity, not the average rate. The change from an increment of 128 per second in 4.2BSD to 125,000 per second in 4.3BSD is meaningless, even though the latter is within a factor of two of the specified rate.

Let us consider whether a counter that operated at a true 250,000 hz rate would help. For simplicity's sake, we will ignore the problem of other connections occurring, and only consider the fixed rate of change of this counter.

To learn a current sequence number, one must send a *SYN* packet, and receive a response, as follows:

$$\begin{aligned} X \rightarrow S: &\quad \mathit{SYN}(\mathrm{ISN}_X) \\ S \rightarrow X: &\quad \mathit{SYN}(\mathrm{ISN}_S), \mathit{ACK}(\mathrm{ISN}_X) \end{aligned} \tag{1}$$

The first spoof packet, which triggers generation of the next sequence number, can immediately follow the server's response to the probe packet:

$$X \rightarrow S: \ \mathit{SYN}(\mathrm{ISN}_X), \mathit{SRC} = T \tag{2}$$

The sequence number ISN_S used in the response

$$S \rightarrow T: \ \mathit{SYN}(\mathrm{ISN}_S), \mathit{ACK}(\mathrm{ISN}_X)$$

is uniquely determined by the time between the origination of message (1) and the receipt at the server of message (2). But this number is precisely the round-trip time between X and S. Thus, if the spoofer can accurately measure (and predict) that time, even a 4 μ-second clock will not defeat this attack.

How accurately can the trip time be measured? If we assume that stability is good, we can probably bound it within 10 milliseconds or so. Clearly, the Internet does not exhibit such stability over the long-term [64], but it is often good enough over the short term.[2] There is thus an uncertainty of 2500 in the possible value for ISN_S. If each trial takes 5 seconds, to allow time to re-measure the round-trip time, an intruder would have a reasonable likelihood of succeeding in 7500 seconds, and a near-certainty within a day. More predictable (i.e., higher quality) networks, or more accurate measurements, would improve the odds even further in the intruder's favor. Clearly, simply following the letter of the TCP specification is not good enough.

We have thus far tacitly assumed that no processing takes places on the target host. In fact, some processing does take place when a new request comes in; the amount of variability in this processing is critical. On a 6 MIPS machine, one tick—4 μ-seconds—is about 25 instructions. There is thus considerable sensitivity to the exact instruction path followed. High-priority interrupts, or a slightly different TCB allocation sequence, will have a comparatively large effect on the actual value of the next sequence number. This randomizing effect is of considerable advantage to the target. It should be noted, though, that faster machines are *more* vulnerable to this attack, since the variability of the instruction path will take less real time, and hence affect the increment less. And of course, CPU speeds are increasing rapidly.

This suggests another solution to sequence number attacks: randomizing the increment. Care must be taken to use sufficient bits; if, say, only the low-order 8 bits were picked randomly, and the granularity of the increment was coarse, the intruder's work factor is only multiplied by 256. A combination of a fine-granularity increment and a small random number generator, or just a 32-bit generator, is better. Note, though, that many pseudo-random number generators are

2 At the moment, the Internet may not have such stability even over the short-term, especially on long-haul connections. It is not comforting to know that the security of a network relies on its low quality of service.

easily invertible [13]. In fact, given that most such generators work via feedback of their output, the enemy could simply compute the next "random" number to be picked. Some hybrid techniques have promise—using a 32-bit generator, for example, but only emitting 16 bits of it—but brute-force attacks could succeed at determining the seed. One would need at least 16 bits of random data in each increment, and perhaps more, to defeat probes from the network, but that might leave too few bits to guard against a search for the seed. More research or simulations are needed to determine the proper parameters.

Rather than go to such lengths, it is simpler to use a cryptographic algorithm (or device) for ISN_S generation. The Data Encryption Standard [73] in *electronic codebook mode* [74] is an attractive choice as the ISN_S source, with a simple counter as input. Alternatively, DES could be used in *output feedback mode* without an additional counter. Either way, great care must be taken to select the key used. The time-of-day at boot time is not adequate; sufficiently good information about reboot times is often available to an intruder, thereby permitting a brute-force attack. If, however, the reboot time is encrypted with a per-host secret key, the generator cannot be cracked with any reasonable effort.

Performance of the initial sequence number generator is not a problem. New sequence numbers are needed only once per connection, and even a software implementation of DES will suffice. Encryption times of 2.3 milliseconds on a 1 MIPS processor have been reported [12].

An additional defense involves good logging and alerting mechanisms. Measurements of the round-trip time—essential for attacking RFC-compliant hosts—would most likely be carried out using ICMP *Ping* messages; a "transponder" function could log excessive ping requests. Other, perhaps more applicable, timing measurement techniques would involve attempted TCP connections; these connections are conspicuously short-lived, and may not even complete *SYN* processing. Similarly, spoofing an active host will eventually generate unusual types of *RST* packets; these should not occur often, and should be logged.

> After many years of thinking about it, I finally came up with a a solution to classical sequence number attacks. The scheme, described in RFC 1948 [10], used a cryptographic hash function to create a separate sequence number space for each "connection", a connection being defined per RFC 791 [81] as the unique 4-tuple <localhost, localport, remotehost, remoteport>. This scheme has not been adopted as widely as I would like; my claim here that extra CPU load during TCP connection establishment was irrelevant was rendered obsolete by the advent of very large Web servers. Indeed, maximum TCP connection rate is a vital metric when assessing modern systems.
>
> Instead, many implementations use random ISNs or (especially) random increments. This has obvious negative effects on the correctness of TCP in the presence of duplicate packets, a property that *is* guaranteed to higher layers. (Also see the appendix of [52].) Worse yet, Newsham pointed out that by the central limit theorem, the sum of a sequence of random increments will have a normal distribution, which implies that the actual range of the ISNs is quite small. (see CERT Advisory CA-2001-09).
>
> There are hybrid schemes that don't fall to these attacks, but the underlying message is the same as it was in 1989: don't rely on TCP sequence numbers for security.
>
> Also worth noting is the suggestion that intrusion detection system can play a role: they can alert you to an attack that for some reason you can't ward off.

3. The Joy of Routing

> As noted at the beginng, routing problems were one of the initial motivations for this work. For a fair number of years, though, I said that "the only attack I discussed in this paper that hasn't been seen in the wild is routing attacks". That's no longer the case; the bad guys have caught up.

Abuse of the routing mechanisms and protocols is probably the simplest protocol-based attack available. There are a variety of ways to do this, depending on the exact routing protocols used. Some of these attacks succeed only if the remote host does source address-based authentication; others can be used for more powerful attacks.

A number of the attacks described below can also be used to accomplish denial of service by confusing the routing tables on a host or gateway. The details are straightforward corollaries of the penetration mechanisms, and will not be described further.

3.1. Source Routing

If available, the easiest mechanism to abuse is IP source routing. Assume that the target host uses the reverse of the source route provided in a TCP open request for return traffic. Such behavior is utterly reasonable; if the originator of the connection wishes to specify a particular path for some

reason—say, because the automatic route is dead—replies may not reach the originator if a different path is followed.

The attacker can then pick any IP source address desired, including that of a trusted machine on the target's local network. Any facilities available to such machines become available to the attacker.

> Again, I'm focusing here on address-based authentication. But I move on to more subtle routing attacks.

Defenses

It is rather hard to defend against this sort of attack. The best idea would be for the gateways into the local net to reject external packets that claim to be from the local net. This is less practical than it might seem since some Ethernet[3] network adapters receive their own transmissions, and this feature is relied upon by some higher-level protocols. Furthermore, this solution fails completely if an organization has two trusted networks connected via a multi-organization backbone. Other users on the backbone may not be trustable to the same extent that local users are presumed to be, or perhaps their vulnerability to outside attack is higher. Arguably, such topologies should be avoided in any event.

> Note that I'm alluding here to what are now called "firewalls". I'm not sure why I didn't use that word in this paper; I was using it in email several years earlier.

A simpler method might be to reject pre-authorized connections if source routing information was present. This presumes that there are few legitimate reasons for using this IP option, especially for relatively normal operations. A variation on this defense would be to analyze the source route and accept it if only trusted gateways were listed; that way, the final gateway could be counted on to deliver the packet only to the true destination host. The complexity of this idea is probably not worthwhile.

> Newer versions of the *r*-utilities do, in fact, reject source-routed connections. But there's a more subtle risk: though they reject the connection attempt, an *ACK* packet is returned; this conveys the information needed to launch a sequence number attack.
>
> The most common configuration today is to reject source-routed packets at border routers, whether or not they fill other firewall-related roles. Source routing *is* permitted on the backbone; ISPs use it to view paths from different vantage points. Internally, such packets may or may not be blocked; the question is irrelevant for many organizations, since the rise of Microsoft Windows and the relative demise of UNIX-style remote login has rendered the attack somewhat less interesting.

