2013 IEEE Security and Privacy Workshops

Reporting Insider Threats via Covert Channels

David N. Muchene, Klevis Luli, and Craig A. Shue
Worcester Polytechnic Institute
{dmuchene.92, klevis}@wpi.edu, cshue@cs.wpi.edu

***Abstract*—Trusted insiders that betray an organization can inflict substantial harm. In addition to having privileged access to organization resources and information, these users may be familiar with the defenses surrounding valuable assets. Computers systems at the organization need a mechanism for communicating suspicious activity that is difficult for a malicious insider (or even an outsider) to detect or block.**

In this work, we propose a covert channel in the Ethernet frame that allows a computer system to report activity inside other, unrelated network communication. The covert channel leverages the differences in the framing approaches used by Ethernet and IP packets to append hidden information to IP packet and transmit it to an organization's administrator. This stealthy communication is difficult for even advanced attackers and is challenging to block since it opportunistically uses unrelated communication. Further, since the transmission is tied to the Ethernet frame, the communication cannot traverse network routers, preventing security information from leaving the organization.

We introduce the covert channel, incorporate it into a working prototype, and combine it with an intrusion detection system to show its promise for security event reporting.

I. Introduction

With detailed insight into an organization, trusted insiders can inflict more harm than even outsiders. In 2009, the United States Department of Homeland Security indicated that the insider threat was one of eleven hard computer security problems [1]. The 2010 exposure of US diplomatic information via WikiLeaks [2] underscored this risk, creating international diplomatic tension and showing how a single trusted insider can undermine an organization's mission objective.

The insider threat problem has received significant attention, with work focusing on individual sectors [3], [4] and other outlining mitigation strategies for all types of organizations [5]. Insider threat research in the computer security community has used diverse approaches, ranging from system log and system call analysis to honey pots [6]. While these mechanisms may be effective at detecting insider threats, they general ignore how these threats are reported to security officials.

The most obvious reporting mechanisms are also the easiest for an attacker to foil. An organization may run a process on each of their systems that sends an alert to a system administrator's computer whenever a threat is detected. This process may communicate with the administrator's machine using a network socket. However, such communication is easily detected. Network reporting utilities, such as `netstat`, are installed by default in most commonly used operating systems and can be invoked by non-administrator users. Simply by monitoring the network connections and the number of bytes transmitted, insiders can detect when security events are being reported. Further, with firewall tools, a user can simply block the security reporting channel, preventing alerts from being transmitted while exfiltrating information.

Organizations need a more robust reporting mechanism that can covertly transmit information to a security official. Such reporting must be invisible to non-administrator users and must be challenging even for administrators to detect. At the same time, this reporting mechanism must be reasonably high bandwidth: not only should administrators know if an attack is occurring, they would ideally be able to obtain a copy of all the attacker's network traffic, for example. Finally, the reporting mechanism must ensure security information does not leave the organization's periphery, without requiring any special configuration on the computer hosts.

In this work, we present a covert channel for communicating security alerts to organization administrators. Covert channels allow the stealthy communication of information in computer networks. Rather than hide the contents of a message, via mechanisms such as encryption, covert channels attempt to hide the very existence of the communication. Such covert channels were first introduced by Lampson [7] in the context of exfiltration of high security information to a low security process in a mainframe. Later work expanded covert channels into computer networking contexts [8]. Importantly, with each of these covert channels, disclosure of the channel mechanism renders the approach useless since monitoring can easily expose usage of the channels once the examiner knows where to look.

Covert channels have traditionally been used by attackers to subvert organizational policy. Instead, we propose using covert channels to enforce security policies. While previous covert channels sacrificed communication bandwidth for stealthiness, we aim for a mechanism that is practical for security reporting while making subversion challenging. In this work, we make the following three contributions:

- **Covert Channel Proposal:** We introduce a covert channel between the end of an IP packet and the closure of an Ethernet frame that allows arbitrary messages.
- **Channel Implementation:** Using existing Linux kernel libraries, we implement senders and receivers of the covert channels.
- **Channel Evaluation:** We evaluate the bandwidth and stealthiness of the channel and use it in reporting events from an intrusion detection system (IDS).

