CITI Technical Report 98–5

The Packet Vault: Secure Storage of Network Data

C.J. Antonelli
cja@citi.umich.edu

M. Undy
mundy@umich.edu

P. Honeyman
honey@citi.umich.edu

ABSTRACT

This paper describes the packet vault, a cryptographically secured archiver of network packet data. The vault captures network packets, encrypts them, and writes them to long-term CD-ROM storage for later analysis and evidentiary purposes. The crypto-graphic organization of the vault permits selected traffic to be made available without compromising the security of other traffic. Some weaknesses remain in the vault; we conclude with a description of present experiences and future plans.

June 25, 1998

The Packet Vault: Secure Storage of Network Data

C.J. Antonelli
`cja@citi.umich.edu`

M. Undy
`mundy@umich.edu`

P. Honeyman
`honey@citi.umich.edu`

INTRODUCTION

The goal of the packet vault project at the Center for Information Technology Integration is to provide cryptographically secured long-term storage of network packets, both as input data for intrusion detection algorithms and for evidentiary purposes.

In any computing environment, security threats are a major concern. Following Voydock and Kent [1], we classify threats into three categories:

- unauthorized release of information,
- unauthorized modification of information, and
- unauthorized denial of resource use.

Intrusions originating from the network, in which an intruder uses specially crafted packets to attack systems, constitute threats in all categories.

Creating a complete, permanent record of all activity on a subnet addresses these threats in several ways. First, intrusion detection algorithms can be executed using the record as the input packet source. Detectors can be run over the same record with different parameter settings, outputs of different detectors can be compared, and new detectors can be created and evaluated against the record. Each of these experiments requires a permanent record of packets.

Second, in response to an intrusion in progress, the packet vault can be attached to a subnet under attack; the packets it stores may be used to determine quickly the source and nature of the intrusion, and thus help shape the response. In addition, the vault can be permanently connected to a suspect subnet and the record examined periodically.

Finally, a properly constructed corpus of packet data may serve as evidence in a court of law.

In the remainder of the paper, we give an overview of the goals of the packet vault, followed by a discussion of the hardware, software, and cryptographic organization of the vault. Finally, we describe our experiences, and conclude with some observations and a discussion of future work.

GOALS

The architecture of the packet vault reflects the following goals:

- **Commodity.** Our fundamental orientation is to build a packet vault from commodity hardware and software, notwithstanding the attraction of expensive machines with fast buses and I/O devices. With a vault built from cheap parts in hand, it is clear that we can trade money for speed by buying faster parts.
- **Completeness.** To create a complete record it is vital to capture and store every packet. We believe that an adversary can exploit any attempt at packet triage; the only way to defend against such attacks is to build a vault that can store packets at the maximum rate the network can deliver them.
- **Permanency.** We decided from the outset that our storage medium would be CD-ROM, because of consistently bad long-term experiences with data storage on magnetic tapes, and because we wanted to learn a bit about CD-ROM writers. We are not concerned with the relatively low data rates of the writers, as

we can depend on them to improve, and in any case we can use multiple writers.

- **Openness.** Finally, we assume that the CD-ROMs containing network traffic will be available for unsupervised inspection, either intentionally or by larceny. It is critical that the data stored on them be protected with strong cryptography and organized in such a way that some subsets of the traffic can be revealed without exposing others. Ideally, we would like to publish keys that unlock certain data on a given CD-ROM, without the possession of those keys exposing other data on it.

We observe that our goals of commodity and completeness are in tension, particularly at network speeds above 10 Mbps. Our goal is to construct a vault that can store all packets on a typically loaded 10 Mbps Ethernet network, and to depend on faster hardware to improve the rate at which packets can be acquired.

ARCHITECTURE

An early design consideration was whether a single commodity machine could accept packets from the network, encrypt them, and write them to CD-ROM without becoming overloaded. Early experiences with the bursty nature of Ethernet networks, coupled with the real-time requirements of CD-ROM recorders† convinced us that two machines would be necessary.

The packet vault hardware is composed of two 133 MHz PCI-bus Pentium machines interconnected via a private 100 Mbps Ethernet. One machine (the "listener") is connected to the network being monitored and is used to capture and encrypt the data, which are then sent over the private Ethernet. The listener never stores packets on magnetic disk.