Some protocols (i.e., Berkeley's *rlogin* and *rsh*) permit ordinary users to extend trust to remote host/user combinations. In that case, individual users, rather than an entire system, may be targeted by source routing attacks.[4] Suspicious gateways [69] will not help here, as the host being spoofed may not be within the security domain protected by the gateways.

> Note the warning here against putting too much trust in firewalls: they don't defend against insider attacks.

3.2. Routing Information Protocol Attacks

The *Routing Information Protocol* [49], (RIP) is used to propagate routing information on local networks, especially broadcast media. Typically, the information received is unchecked. This allows an intruder to send bogus routing information to a target host, and to each of the gateways along the way, to impersonate a particular host. The most likely attack of this sort would be to claim a route to a particular unused host, rather than to a network; this would cause all packets destined for that host to be sent to the intruder's machine. (Diverting packets for an entire network might be too noticeable; impersonating an idle work-station is comparatively risk-free.) Once this is done, protocols that rely on address-based authentication are effectively compromised.

This attack can yield more subtle, and more serious, benefits to the attacker as well. Assume that the attacker claims a route to an active host or workstation instead. All packets for that host will be routed to the intruder's machine for inspection and possible alteration. They are then resent, using IP source address routing, to the intended destination. An outsider may thus capture passwords and other sensitive data. This mode of attack is unique in that it affects outbound calls as well; thus, a user calling out from the targeted host can be tricked into divulging a password. Most of the earlier attacks discussed are used to forge a source address; this one is focused on the destination address.

3 Ethernet is a registered trademark of Xerox Corporation.

4 Permitting ordinary users to extend trust is probably wrong in any event, regardless of abuse of the protocols. But such concerns are beyond the scope of this paper.

This and [77] are the earliest mentions of routing attacks in the literature. The attacks described here—abusing the routing protocols for eavesdropping and/or packet modification—remain a very serious threat. Indeed, a National Research Council study [89] identified routing attacks as one of the two major threats to the Internet. While there are proposals to solve this problem (see, for example, [71, 56, 55]), nothing has been implemented; all of the proposed solutions have their drawbacks. Defense against routing attacks must still be considered a research problem.

Routing attacks have happened frequently by accident. In the most famous case, known as the "AS 7007" incident, an ISP started advertising that it had the best routes to most of the Internet. Even after they powered down their router, it took more than four hours for the global routing tables to stabilize.

As suggested here, more subtle routing problems are harder to diagnose. AT&T's dial-up Internet service was knocked off the air for many hours when another ISP started advertising a route to a small, internal network. There are many other such incidents as well.

Are malicious routing attacks happening? Yes, they are, and the culprits are a very low life form: the spammers. In some cases, they're hijacking a route, injecting spam, and then withdrawing the route. The attack is hard to trace, because by the time someone notices it the source addresses of the email are (again) either non-existent or innocent.

Defenses

A RIP attack is somewhat easier to defend against than the source-routing attacks, though some defenses are similar. A paranoid gateway—one that filters packets based on source or destination address—will block any form of host-spoofing (including TCP sequence number attacks), since the offending packets can never make it through. But there are other ways to deal with RIP problems.

Filtering out packets with bogus source addresses would help against many forms of attack. Too few ISPs do it, even though it is a recommended practice [42].

One defense is for RIP to be more skeptical about the routes it accepts. In most environments, there is no good reason to accept new routes to your own local networks. A router that makes this check can easily detect intrusion attempts. Unfortunately, some implementations rely on hearing their own broadcasts to retain their knowledge of directly-attached networks. The idea, presumably, is that they can use other networks to route around local outages. While fault-tolerance is in general a good idea, the actual utility of this technique is low in many environments compared with the risks.

It would be useful to be able to authenticate RIP packets; in the absence of inexpensive public-key signature schemes, this is difficult for a broadcast protocol. Even if it were done, its utility is limited; a receiver can only authenticate the immediate sender, which in turn may have been deceived by gateways further upstream.

This paragraph summarizes the essential difficulty in defending against routing attacks: the problem can originate with non-local machines. That is, even if your routing link to your neighbors is authenticated, they may be deceived rather than dishonest.

More and more sites are starting to protect their routing protocols against direct attacks. The most commonly used mechanism is described in [50], caveats on key selection are given in [59]. Another mechanism is the so-called TTL Security Hack [45]: if a packet is supposed to originate on-link, send it with a TTL of 255, and verify that on receipt. Any off-link packets will have passed through at least one router which would have decremented the TTL.

Even if the local routers don't implement defense mechanisms, RIP attacks carry another risk: the bogus routing entries are visible over a wide area. Any router (as opposed to host) that receives such data will rebroadcast it; a suspicious administrator almost anywhere on the local collection of networks could notice the anomaly. Good log generation would help, but it is hard to distinguish a genuine intrusion from the routing instability that can accompany a gateway crash.

[104] analyzes how stable routes are to major name servers. The answer is encouraging: such routes change very infrequently..

3.3. Exterior Gateway Protocol

The *Exterior Gateway Protocol* (EGP) [65] is intended for communications between the core gateways and so-called *exterior gateways*. An exterior gateway, after going through a *neighbor acquisition* protocol, is periodically polled by the core; it responds with information about the

networks it serves. These networks must all be part of its *autonomous system*. Similarly, the gateway periodically requests routing information from the core gateway. Data is not normally sent except in response to a poll; furthermore, since each poll carries a sequence number that must be echoed by the response, it is rather difficult for an intruder to inject a false route update. Exterior gateways are allowed to send exactly one spontaneous update between any two polls; this, too, must carry the sequence number of the last poll received. It is thus comparatively difficult to interfere in an on-going EGP conversation.

One possible attack would be to impersonate a second exterior gateway for the same autonomous system. This may not succeed, as the core gateways could be equipped with a list of legitimate gateways to each autonomous system. Such checks are not currently done, however. Even if they were, they could be authenticated only by source IP address.

A more powerful attack would be to claim reachability for some network where the real gateway is down. That is, if gateway G normally handles traffic for network N, and G is down, gateway G' could advertise a route to that network. This would allow password capture by assorted mechanisms. The main defense against this attack is topological (and quite restrictive): exterior gateways must be on the same network as the core; thus, the intruder would need to subvert not just any host, but an existing gateway or host that is directly on the main net.

A sequence number attack, similar to those used against TCP, might be attempted; the difficulty here is in predicting what numbers the core gateway is using. In TCP, one can establish arbitrary connections to probe for information; in EGP, only a few hosts may speak to the core. (More accurately, the core could only speak to a few particular hosts, though as noted such checks are not currently implemented.) It may thus be hard to get the raw data needed for such an attack.

EGP was reasonably secure because of the very restricted topologies it could deal with: a single core that talked with a variety of stub networks. It couldn't possibly work for today's Internet, which is why it's been replaced by BGP [84]. But loosening the restrictions has had negative effects as well: BGP is easier to attack, since it's conveying more complex information.

I asserted in the original paper that it was possible to filter addresses announced in routing packets. That wasn't done in 1989; today, however, major ISPs do filter advertisements from stub networks they talk to. But not all ISPs do that, and it's not possible to do much filtering when talking to peers or transit ISPs.

3.4. The Internet Control Message Protocol

The *Internet Control Message Protocol* (ICMP) [79] is the basic network management tool of the TCP/IP protocol suite. It would seem to carry a rich potential for abuse. Surprisingly, ICMP attacks are rather difficult; still, there are often holes that may be exploited.

The first, and most obvious target, is the ICMP *Redirect* message; it is used by gateways to advise hosts of better routes. As such it can often be abused in the same way that RIP can be. The complication is that a Redirect message must be tied to a particular, existing connection; it cannot be used to make an unsolicited change to the host's routing tables. Furthermore, Redirects are only applicable within a limited topology; they may be sent only from the first gateway along the path to the originating host. A later gateway may not advise that host, nor may it use ICMP Redirect to control other gateways.

Suppose, though, that an intruder has penetrated a secondary gateway available to a target host, but not the primary one. (It may suffice to penetrate an ordinary host on the target's local network, and have it claim to be a gateway.) Assume further that the intruder wishes to set up a false route to trusted host T through that compromised secondary gateway. The following sequence may then be followed. Send a false TCP open packet to the target host, claiming to be from T. The target will respond with its own open packet, routing it through the secure primary gateway. While this is in transit, a false Redirect may be sent, claiming to be from the primary gateway, and referring to the bogus connection. This packet will appear to be a legitimate control message; hence the routing change it contains will be accepted. If the target host makes this change to its global routing tables, rather than just to the per-connection cached route, the intruder may proceed with spoofing host T.