DOI 10.1109/SPW.2013.30

II. Background and Related Work

Covert channels have traditionally focused on allowing an attacker to communicate without detection. These channels can use the actual data, such as the network packet payload or file contents, or metadata, such as network packet headers or file attributes, to communicate information. The former technique falls largely into research on *steganography* and is more challenging to generalize into arbitrary network traffic. However, the latter approach has been explored extensively by the network security research community.

Network protocols have a large number of fields and careful manipulation of these fields can allow the communication of covert information. The most closely related work to ours, by Jankowski *et al.* [9], uses Ethernet, ARP, and TCP to create a channel. Previous work by Arkin and Anderson [10] found that some systems leaked information in the padding bytes added to packets under the 64 byte Ethernet minimum. While the padding bytes should all be zero, some operating systems did this incorrectly. In creating PadSteg, Jankowski *et al.* leveraged these padding bytes to encode secret information, much like our proposed approach, but PadSteg also required modification to upper layer headers, such as ARP and TCP, to indicate the presence of the covert message. A more significant limitation of the approach is the limited bandwidth for covert messages. Many packets exceed the 64 byte Ethernet minimum (a typical TCP packet uses at least 52 bytes in header alone), and for the fewer remaining packets, the padding is typically only 10-15 bytes, limiting the size of covert messages. Other approaches propose using padding for covert channels [11], [12], but each has similar bandwidth limitations that would hinder practical security event reporting.

The IP, TCP, UDP, ICMP, and DNS headers provide fodder for covert channels as well, given the availability of reserved fields and fields that can be slightly altered without undermining communication [11], [13]–[17]. Even IPv6, which still has low levels of adoption, has been explored for covert channels [18], [19]. Many of the covert channels are implemented by inserting data in unused header bits of these protocols [20]. As an example of this type of covert channel, Rowland [13] created a channel by manipulating the IP identification field (ID), the TCP Initial Sequence Number (ISN) field, and the TCP ACK sequence number field. Another approach, by Jones *et al.* [17], used the time-to-live (TTL) field in the IP header to encode covert information. Since the TTL field is modified by routers during packet transit, the capacity of the channel will be related to the natural TTL variation on a path, which can be analyzed to encode a signal while still looking natural. While clever, each of these channels are limited in bandwidth.

Covert channels are fundamentally tied to a notion of "security through obscurity." If the channel mechanism is discovered, the usage of such as channel can be easily detected by a third-party observer. Murdoch and Lewis [21] showed that modifications to header fields, including TCP timestamps, could be efficiently identified because benign traffic exhibits a uniform structure, allowing easy detection of deviations. In other cases, these channels can be closed through preventative measures, such as protocol normalizers, without the third-party even knowing about the details of the channel's operation or existence [12], [20].

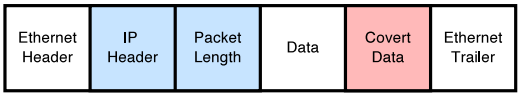

Fig. 1. The proposed approach creates a covert payload after the end of the IP packet payload, but before the end of the Ethernet frame.

III. Our Approach

The key idea in our approach is to leverage the differences in framing techniques used by the Ethernet and IP protocols. The Ethernet frame uses a header and trailer approach to begin and end an Ethernet frame while the IP protocol specifies the packet length as part of its header. Accordingly, we can simply insert covert payload in bytes that are after the length indicated in the IP header, causing them to not be part of the IP payload, but before the Ethernet tailer. In Figure 1, we show where the covert data would be encoded relative to the rest of the packet.

In our approach, we are not tied to fixed bits in the packet header or limited bits in padding packets up to the required length. Instead, we can add covert information to any packet that is less than the local segments maximum transmission unit (MTU). The size of the covert information that can be stored in each packet is the difference of the MTU and the packet's size. In some packets, this space may still be relatively small. In others, such as acknowledgements in the TCP protocol, there may be a large amount of available capacity in the channel. The average IP packet size measured at border routers is about 300 bytes in size [22], leaving an average of around 1200 bytes of covert payload for each packet.