The other machine (the "writer") receives the encrypted packets and assembles them on magnetic disk for subsequent writing to CD-ROM. The two magnetic disks on the writer are attached to a common SCSI bus. A second SCSI bus dedicated to the CD-ROM recorder (CD-R) avoids bus contention. Figure 1 shows the hardware architecture of the packet vault.

† Our CD-ROM recorder, like all early commodity recorders, possesses a small (512 KB) data buffer and thus requires the attached host to maintain a constant data rate. Failure to maintain the required rate results in a ruined CD-ROM session or disk. In addition, data cannot be added incrementally to a CD-ROM; they must first be formatted into an ISO-9660-compliant image on magnetic disk and then written to the CD-ROM *in toto.*

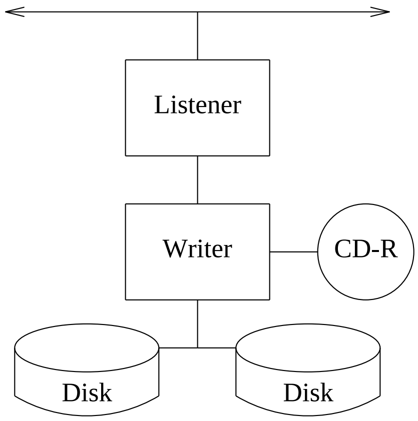

Figure 1

We chose UNIX for both listener and writer because of its familiarity and flexibility. OpenBSD was chosen for the listener for its kernel packet filtering support; early availability of CD-R drivers dictated the choice of Linux for the writer.

We use BPF [2] on the listener to capture all packets seen on the 10 Mbps network being monitored and write them to an *accumulator* file in a memory file system (MFS [3]). We modified the BPF code to pass packets directly from the kernel network buffers to MFS, obviating two copies between user and kernel space. A listener process monitors the size of the accumulator file and renames it when it reaches 4 MB in size or after 1 minute has elapsed, which keeps the sizes of the MFS packet files manageable. The names of the packet files reflect the time of day they were created. Another process on the listener polls the MFS for new packet files, encrypts the contents, and uses `rcp` to copy the files over the private 100 Mbps link to the writer. Unencrypted data are stored only in the MFS, so in the event of a system failure no unencrypted data remain.†

When enough packet files have accumulated on the writer to fill a CD-ROM, a background process is spawned on the writer. The writer process generates an ISO-9660-compliant image on magnetic disk containing the packet files and the cryptographic material necessary to permit

† We do not run the listener with swapping enabled, and ignore potential attacks on RAM hardware [4].

recovery of the packet data. The image is written and later purged from magnetic disk. A double-buffering scheme is used so that image generation writes and subsequent packet file writes do not contend for the same physical disk. The packet data path is shown in Figure 2.

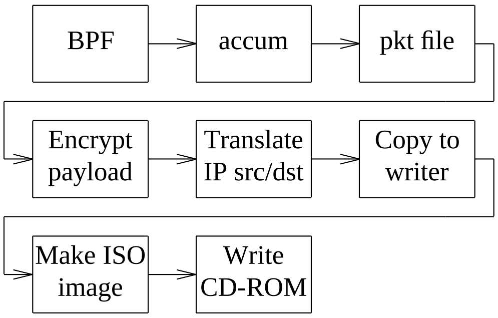

Figure 2

CRYPTOGRAPHIC ORGANIZATION

The cryptographic organization of the vault follows from our requirement that vault data be publishable. Ideally, we would like to provide free access to a mass storage device filled with vault data while providing fine-grained access to individual packet contents. Our basic strategy is to encrypt all packet payloads; the challenge is then to devise a means of associating different keys with different packets, to some level of granularity. Clearly, both ends of the spectrum are unacceptable: one key per CD-ROM risks a serious breach if lost; managing a different key for each packet quickly becomes unmanageable.

Our unit of granularity is the *conversation,* defined as a set of packets with the same source and destination IP addresses. We considered including port numbers for finer control, but this would require special treatment for non-TCP streams, and created problems with port-agile applications.

Each CD-ROM *volume* holds sufficient information to reconstruct the packet traffic it stores, thus no ancillary information need be managed. We use a multi-level encryption scheme. Symmetric key encryption is used to seal packet payloads and any additional information necessary to reconstruct the packets (explained below). Asymmetric key encryption is used to encrypt the symmetric keys. A trusted third party such as the Regents of the University of Michigan holds the private key. Figure 4 shows the cryptographic organization on CD-ROM.