Some hosts do not perform enough validity checks on ICMP Redirect messages; in such cases, the impact of this attack becomes similar to RIP-based attacks.

ICMP may also be used for targeted denial of service attacks. Several of its messages, such as *Destination Unreachable* and *Time to Live Exceeded*, may be used to reset existing connections. If the intruder knows the local and remote port numbers of a TCP connection, an ICMP packet aimed at that connection may be forged.[5] Such information is sometimes available through the *netstat* service.

A more global denial of service attack can be launched by sending a fraudulent *Subnet Mask Reply* message. Some

5 In fact, such programs are available today; they are used as administrative tools to reset hung TCP connections.

hosts will accept any such message, whether they have sent a query or not; a false one could effectively block all communications with the target host.

Defenses

Most ICMP attacks are easy to defend against with just a modicum of paranoia. If a host is careful about checking that a message really does refer to a particular connection, most such attacks will not succeed. In the case of TCP, this includes verifying that the ICMP packet contains a plausible sequence number in the returned-packet portion. These checks are less applicable to UDP, though.

A defense against Redirect attacks merits additional attention, since such attacks can be more serious. Probably, the best option is to restrict route changes to the specified connection; the global routing table should not be modified in response to ICMP Redirect messages.[6]

Finally, it is worth considering whether ICMP Redirects are even useful in today's environment. They are only usable on local networks with more than one gateway to the outside world. But it is comparatively easy to maintain complete and correct local routing information. Redirect messages would be most useful from the core gateways to local exterior gateways, as that would allow such local gateways to have less than complete knowledge of the Internet; this use is disallowed, however.

Subnet Mask attacks can be blocked if the Reply packet is honored only at the appropriate time. In general, a host wants to see such a message only at boot time, and only if it had issued a query; a stale reply, or an unsolicited reply, should be rejected out of hand. There is little defense against a forged reply to a genuine Subnet Mask query, as a host that has sent such a query typically has few resources with which to validate the response. If the genuine response is not blocked by the intruder, though, the target will receive multiple replies; a check to ensure that all replies agree would guard against administrative errors as well.

> ICMP attacks against routing have never been very real. Anyone with the ability launch such an attack can use ARP-spoofing much more easily; that, in fact, has been done.
>
> Early RFCs suggested that routers could also listen to ICMP Redirect messages; a later document [44] permits routers to ignore such packets if directed to them. Routers, almost by definition, run routing protocols, which gives them a much better idea of topology; hosts, by contrast, should not listen to routing traffic. (This contradicts advice I gave in 1989.) That said, there is a current trend towards hosts that know more about the network topology.
>
> The paper mentions a number of denial of service attacks that could be launched by sending spurious ICMP error packets. Those became reasonably common in the early and mid-1990s. But as systems started complying with the advice in RFC 1122 [1], that trend died down; [1] mandates that most ICMP errors be treated as advisory messages, rather than fatal errors.

4. The "Authentication" Server

As an alternative to address-based authentication, some implementations use the *Authentication Server* [96] A server that wishes to know the identity of its client may contact the client host's Authentication Server.[7] and ask it for information about the user owning a particular connection. This method is inherently more secure than simple address-based authentication, as it uses a second TCP connection not under control of the attacker. It thus can defeat sequence number attacks and source routing attacks. There are certain risks, however.

The first, and most obvious, is that not all hosts are competent to run authentication servers. If the client host is not secure, it does not matter who the user is claimed to be; the answer cannot be trusted. Second, the authentication message itself can be compromised by routing table attacks. If RIP has been used to alter the target's idea of how to reach some host, the authentication query will rely on the same altered routing data. Finally, if the target host is down, a variant on the TCP sequence number attack may be used; after the server sends out a TCP open request to the presumed authentication server, the attacker can complete the open sequence and send a false reply. If the target runs a *netstat* server, this is even easier; as noted, *netstat* will often supply the necessary sequence numbers with no need to guess.

A less-obvious risk is that a fake authentication server can always reply "no". This constitutes a denial of service attack.

Defenses

A server that wishes to rely on another host's idea of a user should use a more secure means of validation, such as

6 This has other benefits as well, especially in environments where ICMP-initiated route changes are not timed out. The author has seen situations where RIP instability following a gateway crash has led to erroneous ICMP Redirect messages. These had the effect of permanently corrupting the routing tables on other hosts.

7 The Internet Activities Board does not currently recommend the Authentication Server for implementation [14]. However, the decision was not made because of security problems [80].

the Needham-Schroeder algorithm [75, 28, 76]. TCP by itself is inadequate.

> The original paper strongly suggested that the authentication server was a bad idea. Unfortunately, it has been modernized [98] and is still used today. Fortunately, its primary use is for auditing (especially of email), rather than authentication; even so, the weaknesses outlined here (as well as in the Security Considerations section of [98] remain. Indeed, I personally run a readily-available implementation that replies to all queries with the message "ident-is-a-completely-pointless-protocol-that-offers-no-security-or-traceability-at-all-so-take-this-and-log-it!"

5. Here be Dragons

Some protocols, while not inherently flawed, are nevertheless susceptible to abuse. A wise implementor would do well to take these problems into account when providing the service.

5.1. The "Finger" Service

Many systems implement a *finger* service [48]. This server will display useful information about users, such as their full names, phone numbers, office numbers, etc. Unfortunately, such data provides useful grist for the mill of a password cracker [46]. By running such a service, a system administrator is giving away this data.

> It is debatable whether or not this is an architectural problem or an implementation problem. The RFC never says precisely what information shoudl be returned, though the samples do show full names and a few phone numbers. The precise question is generally moot today for external attacks—firewalls will generally block the *finger* protocol—but modern Web servers often release the same sort of information. Search engines yield even more data. Is password-guessing still useful to attackers? Beyond question, yes; I've seen new implementations within the last few months.

5.2. Electronic Mail

Electronic mail is probably the most valuable service on the Internet. Nevertheless, it is quite vulnerable to misuse. As normally implemented [24, 82], the mail server provides no authentication mechanisms. This leaves the door wide open to faked messages. RFC 822 does support an *Encrypted* header line, but this is not widely used. (However, see RFC 1040 [60] for a discussion of a proposed new encryption standard for electronic mail.)

> Authenticating and encrypting email have become far more important today than in 1989. There is still no widely-deployed method of authenticating email; one is likely to be deployed in the near future, though arguably for the wrong reason. Spammers and "phishers" use fake email addresses; there are a number of proposals on the table to somehow authenticate the source of the email. But they won't work. The problem is that authentication proves an *identity*; it says nothing of whether or not that party is *authorized* to send you email. Most people are willing to accept email from anyone; spammers can and do claim to be sending email from sites like asdfghij.com, and they're right: that is their email address. What does authentication prove?
>
> It may slow down the phishers slightly, but only slightly. True, if email were authenticated they could no longer claim to be YourRealBank.com, but they could claim to be E-YourRealBank.com, YourRealBank-Onilne.com, www-YourRealBank.com, etc. (The very first phishing attempt I know of claimed to be from paypa1.com—and it was. Of course, lots of people read that as paypal.com.)
>
> Fake email has been used for other sinister purposes, such as stock market fraud: the perpetrator sent a message to an investor's newswire saying that wonderful things were happening to some company. Naturally, its stock went up—and he sold at the peak. Digitally-signed press releases would prevent that sort of thing—*if* the recipients checked the signatures, the certificates, etc., against some known-good values.
>
> Encrypting email is also useful in some situations, and there are a number of choices available [39, 5]. The practical problem is that the endpoints aren't secure.

5.2.1. The Post Office Protocol The *The Post Office Protocol* (POP) [15] allows a remote user to retrieve mail stored on a central server machine. Authentication is by means of

a single command containing both the user name and the password. However, combining the two on a single command mandates the use of conventional passwords. And such passwords are becoming less popular; they are too vulnerable to wire-tappers, intentional or accidental disclosure, etc.

As an alternative, many sites are adopting "one-time passwords".[8] With one-time passwords, the host and some device available to the user share a cryptographic key. The host issues a random challenge; both sides encrypt this number, and the user transmits it back to the host. Since the challenge is random, the reply is unique to that session, thereby defeating eavesdroppers. And since the user does not know the key—it is irretrievably stored in the device—the password cannot be given away without depriving the user of the ability to log in.

The newest version of POP [87] has split the user name and password into two commands, which is useful. However, it also defines an optional mechanism for preauthenticated connections, typically using Berkeley's mechanisms. Commendably, the security risks of this variant are mentioned explicitly in the document.