To determine the feasibility of this covert channel for security threat reporting, we first implement the approach and describe its feasibility for broader adoption. We then evaluate the channel's compatibility with security reporting software.

A. Prototype Implementation

To allow us to determine whether local networks would properly deliver packets using our covert channel, we implemented a simple proof-of-concept test. We created a client and server that would communicate with each other using UDP. Whenever the server would transmit a message, it would do so using raw sockets. This allowed us to provide the entire IP portion of the packet, which would be encapsulated in an Ethernet frame by the underlying network hardware. In creating these packets, we intentionally set the packet length field in the IP header to be shorter than the buffer we supplied to the underlying network call. By monitoring the client end of the communication with a packet capture tool, such as Wireshark [23], we could confirm whether the intervening network would support the covert channel.

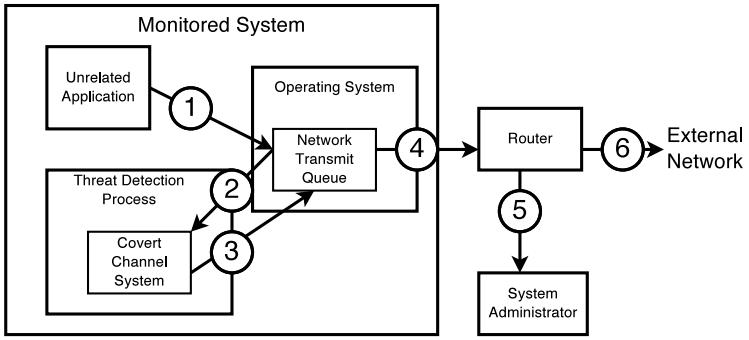

Fig. 2. An example network using covert channels. Each arrow represents network communication in one of six steps in the process. Message 1 is a packet from an unrelated message that is transmitted to the kernel. Message 2 is an interception of Message 1 using divert sockets. Message 3 is an altered version of Message 1 that includes a covert payload after the IP packet data. Message 4 is the transmission of the prior message from the kernel through the network interface. If the packet arriving at the router contains a covert payload, the router creates a packet (Message 5) that includes the entire incoming packet and covert payload, sending it to the organization's system administrator. Finally, in Message 6, the router strips the Ethernet frame (including the covert message) and repackages the IP packet and payload for transmission to the destination.

In our first test, we implemented the approach using two computers running Ubuntu Linux 11.10 with a direct network link. Monitoring at the client showed that the packet correctly arrived with its covert payload, allowing the client to harvest this information. We then repeated the experiment using a Netgear Ethernet switch between the two hosts. The switch did not affect the transmission, showing that the approach could be used in a local switched network. However, as expected, the covert channel payload was discarded when a device using network address translation (NAT) was placed between the hosts. In this case, the destination host could still receive the original packet, but without the associated covert data.

This implementation allowed us to create a covert channel simply because the host operating system did not check the IP packet header's length before encapsulating the packet. However, some operating systems do employ such checks. In particular, the same raw socket server program executing on a FreeBSD 9.0 machine produced the packet, but without the covert payload. This indicates that the covert channel may be natively supported on some operating systems, but others may require kernel-level modifications to support the covert channel.

B. Encoding Covert Channels in Unrelated Network Traffic

While our initial prototype demonstrated that covert channels are feasible in modern network hardware, it uses raw sockets to create a separate communication flow. To be more robust, we need an approach that allows the channel to blend in with normal network traffic from the system. In particular, we would like to intercept out-bound packets from the machine and, if the packet is small enough, add the covert data at the end of the packet.

Fortunately, such mechanisms are readily available in some operating systems. FreeBSD 9.0 provides kernel support for a divert socket module that allows an intercepted packet to be modified and requeued for transmission by the operating system. Linux also appears to offer support for packet diversion through the Netlink Protocol Library [24]. For our experiments, we chose to use the FreeBSD divert sockets due to their longer support history.