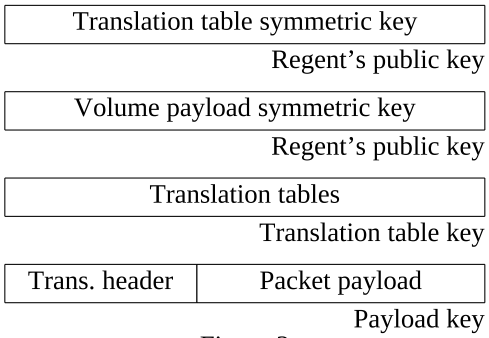

Figure 3

Our implementation uses 1024 bit PGP [5] for asymmetric key and DESX [6] for symmetric key encryption. Starting with Karn's DES implementation [7] we added both pre- and post-whitening steps for each block:

$$DESX_{k.k1.k2}(x) = k2 \oplus DES_k(k1 \oplus x)$$

This requires 184 bits of key material, and conservatively extends the effective key length of DES in our environment to at least 95 bits, with respect to key search in the sense of Kilian and Rogaway [6], while adding a trivial amount of computation to each block encryption.†

To hinder traffic analysis, we obscure source and destination addresses by substitution. A translation table mapping real to substituted addresses is encrypted with DESX using a *translation table key* K_T unique to each volume. A second table mapping pairs of real to substituted source and destination addresses, also encrypted with the translation table key, allows reconstruction of the conversations contained on a disk without requiring a search of the entire disk to establish conversation pairs. Both translation tables are written to CD-ROM.

A key is constructed for a given conversation by combining the concatenated, untranslated source and destination IP addresses with a 192-bit *volume master key* K_V using exclusive-or, and then using DESX in CBC mode to encrypt a 192-bit constant with the combined value:

$$K_{C_i} = DESX_{K_V \oplus (SA_i \,||\, DA_i)}(CONST)$$

The resulting 192-bit *conversation key* K_{c_i} is used to encrypt packet payloads of the conversation:

† We assume an attacker could obtain all the plaintexts for all encrypted packets on a volume, and that the average packet length is 100 bytes, yielding 6 million plaintext/cyphertext pairs. Rogaway's effective key length expression [8] then becomes $55+64-1-lg\,(6\times10^6)$

$$C_i = DESX_{K_{C_i}}(P_i)$$

A new volume master key and translation table key are generated for each volume. Currently, they are computed from previous keys:

$$K_{V_{i+1}} = DESX_{K_{V_i}}(K_{V_i})$$

$$K_{T_{i+1}} = DESX_{K_{T_i}}(K_{T_i})$$

where K_{V_0} and K_{T_0} were randomly generated. We plan to replace this with a practically strong random data accumulator and generator, implemented according to Gutmann [9].

Finally, a new pair of PGP keys are generated per vault instance. The public key is used to seal the volume master and translation table keys before they are written to CD-ROM.

EXPERIENCES

The packet vault has been up and running sporadically for the last year, collecting packets from a 10 Mbps Ethernet that is usually lightly loaded but exhibits periods when traffic exceeds 70% due to experimental video work.

The major challenges in the construction and operation of the vault have been systems engineering and integration. Bottlenecks discovered along the way were removed until the vault could handle the incoming network traffic. For example, it was discovered that passing packets in and out of the kernel from BPF to MFS was too slow, so we modified the listener's kernel to skip the kernel/user space copies.

Disk usage on the writer must be monitored closely because of the large volumes of data involved. Currently, the vault does not clean up when interrupted; to achieve reliable operation upon restarting, six locations spread across both machines must first be checked for abandoned temporary files.

The data path consists of several stages, some of which process data in parallel, and some sequentially. Payload encryption and network copying are the most costly operations in this pipeline, yet both of these operations occur sequentially. Generating the image and writing the image to a CD are also costly, but as larger buffers are available for these steps only the average throughput is of importance.

If the sustained input rate exceeds the throughput of any stage in the data path, eventually some buffer becomes exhausted and the vault fails. The first failure is almost always caused by the MFS filling up, which crashes the listening process. Experimentally, with a 70% utilization of the source Ethernet, the vault crashes after about two minutes. Increasing buffer sizes is of limited practical value; doubling the memory allocated to MFS extends this time to four minutes.