> POP3 [72] has gained in importance; it's the princple mechanism people use to retrieve email from servers. Simple passwords are still the most common authentication mechanism; while a variant that uses SSL encryption [85] is available, most people don't use it. Another mail retrieval protocol, IMAP4 [23], has similar security properties: encryption is available but largely unused.

5.2.2. PCMAIL The *PCMAIL* protocol [58] uses authentication mechanisms similar to those in POP2. In one major respect, PCMAIL is more dangerous: it supports a password-change command. This request requires that both the old and new passwords be transmitted unencrypted.

> This protocol is no longer used.

5.3. The Domain Name System

The *Domain Name System* (DNS) [67, 68] provides for a distributed database mapping host names to IP addresses. An intruder who interferes with the proper operation of the DNS can mount a variety of attacks, including denial of service and password collection. There are a number of vulnerabilities.

In some resolver implementations, it is possible to mount a sequence number attack against a particular user. When the target user attempts to connect to a remote machine, an attacker can generate a domain server response to the target's query. This requires knowing both the UDP port used by the client's resolver and the DNS sequence number used for the query. The latter is often quite easy to obtain, though, since some resolvers always start their sequence numbers with 0. And the former may be obtainable via *netstat* or some analogous host command.

A combined attack on the domain system and the routing mechanisms can be catastrophic. The intruder can intercept virtually all requests to translate names to IP addresses, and supply the address of a subverted machine instead; this would allow the intruder to spy on all traffic, and build a nice collection of passwords if desired.

For this reason, domain servers are high-value targets; a sufficiently determined attacker might find it useful to take over a server by other means, including subverting the machine one is on, or even physically interfering with its link to the Internet. There is no network defense against the former, which suggests that domain servers should only run on highly secure machines; the latter issue may be addressed by using authentication techniques on domain server responses.

The DNS, even when functioning correctly, can be used for some types of spying. The normal mode of operation of the DNS is to make specific queries, and receive specific responses. However, a *zone transfer* (AXFR) request exists that can be used to download an entire section of the database; by applying this recursively, a complete map of the name space can be produced. Such a database represents a potential security risk; if, for example, an intruder knows that a particular brand of host or operating system has a particular vulnerability, that database can be consulted to find all such targets. Other uses for such a database include espionage; the number and type of machines in a particular organization, for example, can give away valuable data about the size of the organization, and hence the resources committed to a particular project.

Fortunately, the domain system includes an error code for "refused"; an administrative prohibition against such zone transfers is explicitly recognized as a legitimate reason for refusal. This code should be employed for zone transfer requests from any host not known to be a legitimate secondary server. Unfortunately, there is no authentication mechanism provided in the AXFR request; source address authentication is the best that can be done.

Recently, a compatible authentication extension to the DNS has been devised at M.I.T. The Hesiod name server [36] uses Kerberos [99] tickets to authenticate queries and responses. The *additional information* section of the query carries an encrypted ticket, which includes a session key; this key, known only to Hesiod and the client, is used to compute a cryptographic checksum of the both the query

8 One-time passwords were apparently first used for military IFF (Identification Friend or Foe) systems [29].

and the response. These checksums are also sent in the additional information field.

> The DNS remains a crucial weak spot in the Internet [89]. Other attacks have been found in the intervening years.
>
> I was told of the first shortly after this paper was published. Though I had railed against *address*-based authentication, in fact the *r*-utilities do *name*-based authentication: they look up the hostname corresponding to the originator's IP address, and use it to make an authentication decision. But given the way hostname lookups work with the DNS, the owner of the IP address block involved controls what names are returned. Thus, if I own 192.0.2.0/24, I can create a PTR record for, say, 192.0.2.1; this record would identify the host as YourTrustedHost.com. There are no violations of the DNS or *r*-utility protocols involved here, nor any tricky address spoofing. Instead, the attacker simply needs to lie.
>
> The lesson is clear: when building security systems, understand exactly what elements are being trusted. In this case, not only was trust residing in the DNS (itself a problem), trust was residing in a piece of the DNS controlled by the enemy. The fix (against this particular attack) was simple: use the returned hostname and look up its IP address, and verify that it matches the address used in the connection. Assuming that you control the relevant portion of the DNS tree, this foils the attack. Of course, and as noted above, users are able to extend trust, possibly to a part of the DNS tree not controlled by someone trustworthy.
>
> Even without ill-advised trust, you're relying on the rest of the DNS. Cache contamination attacks [9] can create false entries. For that matter, the technical and administrative procedures used to update zones such as .com can be subverted; that's happened, too.
>
> There are defenses against some of the DNS attacks. Filtering [20] can prevent certain kinds of cache contamination attacks. Many popular implementations have been hardened against sequence number attacks [102]. A comprehensive discussion of DNS-related threats can be found in [8].
>
> I was wrong to laud Hesiod as a solution to DNS security problems. Hesiod protects the transmission and response; it does not protect the data, and the responding name server might itself have been deceived. The right solution is DNSsec [37], which provides for digitally-signed resource records. DNSsec deployment has been very slow, partially because it was so hard to get many of the design details right; see, for example, [47, 4, 105],

5.4. The File Transfer Protocol

The *File Transfer Protocol* (FTP) [83] itself is not flawed. However, a few aspects of the implementation merit some care.

5.4.1. FTP Authentication FTP relies on a login and password combination for authentication. As noted, simple passwords are increasingly seen as inadequate; more and more sites are adopting one-time passwords. Nothing in the FTP specification precludes such an authentication method. It is vital, however, that the "331" response to a *USER* subcommand be displayed to the user; this message would presumably contain the challenge. An FTP implementation that concealed this response could not be used in this mode; if such implementations are (or become) common, it may be necessary to use a new reply code to indicate that the user must see the content of the challenge.

5.4.2. Anonymous FTP A second problem area is "anonymous FTP". While not required by the FTP specification, anonymous FTP is a treasured part of the oral tradition of the Internet. Nevertheless, it should be implemented with care.

One part of the problem is the implementation technique chosen. Some implementations of FTP require creation of a partial replica of the directory tree; care must be taken to ensure that these files are not subject to compromise. Nor should they contain any sensitive information, such as encrypted passwords.

The second problem is that anonymous FTP is truly anonymous; there is no record of who has requested what information. Mail-based servers will provide that data; they also provide useful techniques for load-limiting,[9] background transfers, etc.

> FTP is hard to secure. It's reasonably straight-forward to encrypt the control channel; protecting the data channels is harder, because they're dynamic. Worse yet, a malicious FTP client can use the data channels

9 Recently, a host was temporarily rendered unusable by massive numbers of FTP requests for a popular technical report. If this were deliberate, it would be considered a successful denial of service attack.

to cause an innocent FTP server to attack a third host, in what's known as a *bounce attack*. A security analysis of FTP (including details of the bounce attack) can be found in [7]; [51] describes cryptogprahic protection for FTP.

5.5. Simple Network Management Protocol

The *Simple Network Management Protocol* (SNMP) [17] has recently been defined to aid in network management. Clearly, access to such a resource must be heavily protected. The RFC states this, but also allows for a null authentication service; this is a bad idea. Even a "read-only" mode is dangerous; it may expose the target host to *netstat*-type attacks if the particular Management Information Base (MIB) [62] used includes sequence numbers. (The current standardized version does not; however, the MIB is explicitly declared to be extensible.)

SNMP authentication, as originally defined, boils down to a simple plaintext password known as the *community string*. All the usual problems with plaintext passwords apply, including eavesdropping and guessability. SNMPv3 [18] defines a *User-based Security Model* [92] with cryptographic authentication. In addition, new MIBs are carefully scrutinized for security-sensitive elements: one proposal that would have put TCP sequence numbers into the MIB was caught.

5.6. Remote Booting

Two sets of protocols are used today to boot diskless workstations and gateways, *Reverse ARP* (RARP) [43] with the *Trivial File Transfer Protocol* (TFTP) [94] and BOOTP [25] with TFTP. A system being booted is a tempting target; if one can subvert the boot process, a new kernel with altered protection mechanisms can be substituted. RARP-based booting is riskier because it relies on Ethernet-like networks, with all the vulnerabilities adhering thereto. One can achieve a modest improvement in security by ensuring that the booting machine uses a random number for its UDP source port; otherwise, an attacker can impersonate the server and send false DATA packets.

BOOTP adds an additional layer of security by including a 4-byte random *transaction id*. This prevents an attacker from generating false replies to a workstation known to be rebooting. It is vital that these numbers indeed be random; this can be difficult in a system that is freshly powered up, and hence with little or no unpredictable state. Care should be taken when booting through gateways; the more networks traversed, the greater the opportunity for impersonation.

The greatest measure of protection is that normally, the attacker has only a single chance; a system being booted does not stay in that state. If, however, communications between the client and the standard server may be interrupted, larger-scale attacks may be mounted.