Our packet modification approach involves two stages. In the first, we redirected network communication using FreeBSD's packet filter, `ipfw`, to our application. We then checked the packet's size and appended our message to be transmitted and returned the packet to the operating system's queue. We could see these modification, showing that packet diversion works correctly, but only after we increased the length in the IP packet header to bypass FreeBSD's packet size validation. Accordingly, we could demonstrate the feasibility of diversion for the covert channel, but we were unable to do such out-of-packet covert data without kernel modifications.

C. Security Alert Reporting System

With evidence that we can create the covert channel, along with the ability to modify unrelated traffic from other applications, we now consider the broader security implications. In Figure 2, we provide an overview diagram of the approach. We depict a threat detection process that uses a kernel API to obtain queued network traffic from another application, modifies it to include security alerts as a covert channel, and returns it to the operating system's queue. The operating system then transmits the packet from the queue through the system's network card. The packet then traverses the local network until it reaches a routing device. The router, aware of the covert channel, can send a copy of the packet to a system administrator for analysis. The router then proceeds with normal Internet operation, in which the IP packet is extracted and reencapsulated in a new datalink layer frame. This has the effect of discarding the covert payload for packets leaving the network.

We implemented this system using the Snort IDS [25], appending messages as covert payload in our prototype application. The Snort alerts were relatively small, easily fitting within the confines of the 1200 byte available in the average packet. When we induced an ICMP ping flood attack, we were able to encode the message and extract it at a router process

in the network, which could redirect it to an administrator system.

One of the key advantages of this approach is that it allows reporting of events without a separate, dedicated network connection. A monitoring application running with administrator privileges can communicate without a normal user being able to detect or prevent the communication. Even users with administrator privileges would be unlikely to detect the covert communication without a packet analyzer and insight into the channel's operation. Encryption could additionally be used to decrease the chance of discovering the covert payload. When combining our approach with covert monitoring process, such as those enabled through DLL injections and API hooking [26], organizations can enable insider monitoring without an identifiable monitoring process or communication associated with it.

Finally, organizations may be concerned about leaking security information to off-site systems, including to individuals colluding with an attacker. Because we encode the covert channel inside an Ethernet frame, this communication cannot traverse network gateways. The approach does not require any custom network configuration on the host to control where the alerts are sent; only the organization routers need to be aware of the protocol to redirect traffic. If the host leaves the network, such as a laptop, the security information will be dropped by legacy routers without hindering network communication. These features are intrinsic to the covert channel and lessen the risk of deploying the approach.

IV. Conclusion

In this work, we propose using covert channels for reporting security alerts to organization administrators. We further introduce a covert channel that provides sufficient bandwidth for the timely transmission of these security alerts, which was not possible in previous covert channel approaches. We evaluated the approach using two operating systems, Ubuntu Linux and FreeBSD, and showed that the channel works in modern network hardware and operating systems. We further found that we can modify unrelated network packets to include security information, complicating detection and prevention of security alert transmission. Our testing with a popular intrusion detection system confirmed that the approach would be compatible with security reporting.

While malicious insiders can be damaging to an organization, such attackers may be discouraged when they know the risk of detection and apprehension is higher. By combining sophisticated insider threat monitoring tools and covert network communication, an organization can raise the costs and risks for attackers. Our approach allows organizations to inform their employees about monitoring and alerting, without exposing the implementation details that may aid malicious insiders in their circumvention efforts.

In future work, we plan to integrate the covert channel creation and packet diversion approaches to create a unified reporting mechanism. We further will explore additional operating systems, including Microsoft Windows and Mac OS X, which have a higher user base than the Linux and Unix systems we investigated.

References

[1] US Department of Homeland Security, "A roadmap for cybersecurity research," US Department of Homeland Security Whitepaper, November 2009.

[2] Sunshine Press, "Wikileaks," 2012. [Online]. Available: http://wikileaks.org/

[3] M. Keeney and the United States Secret Service, *Insider threat study: Computer system sabotage in critical infrastructure sectors*. US Secret Service and CERT Coordination Center, 2005.