At 70% network utilization, while the writer is busy generating a CD image, its disk and processor utilizations increases dramatically, and the `rcp` time increases by a factor of two to three. It takes about 7 minutes to generate an image under these conditions. A bug validated our assumption that double-buffering was needed: a failure to toggle the drive on which the image was being created resulted in packet files for every other volume being written to the same disk on which an image was being built; the resulting overload backed up the data path and crashed the vault.

The other obvious target for performance optimization is the encryption code. We use a machine-specific implementation of the DES code compiled with full optimization and aggressively cache the DES key schedules. These changes speed up the encryption task by over 80%, but opens the door to a denial of service attack by an adversary who manufactures packets that defeat the caching.

FUTURE WORK

The focus of this work is the creation of a cryptographically secured record of packet activity on a given subnet. The next major step involves focusing on intrusion detection methods, using this corpus as a virtual network testbed. We have plans for more specific improvements to the packet vault.

Hardware

Our vault could clearly benefit from more capable hardware.

Software

Better administrative and fault-handling scripts are needed for graceful shutdown and restart of the vault. An occasional inability of the writer to allocate buffer space for the private Ethernet link remains to be resolved.

The high disk loads caused by creating an ISO-9660 image *en masse* could be ameliorated by constructing the image incrementally.

Limits of Passive Protocol Analysis

Ptacek and Newsham [10], point out a shortcoming in passive protocol analysis due to the inability of an intrusion detection system to determine accurately what is happening on networked machines. They identify three classes of attacks: insertion, in which the detector is made to see traffic that the victim does not; evasion, in which the victim sees traffic the detector does not; and denial of service, in which the detector is fed traffic designed to cause it to fail.

The packet vault is subject to these attacks. However, since the vault obtains packets directly from the link level device driver, it does nothing beyond reading and storing each packet as it arrives on the interface. Fragment reassembly, management of TCP connection state, and so forth are left to the analysis phase after the CD-ROMs are written. This causes attacks on the vault by, say, deliberately overlapping fragments to fail, as the vault does not reassemble them; further, the complete evidence is stored for later analysis.

As long as the recording rate exceeds the arrival rate, then the packet vault defeats evasion and denial of service attacks. Insertion attacks are still problematic; however, permanently storing all packets does permit them to be replayed against an instrumented networked machine.

ACKNOWLEDGEMENTS

We thank Mike Stolarchuk for his contributions to the architecture of the packet vault. He also wrote the BPF layer modifications, and provided invaluable systems engineering assistance. Dan Boneh suggested the conversation key mechanism. This work was partially supported by Bellcore.

1. V.L. Voydock and S.T. Kent, ‘‘Security Mechanisms on High-Level Network Protocols,’’ *Computing Surveys* **15**(2), pp. 135–171 (June, 1983).
2. Steven McCanne and Van Jacobson, ‘‘The BSD Packet Filter: A New Architecture for User-level Packet Capture,’’ pp. 259–269 in *Winter 1993 USENIX Conference Proceedings*, San Diego (January, 1993).
3. Marshall Kirk McKusick, Michael J. Karels, and Keith Bostic, ‘‘A Pageable Memory Based Filesystem,’’ pp. 137–143 in *Summer 1990 USENIX Conference Proceedings*, Anaheim (June, 1990).
4. Peter Gutmann, ‘‘Secure Deletion of Data from Magnetic and Solid-State Memory,’’ pp. 77–89 in *Sixth USENIX Security Symposium*, San Jose (July, 1996).
5. William Stallings, ‘‘Protect Your Privacy: The PGP User’s Guide,’’ *Prentice-Hall*, New Jersey (1995).
6. Joe Kilian and Phillip Rogaway, ‘‘How to Protect DES Against Exhaustive Key Search,’’ pp. 252–267 in *Advances in Cryptology - Crypto ’96, Lecture Notes in Computer Science*, ed. N. Koblitz, Springer-Verlag (1996).
7. Phil Karn, karn@unix.ka9q.ampr.org (December, 1995).
8. Phillip Rogaway, *RSA Laboratories’ CryptoBytes* **2**(2) (Summer, 1996).
9. Peter Gutmann, ‘‘Software Generation of Cryptographically Strong Random Numbers,’’ pp. 243–257 in *Winter 1998 USENIX Security Symposium Proceedings*, San Antonio (January, 1998).
10. Thomas H. Ptacek and Timothy N. Newsham, *Insertion, Deletion, and Denial of Service: Eluding Network Intrusion Detection*, Secure Networks, Inc. (January, 1998).