A newer boot-time protocol, DHCP [34] is even more important. It provides hosts with IP addresses, DNS servers, default router, and more. Furthermore, DHCP queries can happen with some frequency, if the lease time of the address is short; this gives an attacker many more opportunities to do mischief.

There is a DHCP authentication option [3], but it is little-used. One reason is that anyone who can mount a DHCP attack can launch a local network attack just as easily; merely protecting DHCP does little to protect the client.

6. Trivial Attacks

A few attacks are almost too trivial to mention; nevertheless, completeness demands that they at least be noted.

6.1. Vulnerability of the Local Network

Some local-area networks, notably the Ethernet networks, are extremely vulnerable to eavesdropping and host-spoofing. If such networks are used, physical access must be strictly controlled. It is also unwise to trust any hosts on such networks if any machine on the network is accessible to untrusted personnel, unless authentication servers are used.

If the local network uses the Address Resolution Protocol (ARP) [78] more subtle forms of host-spoofing are possible. In particular, it becomes trivial to intercept, modify, and forward packets, rather than just taking over the host's role or simply spying on all traffic.

It is possible to launch denial of service attacks by triggering *broadcast storms*. There are a variety of ways to do this; it is quite easy if most or all of the hosts on the network are acting as gateways. The attacker can broadcast a packet destined for a non-existent IP address. Each host, upon receiving it, will attempt to forward it to the proper destination. This alone will represent a significant amount of traffic, as each host will generate a broadcast ARP query for the destination. The attacker can follow up by broadcasting an ARP reply claiming that the broadcast Ethernet address is the proper way to reach that destination. Each suspectible host will then not only resend the bogus packet, it

will also receive many more copies of it from the other susceptible hosts on the network.

ARP attacks are easy to launch and hard to spot. End-to-end encryption can prevent the worst problems, but they remain a potent vehicle for denial of service attacks. This is being addressed for IPv6 with the SEND enhancements to IPv6 Neighbor Discovery, its version of ARP.

There are two modern venues where ARP attacks are particularly nasty. One is on wireless networks, especially in public hotspots. Another is using ARP to defeat the presumed security properties of Ethernet switches. Some administrators assume that using a switch prevents classical eavesdropping; it does, but ARP can be used to redirect the traffic despite that.

Broadcast storms have been launched maliciously, but in a more dangerous fashion. The attacker sends an ICMP Echo Request packet to the broadcast address of some local network; *all* hosts on that network send a reply to the originator of the ICMP message. Of course, the source IP address in that packet isn't that of the attacker; instead, it's the address of the victim, who gets bombarded with enough Echo Reply messages to clog the link. *Smurf attacks* (see CERT Advisory CA-1998-01) have become relatively rare since the default router configurations were changed to disable directed broadcasts [91].

6.2. The Trivial File Transfer Protocol

TFTP [94] permits file transfers without any attempt at authentication. Thus, any publicly-readable file in the entire universe is accessible. It is the responsibility of the implementor and/or the system administrator to make that universe as small as possible.

6.3. Reserved Ports

Berkeley-derived TCPs and UDPs have the notion of a "privileged port". That is, port numbers lower than 1024 may only be allocated to privileged processes. This restriction is used as part of the authentication mechanism. However, neither the TCP nor the UDP specifications contain any such concept, nor is such a concept even meaningful on a single-user computer. Administrators should never rely on the Berkeley authentication schemes when talking to such machines.

Privileged ports are a bad idea, but they could have been even worse. When FTP bounce attacks are launched, the source port is 21, in the privileged range. In other words, bounce attacks could have been used to attack *rlogin* and *rsh* servers. Fortunately, out of a nagging sense of unease, those servers only accepted connections coming from ports in the range 512–1023.

7. Comprehensive Defenses

Thus far, we have described defenses against a variety of individual attacks. Several techniques are broad-spectrum defenses; they may be employed to guard against not only these attacks, but many others as well.

7.1. Authentication

Many of the intrusions described above succeed only because the target host uses the IP source address for authentication, and assumes it to be genuine. Unfortunately, there are sufficiently many ways to spoof this address that such techniques are all but worthless. Put another way, source address authentication is the equivalent of a file cabinet secured with an S100 lock; it may reduce the temptation level for more-or-less honest passers-by, but will do little or nothing to deter anyone even slightly serious about gaining entry.

Some form of cryptographic authentication is needed. There are several possible approaches. Perhaps the best-known is the Needham-Schroeder algorithm [75, 28, 76]. It relies on each host sharing a key with an authentication server; a host wishing to establish a connection obtains a session key from the authentication server and passes a sealed version along to the destination. At the conclusion of the dialog, each side is convinced of the identity of the other. Versions of the algorithm exist for both private-key and public-key [30] cryptosystems.

How do these schemes fit together with TCP/IP? One answer is obvious: with them, preauthenticated connections can be implemented safely; without them, they are quite risky. A second answer is that the DNS provides an ideal base for authentication systems, as it already incorporates the necessary name structure, redundancy, etc. To be sure, key distribution responses must be authenticated and/or encrypted; as noted, the former seems to be necessary in any event.

In some environments, care must be taken to use the session key to encrypt the entire conversation; if this is not done, an attacker can take over a connection via the mechanisms described earlier.

> Doing cryptography properly is a lot harder than I made it seem. Indeed, the Needham-Schroeder scheme—the oldest cryptographic protocol in the open literature—was found in 1996 to be vulnerable to a new flaw [61]. That variant, expressed in modern notation, is only three lines long...

7.2. Encryption

Suitable encryption can defend against most of the attacks outlined above. But encryption devices are expensive, often slow, hard to administer, and uncommon in the civilian sector. There are different ways to apply encryption; each has its strengths and weaknesses. A comprehensive treatment of encryption is beyond the scope of this paper; interested readers should consult Voydock and Kent [103] or Davies and Price [26]

Link-level encryption—encrypting each packet as it leaves the host computer—is an excellent method of guarding against disclosure of information. It also works well against physical intrusions; an attacker who tapped in to an Ethernet cable, for example, would not be able to inject spurious packets. Similarly, an intruder who cut the line to a name server would not be able to impersonate it. The number of entities that share a given key determines the security of the network; typically, a key distribution center will allocate keys to each pair of communicating hosts.

Link-level encryption has some weaknesses, however. Broadcast packets are difficult to secure; in the absence of fast public-key cryptosystems, the ability to decode an encrypted broadcast implies the ability to send such a broadcast, impersonating any host on the network. Furthermore, link-level encryption, by definition, is not end-to-end; security of a conversation across gateways implies trust in the gateways and assurance that the full concatenated internet is similarly protected. (This latter constraint may be enforced administratively, as is done in the military sector.) If such constraints are not met, tactics such as source-routing attacks or RIP-spoofing may be employed. Paranoid gateways can be deployed at the entrance to security domains; these might, for example, block incoming RIP packets or source-routed packets.

> As pointed out in [54], I had Ethernet link encryptors in mind. Link encryptors on point-to-point links are quite different; they protect traffic down to the bit level, sometimes hiding even the HDLC framing.

Many portions of the DARPA Internet employ forms of link encryption. All Defense Data Network (DDN) IMP-to-IMP trunks use DES encryption, even for non-classified traffic; classified lines use more secure cryptosystems [27]. These, however, are point-to-point lines, which are comparatively easy to protect.

A multi-point link encryption device for TCP/IP is the *Blacker Front End* (BFE) [40]. The BFE looks to the host like an X.25 DDN interface, and sits between the host and the actual DDN line. When it receives a call request packet specifying a new destination, it contacts an Access Control Center (ACC) for permission, and a Key Distribution Center (KDC) for cryptographic keys. If the local host is denied permission to talk to the remote host, an appropriate diagnostic code is returned. A special "Emergency Mode" is available for communications to a restricted set of destinations at times when the link to the KDC or ACC is not working.

The permission-checking can, to some extent, protect against the DNS attacks described earlier. Even if a host has been mislead about the proper IP address for a particular destination, the BFE will ensure that a totally unauthorized host does not receive sensitive data. That is, assume that a host wishes to send Top Secret data to some host *foo*. A DNS attack might mislead the host into connecting to penetrated host 4.0.0.4, rather than 1.0.0.1. If 4.0.0.4 is not cleared for Top Secret material, or is not allowed communications with the local host, the connection attempt will fail. To be sure, a denial of service attack has taken place; this, in the military world, is far less serious than information loss.

The BFE also translates the original ("Red") IP address to an encrypted ("Black") address, using a translation table supplied by the ACC. This is done to foil traffic analysis techniques, the bane of all multi-point link encryption schemes.