[4] A. Cummings, T. Lewellen, D. McIntire, A. Moore, and R. Trzeciak, "Insider threat study: Illicit cyber activity involving fraud in the US financial services sector," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Special Report CMU/SEI-2012-SR-004, 2012.

[5] G. Silowash, D. Cappelli, A. Moore, R. Trzeciak, T. Shimeall, and L. Flynn, "Common sense guide to mitigating insider threats, 4th edition," Software Engineering Institute, Carnegie Mellon University, Tech. Rep. CMU/SEI-2012-TR-012, 2012.

[6] M. Salem, S. Hershkop, and S. Stolfo, "A survey of insider attack detection research," *Insider Attack and Cyber Security*, pp. 69–90, 2008.

[7] B. Lampson, "A note on the confinement problem," *Communications of the ACM*, vol. 16, no. 10, pp. 613–615, 1973.

[8] G. Simmons, "The prisoner's problem and the subliminal channel," in *Advances in Cryptology: Proceedings of Crypto*, vol. 83, 1984, pp. 51–67.

[9] B. Jankowski, W. Mazurczyk, and K. Szczypiorski, "Information hiding using improper frame padding," in *Telecommunications Network Strategy and Planning Symposium (NETWORKS), 2010 14th International*. IEEE, 2010, pp. 1–6.

[10] O. Arkin and J. Anderson, "Etherleak: Ethernet frame padding information leakage," http://leetupload.com/database/Misc/Papers/atstake_etherleak_report.pdf, 2003.

[11] T. Handel and M. Sandford, "Hiding data in the OSI network model," in *Information Hiding*. Springer, 1996, pp. 23–38.

[12] G. Fisk, M. Fisk, C. Papadopoulos, and J. Neil, "Eliminating steganography in internet traffic with active wardens," in *Information Hiding*. Springer, 2003, pp. 18–35.

[13] C. Rowland, "Covert channels in the TCP/IP protocol suite," *First Monday*, vol. 2, no. 5, 1997.

[14] A. Hintz, "Covert channels in TCP and IP headers," *Presentation at DEFCON*, vol. 10, 2002.

[15] K. Ahsan and D. Kundur, "Practical data hiding in TCP/IP," in *Proc. Workshop on Multimedia Security at ACM Multimedia*, vol. 2, no. 7, 2002.

[16] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibbetts, "Covert messaging through TCP timestamps," in *Privacy Enhancing Technologies*. Springer, 2003, pp. 189–193.

[17] E. Jones, O. Le Moigne, and J. Robert, "IP traceback solutions based on time to live covert channel," in *IEEE International Conference on Networks*, vol. 2. IEEE, 2004, pp. 451–457.

[18] N. Lucena, G. Lewandowski, and S. Chapin, "Covert channels in IPv6," in *Privacy Enhancing Technologies*. Springer, 2006, pp. 147–166.

[19] T. Graf, "Messaging over IPv6 destination options," *The Swiss Unix User Group, Switzerland, http://gray-world.net/papers/messip6.txt*, 2003.

[20] S. Zander, G. Armitage, and P. Branch, "A survey of covert channels and countermeasures in computer network protocols," *IEEE Communications Surveys and Tutorials*, vol. 9, no. 3, pp. 44–57, 2007.

[21] S. Murdoch and S. Lewis, "Embedding covert channels into TCP/IP," in *Information Hiding*. Springer, 2005, pp. 247–261.

[22] L. Peterson and B. Davie, *Computer Networks: A Systems Approach*, 5th ed. Elsevier, 2011, p. 278.

[23] The Wireshark Foundation, "Wireshark," http://www.wireshark.org, 2013.

[24] "Netlink protocol library suite," http://www.infradead.org/~tgr/libnl/, 2013.

[25] Sourcefire, Inc., "Snort," http://www.snort.org/, 2013.

[26] J. Berdajs and Z. Bosnić, "Extending applications using an advanced approach to dll injection and api hooking," *Software: Practice and Experience*, vol. 40, no. 7, pp. 567–584, 2010.