> I got a lot wrong here (see [54] for some details), partly because of my lack of experience with crypto at the time and partly because of the sketchy information publicly available on Blacker. With the benefit of hindsight (and a lot more experience with cryptography), I'd call the BFE a network-layer encryptor for non-broadcast multiple access networks, rather than a multi-point link-level encryptor. But Blacker was obsolescent even as I wrote the original paper; SP3—the ancestor of IPsec [57] —was being defined as part of the Secure Data Network System. Regardless, the BFE (and SP3 and IPsec) all create virtual private networks, with their own address spaces; while there is some protection against traffic analysis, these technologies do not prevent an adversary from noticing which protected networks are talking to which other protected networks.

End-to-end encryption, above the TCP level, may be used to secure any conversation, regardless of the number of hops or the quality of the links. This is probably appropriate for centralized network management applications, or other point-to-point transfers. Key distribution and management is a greater problem, since there are more pairs of correspondents involved. Furthermore, since encryption and decryption are done before initiation or after termination of the TCP processing, host-level software must arrange for the translation; this implies extra overhead for each such conversation.[10]

End-to-end encryption is vulnerable to denial of service attacks, since fraudulently-injected packets can pass the TCP checksum tests and make it to the application. A combination of end-to-end encryption and link-level encryption can be employed to guard against this. An intriguing alternative would be to encrypt the data portion of the TCP segment, but not the header; the TCP checksum would be calculated on the cleartext, and hence would detect spurious packets. Unfortunately, such a change would be incompatible with other implementations of TCP, and could not be done transparently at application level.

> I wrote "link-level encryption" here; it should have been "network-level". But there's a more subtle problem: the TCP checksum is not cryptographically strong; it's pretty easy to launch a variety of attacks against such a weak integrity protection mechanism. See, for example, [11].

Regardless of the method used, a major benefit of encrypted communications is the implied authentication they provide. If one assumes that the key distribution center is secure, and the key distribution protocols are adequate, the very ability to communicate carries with it a strong assurance that one can trust the source host's IP address for identification.

This implied authentication can be especially important in high-threat situations. A routing attack can be used to "take over" an existing connection; the intruder can effectively cut the connection at the subverted machine, send dangerous commands to the far end, and all the while translate sequence numbers on packets passed through so as to disguise the intrusion.

> Today's network-level encryptors would foil this; they protect the TCP (and sometimes IP) headers as well as the payload. I suspect that the BFE would as well.

10 We are assuming that TCP is handled by the host, and not by a front-end processor.

It should be noted, of course, that any of these encryption schemes provide privacy. Often that is the primary goal of such systems.

7.3. Trusted Systems

Given that TCP/IP is a Defense Department protocol suite, it is worth asking to what extent the Orange Book [31] and Red Book [33] criteria would protect a host from the attacks described above. That is, suppose that a target host (and the gateways!) were rated B1 or higher. Could these attacks succeed? The answer is a complex one, and depends on the assumptions we are willing to make. In general, hosts and routers rated at B2 or higher are immune to the attacks described here, while C2-level systems are susceptible. B1-level systems are vulnerable to some of these attacks, but not all.

In order to understand how TCP/IP is used in secure environments, a brief tutorial on the military security model is necessary. All *objects* in the computer system, such as files or network channels, and all data exported from them, must have a *label* indicating the sensitivity of the information in them. This label includes hierarchical components (i.e., Confidential, Secret, and Top Secret) and non-hierarchical components. *Subjects* —i.e., processes within the computer system—are similarly labeled. A subject may *read* an object if its label has a higher or equal hierarchical level and if all of the object's non-hierarchical components are included in the subject's label. In other words, the process must have sufficient clearance for the information in a file. Similarly, a subject may write to an object if the object has a *higher* or equal level and the object's non-hierarchical components include all of those in the subject's level. That is, the sensitivity level of the file must be at least as high as that of the process. If it were not, a program with a high clearance could write classified data to a file that is readable by a process with a low security clearance.

A corollary to this is that for read/write access to any file, its security label must exactly match that of the process. The same applies to any form of bidirectional interprocess communication (i.e., a TCP virtual circuit): both ends must have identical labels.

We can now see how to apply this model to the TCP/IP protocol suite. When a process creates a TCP connection, that connection is given the process's label. This label is encoded in the IP security option. The remote TCP must ensure that the label on received packets matches that of the receiving process. Servers awaiting connections may be eligible to run at multiple levels; when the connection is instantiated, however, the process must be forced to the level of the connection request packet.

IP also makes use of the security option [97]. A packet may not be sent over a link with a lower clearance level.

If a link is rated for Secret traffic, it may carry Unclassified or Confidential traffic, but it may not carry Top Secret data. Thus, the security option constrains routing decisions. The security level of a link depends on its inherent characteristics, the strength of any encryption algorithms used, the security levels of the hosts on that network, and even the location of the facility. For example, an Ethernet cable located in a submarine is much more secure than if the same cable were running through a dormitory room in a university.

Several points follow from these constraints. First, TCP-level attacks can only achieve penetration at the level of the attacker. That is, an attacker at the Unclassified level could only achieve Unclassified privileges on the target system, regardless of which network attack was used.[11] Incoming packets with an invalid security marking would be rejected by the gateways.

Attacks based on any form of source-address authentication should be rejected as well. The Orange Book requires that systems provide secure means of identification and authentication; as we have shown, simple reliance on the IP address is not adequate. As of the B1 level, authentication information must be protected by cryptographic checksums when transmitted from machine to machine.[12]

The *authentication* server is still problematic; it can be spoofed by a sequence number attack, especially if *netstat* is available. This sort of attack could easily be combined with source routing for full interactive access. Again, cryptographic checksums would add significant strength.

B1-level systems are not automatically immune from routing attacks; RIP-spoofing could corrupt their routing tables just as easily. As seen, that would allow an intruder to capture passwords, perhaps even some used on other trusted systems. To be sure, the initial penetration is still restricted by the security labelling, but that may not block future logins captured by these means.

Routing attacks can also be used for denial of service. Specifically, if the route to a secure destination is changed to require use of an insecure link, the two hosts will not be able to communicate. This change would probably be detected rather quickly, though, since the gateway that noticed the misrouted packet would flag it as a security problem.

At the B2 level, secure transmission of routing control information is required. Similar requirements apply to other network control information, such as ICMP packets.

Several attacks we have described rely on data derived from "information servers", such as *netstat* and *finger*. While these, if carefully done, may not represent a direct penetration threat in the civilian sense, they are often seen to represent a *covert channel* that may be used to leak information. Thus, many B-division systems do not implement such servers.

In a practical sense, some of the technical features we have described may not apply in the military world. Administrative rules [32] tend to prohibit risky sorts of interconnections; uncleared personnel are not likely to have even indirect access to systems containing Top Secret data. Such rules are, most likely, an accurate commentary on anyone's ability to validate any computer system of non-trivial size.

> This is an odd section for this paper, in that it attempts a very brief look at a completely different technology architecture: Orange Book-style secure systems. Worse yet, the Orange Book prescribes results, not methods; as such, it is difficult to find a precise match between the attacks I described and abstract mechanisms.
>
> In any event, the question is moot today. The Orange Book and its multi-hued kin have been replaced by the Common Criteria [19], an international effort at defining secure systems. In most ways, the Common Criteria is even more abstract than its DoD predecessor; one could not make meaningful statements without at least specifying what protection profile one was interested in. (Apart from that, one can make a strong argument that that entire approach to system security is flawed [89], but such a discussion is out of scope for this paper.)
>
> Two technical points are worth noting. First, routing attacks could be mitigated by maintenance of separate routing tables (by multi-level secure routers) for different security classifications. Second, exactly what forms of authentication are acceptable in any situation would depend critically on detailed knowledge of exactly what sorts of hosts were connected to what sorts of network. In other words, eavesdropping may or may not be a concern.

8. Conclusions

Several points are immediately obvious from this analysis. The first, surely, is that in general, relying on the IP source address for authentication is extremely dangerous.[13]

11 We are assuming, of course, that the penetrated system does not have bugs of its own that would allow further access.

12 More precisely, user identification information must be protected to an equal extent with data sensitivity labels. Under certain circumstances, described in the Red Book, cryptographic checks may be omitted. In general, though, they are required.

13 There are some exceptions to this rule. If the entire network, and all of its components (hosts, gateways, cables, etc.) are physically protected,

Fortunately, the Internet community is starting to accept this on more than an intellectual level. The Berkeley manuals [22] have always stated that the authentication protocol was very weak, but it is only recently that serious attempts (i.e., Kerberos [99] and SunOS 4.0's DES authentication mode [101]) have been made to correct the problem. Kerberos and SunOS 4.0 have their weaknesses, but both are far better than their predecessor. More recently, an extension to the *Network Time Protocol* (NTP) [66] has been proposed that includes a cryptographic checksum [63].

A second broad class of problems is sequence number attacks. If a protocol depends on sequence numbers—and most do—it is vital that they be chosen unpredictably. It is worth considerable effort to ensure that these numbers are not knowable even to other users on the same system.

We may generalize this by by stating that hosts should not give away knowledge gratuitously. A *finger* server, for example, would be much safer if it only supplied information about a known user, rather than supplying information about everyone logged on. Even then, some censorship might be appropriate; a refusal to supply the last login date and other sensitive information would be appropriate if the account was not used recently. (Never-used accounts often have simple default passwords. Infrequently-used accounts are often set up less carefully by the owner.) We have also seen how *netstat* may be abused; indeed, the combination of *netstat* with the *authentication* server is the single strongest attack using the standardized Internet protocols.

Finally, network control mechanisms are dangerous, and must be carefully guarded. Static routes are not feasible in a large-scale network, but intelligent use of default routes and verifiable point-to-point routing protocols (i.e., EGP) are far less vulnerable than broadcast-based routing.

9. Acknowledgments

Dave Presotto, Bob Gilligan, Gene Tsudik, and especially Deborah Estrin made a number of useful suggestions and corrections to a draft of this paper.

10. Retrospective Conclusions

The Internet of 1989 was a much simpler—and much friendlier—place than it is today. Most of the protocols I looked at were comparatively simple client-server protocols; today's multi-party protocols—SIP [88], Diameter [16], various peer-to-peer protocols, etc.—are much harder to analyze. Often, the crucial question is not authentication but *authorization*: how do you know if a certain party is permitted to perform a certain action?

The overall trend has been good. The Internet Engineering Task Force (IETF) will not standardize protocols where the only mandatory-to-implement form of authentication is plaintext passwords; address-based authentication is acceptable only in very restricted circumstances. There are standardized and/or widely deployed cryptographic protocols for remote login, sending and receiving email, Web browsing, etc. As discussed earlier, outing is the major exception; operationally, it is still not securable.

Most of the security problems we encounter on the Internet today are due to buggy code (including, of course, buggy code in cryptographic modules). To some extent, of course, that's because there are so many bugs; why launch a difficult attack when there's so much low-hanging fruit available?

Password-guessing is still a common attack; indeed, a new wave of password-guessing has recently been observed against *ssh* [106], a cryptographically-protected replacement for the *r*-utilities.

There is no need to belabor the earlier conclusions that predictable sequence numbers and address-based authentication are bad. This is now well-accepted; for example, the specification for a new transport protocol, SCTP [100], mandates strong random number generation [6] for its analogous fields. But how does one design a secure protocol?

One answer is to look at data flow of the protocol. On what elements does authentication depend? Can an attacker tamper with or mimic those elements? What are the guarantees, especially the security guarantees, of each element? Naturally, the powers of the attacker must be taken into account. Seen from this perspective, the flaws in the *r*-utilities are clear.

One more point should be mentioned: it is often the availability of auxiliary data that makes attacks possible. An attacker who can create just one TCP connection cannot guess the correct sequence number. This is often a fruitful mechanism for blocking (or at least detecting) attacks.

and if all of the operating systems are sufficiently secure, there would seem to be little risk.

References

[1] R. Braden, editor. Requirements for Internet hosts - communication layers. RFC 1122, Internet Engineering Task Force, Oct. 1989.

[2] Shadow Hawk gets prison term, February 1989. Phrack World News XXIV, file 12 of 13.

[3] R. E. Droms and W. Arbaugh, editors. Authentication for DHCP messages. RFC 3118, Internet Engineering Task Force, June 2001.

[4] J. Schlyter, editor. DNS security (DNSSEC) NextSECure (NSEC) RDATA format. RFC 3845, August 2004.

[5] B. Ramsdell, editor. Secure/multipurpose internet mail extensions (S/MIME) version 3.1 message specification. RFC 3851, July 2004.

[6] D. E. 3rd, S. D. Crocker, and J. Schiller. Randomness recommendations for security. RFC 1750, Internet Engineering Task Force, Dec. 1994.

[7] M. Allman and S. Ostermann. FTP security considerations. RFC 2577, Internet Engineering Task Force, May 1999.

[8] D. Atkins and R. Austein. Threat analysis of the Domain Name System (DNS). RFC 3833, August 2004.

[9] S. M. Bellovin. Using the domain name system for system break-ins. In *Proceedings of the Fifth Usenix Unix Security Symposium*, pages 199–208, Salt Lake City, UT, June 1995.

[10] S. M. Bellovin. Defending against sequence number attacks. RFC 1948, Internet Engineering Task Force, May 1996.

[11] S. M. Bellovin. Problem areas for the IP security protocols. In *Proceedings of the Sixth Usenix Unix Security Symposium*, pages 205–214, July 1996.

[12] M. Bishop. An application of a fast data encryption standard implementation. Technical Report PCS-TR88-138, Department of Mathematics and Computer Science, Dartmouth College, Hanover, NH, 1988.

[13] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, November 1984.

[14] I. A. Board. IAB official protocol standards. RFC 1083, Internet Engineering Task Force, Dec. 1988.

[15] M. Butler, J. B. Postel, D. Chase, J. Goldberger, and J. F. Reynolds. Post office protocol: Version 2. RFC 937, Internet Engineering Task Force, Feb. 1985.

[16] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. Diameter base protocol. RFC 3588, Internet Engineering Task Force, Sept. 2003.

[17] J. D. Case, M. S. Fedor, M. L. Schoffstall, and J. R. Davin. Simple network management protocol. RFC 1067, Internet Engineering Task Force, Aug. 1988.

[18] J. D. Case, R. Mundy, D. Partain, and B. Stewart. Introduction and applicability statements for internet-standard management framework. RFC 3410, Internet Engineering Task Force, Dec. 2002.

[19] Common criteria for information technology security evaluation, August 1999. Version 2.1.

[20] B. Cheswick and S. M. Bellovin. A DNS filter and switch for packet-filtering gateways. In *Proceedings of the Sixth Usenix Unix Security Symposium*, pages 15–19, San Jose, CA, 1996.

[21] D. E. Comer. *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, volume I. Prentice-Hall, Englewood Cliffs, NJ, second edition, 1991.

[22] Computer Systems Research Group. *UNIX User's Reference Manual, 4.3 Berkeley Software Distrubtion, Virtual Vax-11 Version*. Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1986.

[23] M. R. Crispin. INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1. RFC 3501, Internet Engineering Task Force, Mar. 2003.

[24] D. H. Crocker. Standard for the format of ARPA Internet text messages. RFC 822, Internet Engineering Task Force, Aug. 1982.

[25] W. J. Croft and J. Gilmore. Bootstrap protocol. RFC 951, Internet Engineering Task Force, Sept. 1985.

[26] D. W. Davies and W. L. Price. *Security for Computer Networks*. John Wiley & Sons, second edition, 1989.

[27] Defense Commuications Agency. Defense data network subscriber security guide, 1983.

[28] D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, August 1981.

[29] W. Diffie. The first ten years of public key cryptography. *Proceedings of the IEEE*, 76(5):560–577, May 1988.

[30] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-11:644–654, November 1976.

[31] DoD trusted computer system evaluation criteria. DoD 5200.28-STD, DoD Computer Security Center, 1985.

[32] Technical rationale behind CSC-STD-003-83: Computer security requirements. DoD CSC-STD-004-85, DoD Computer Security Center, 1985.

[33] Trusted network interpretation of the "trusted computer system evaluation criteria". DoD NCSC-TG-005, DoD Computer Security Center, 1987.

[34] R. E. Droms. Dynamic host configuration protocol. RFC 2131, Internet Engineering Task Force, Mar. 1997.

[35] M. Dworkin. Recommendation for block cipher modes of operation: Methods and techniques. National Institute of Standards and Technology, December 2001. Federal Information Processing Standards Publication 81.

[36] S. Dyer. Hesiod. In *Proceedings of the Winter Usenix Conference*, Dallas, TX, 1988.

[37] D. Eastlake 3rd. Domain name system security extensions. RFC 2535, Internet Engineering Task Force, Mar. 1999.

[38] M. W. Eichin and J. A. Rochlis. With microscope and tweezers: An analysis of the Internet virus of november 1988. In *Proc. IEEE Symposium on Research in Security and Privacy*, pages 326–345, Oakland, CA, May 1989.

[39] M. Elkins. MIME security with pretty good privacy (PGP). RFC 2015, Internet Engineering Task Force, Oct. 1996.

[40] E. Feinler, O. Jacobsen, M. Stahl, and C. Ward, editors. *Blacker Front End Interface Control Document*. In Feinler et al. [41], 1985.

[41] E. Feinler, O. Jacobsen, M. Stahl, and C. Ward, editors. *DDN Protocol Handbook*. SRI International, 1985.

[42] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. RFC 2827, Internet Engineering Task Force, May 2000.

[43] R. Finlayson, T. P. Mann, J. C. Mogul, and M. M. Theimer. Reverse address resolution protocol. RFC 903, Internet Engineering Task Force, June 1984.

[44] E. Gerich. Unique addresses are good. RFC 1814, Internet Engineering Task Force, June 1995.

[45] V. Gill, J. Heasley, and D. Meyerx. The generalized TTL security mechanism (GTSM). RFC 3682, February 2004.

[46] F. T. Grampp and R. H. Morris. Unix operating system security. *AT&T Bell Laboratories Technical Journal*, 63(8, Part 2):1649–1672, October 1984.

[47] O. Gudmundsson. Delegation signer (DS) resource record (RR). RFC 3658, December 2003.

[48] K. Harrenstien. NAME/FINGER protocol. RFC 742, Internet Engineering Task Force, Dec. 1977.

[49] C. Hedrick. Routing information protocol. RFC 1058, Internet Engineering Task Force, June 1988.

[50] A. Heffernan. Protection of BGP sessions via the TCP MD5 signature option. RFC 2385, Internet Engineering Task Force, Aug. 1998.

[51] M. Horowitz and S. J. Lunt. FTP security extensions. RFC 2228, Internet Engineering Task Force, Oct. 1997.

[52] V. Jacobson, R. Braden, and L. Zhang. TCP extension for high-speed paths. RFC 1185, Internet Engineering Task Force, Oct. 1990.

[53] L. Joncheray. A simple active attack against TCP. In *Proceedings of the Fifth Usenix Unix Security Symposium*, Salt Lake City, UT, 1995.

[54] S. Kent. Comments on "Security problems in the TCP/IP protocol suite. 19(3):1–20, July 1989.

[55] S. Kent, C. Lynn, J. Mikkelson, and K. Seo. Secure border gateway protocol (S-BGP) – real world performance and deployment issues. In *Proceedings of the IEEE Network and Distributed System Security Symposium*, February 2000.

[56] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (Secure-BGP). *IEEE Journal on Selected Areas in Communications*, 18(4):582–592, April 2000.

[57] S. A. Kent and R. Atkinson. Security architecture for the Internet protocol. RFC 2401, Internet Engineering Task Force, Nov. 1998.

[58] M. L. Lambert. PCMAIL: a distributed mail system for personal computers. RFC 1056, Internet Engineering Task Force, June 1988.

[59] M. Leech. Key management considerations for the TCP MD5 signature option. RFC 3562, Internet Engineering Task Force, July 2003.

[60] J. R. Linn. Privacy enhancement for Internet electronic mail: Part I: message encipherment and authentication procedures. RFC 1040, Internet Engineering Task Force, Jan. 1988.

[61] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055, pages 147–166. Springer-Verlag, Berlin Germany, 1996.

[62] K. McCloghrie and M. T. Rose. Management information base for network management of TCP/IP-based internets. RFC 1066, Internet Engineering Task Force, Aug. 1988.

[63] D. Mills. Mail list message <8901192354.aa03743@huey.udel.edu>, January 19 1989.

[64] D. L. Mills. Internet delay experiments. RFC 889, Internet Engineering Task Force, Dec. 1983.

[65] D. L. Mills. Exterior gateway protocol formal specification. RFC 904, Internet Engineering Task Force, Apr. 1984.

[66] D. L. Mills. Network time protocol (version 1) specification and implementation. RFC 1059, Internet Engineering Task Force, July 1988.

[67] P. V. Mockapetris. Domain names - concepts and facilities. RFC 1034, Internet Engineering Task Force, Nov. 1987.

[68] P. V. Mockapetris. Domain names - implementation and specification. RFC 1035, Internet Engineering Task Force, Nov. 1987.

[69] J. C. Mogul. Simple and flexible datagram access controls for unix-based gateways. In *USENIX Conference Proceedings*, pages 203–221, Baltimore, MD, Summer 1989.

[70] R. T. Morris. A weakness in the 4.2BSD unix TCP/IP software. Computing Science Technical Report 117, AT&T Bell Laboratories, Murray Hill, NJ, February 1985.

[71] S. Murphy, M. Badger, and B. Wellington. OSPF with digital signatures. RFC 2154, Internet Engineering Task Force, June 1997.

[72] J. Myers and M. P. Rose. Post office protocol - version 3. RFC 1939, Internet Engineering Task Force, May 1996.

[73] NBS. Data encryption standard, January 1977. Federal Information Processing Standards Publication 46.

[74] NBS. DES modes of operation, December 1980. Federal Information Processing Standards Publication 81.

[75] R. M. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[76] R. M. Needham and M. Schroeder. Authentication revisited. *Operating Systems Review*, 21(1):7, January 1987.

[77] R. Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, M.I.T., 1988.

[78] D. C. Plummer. Ethernet address resolution protocol: Or converting network protocol addresses to 48.bit ethernet address for transmission on ethernet hardware. RFC 826, Internet Engineering Task Force, Nov. 1982.

[79] J. Postel. Internet control message protocol. RFC 792, Internet Engineering Task Force, Sept. 1981.

[80] J. Postel, 1989. private communication.

[81] J. B. Postel. Internet protocol. RFC 791, Internet Engineering Task Force, Sept. 1981.

[82] J. B. Postel. Simple mail transfer protocol. RFC 821, Internet Engineering Task Force, Aug. 1982.

[83] J. B. Postel and J. F. Reynolds. File transfer protocol. RFC 959, Internet Engineering Task Force, Oct. 1985.

[84] Y. Rekhter and T. Li. A border gateway protocol 4 (BGP-4). RFC 1771, Internet Engineering Task Force, Mar. 1995.

[85] E. Rescorla. *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley, 2000.

[86] J. F. Reynolds and J. B. Postel. Assigned numbers. RFC 990, Internet Engineering Task Force, Nov. 1986.

[87] M. T. Rose. Post office protocol: Version 3. RFC 1081, Internet Engineering Task Force, Nov. 1988.

[88] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. R. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: session initiation protocol. RFC 3261, Internet Engineering Task Force, June 2002.

[89] F. B. Schneider, editor. *Trust in Cyberspace*. National Academy Press, 1999.

[90] D. Seeley. A tour of the worm, 1988.

[91] D. Senie. Changing the default for directed broadcasts in routers. RFC 2644, Internet Engineering Task Force, Aug. 1999.

[92] U. S. P. Service and B. Wijnen. User-based security model (USM) for version 3 of the simple network management protocol (snmpv3). RFC 3414, Internet Engineering Task Force, Dec. 2002.

[93] T. Shimomura. *Takedown*. Hyperion, 1996.

[94] K. Sollins. TFTP protocol (revision 2). RFC 783, Internet Engineering Task Force, June 1981.

[95] E. H. Spafford. The Internet worm program: An analysis. *Computer Communication Review*, 19(1):17–57, January 1989.

[96] M. St. Johns. Authentication server. RFC 931, Internet Engineering Task Force, Jan. 1985.

[97] M. St. Johns. Draft revised IP security option. RFC 1038, Internet Engineering Task Force, Jan. 1988.

[98] M. St. Johns. Identification protocol. RFC 1413, Internet Engineering Task Force, Feb. 1993.

[99] J. Steiner, B. C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *Proc. Winter USENIX Conference*, pages 191–202, Dallas, TX, 1988.

[100] R. J. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, and M. Kalla. Stream control transmission protocol. RFC 2960, Internet Engineering Task Force, Oct. 2000.

[101] B. Taylor and D. Goldberg. Secure networking in the Sun environment. In *Procedings of the Summer Usenix Conference*, Atlanta, GA, 1986.

[102] P. Vixie. DNS and BIND security issues. In *Proceedings of the Fifth Usenix Unix Security Symposium*, pages 209–216, Salt Lake City, UT, 1995.

[103] V. L. Voydock and S. T. Kent. Security mechanisms in high-level network protocols. *ACM Computing Surveys*, 15(2):135–171, June 1983.

[104] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. F. Wu, and L. Zhang. Protecting bgp routes to top level DNS servers. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS)*, May 2003.

[105] B. Wellington and O. Gudmundsson. Redefinition of DNS authenticated data (AD) bit. RFC 3655, Internet Engineering Task Force, Nov. 2003.

[106] T. Ylonen. SSH – secure login connections over the internet. In *Proceedings of the Sixth Usenix Unix Security Symposium*, pages 37–42, July 1996.