The Case for Network Witnesses

Wu-chang Feng
Portland State University

Travis Schluessler
Intel Corporation

***Abstract*—Network abuse is prevalent in today's Internet. To combat abuse, this paper examines a general approach for constructing network protocols based on the use of "network witnesses": tamper-resistant, trusted third parties that reside at network protocol end-points. By providing authentic measurements of network use and by ensuring the correct usage of network protocols, we show how network witnesses can enable fundamentally new protocol designs that can protect networks against malicious use.**

I. Introduction

The original Internet protocols assumed cooperative entities were running them. As a result, many protocols did not adequately consider malicious users which has lead to widespread abuse. Although a number of mechanisms have been deployed to address individual problems, network abuse ranging from botnets sending spam to hackers performing scans is still prevalent. In this paper, we explore a general approach for combating the problem of network abuse by constructing network protocols around "network witnesses". Network witnesses are tamper-resistant, trusted third parties that reside at network protocol end-points and can be used to help ensure the correct usage of network protocols by those endpoints. Network witnesses enable fundamentally new protocol designs that can protect networks against malicious use. Towards this end, we examine how future networks might change if a network witness were made mandatory for participants.

II. System Model

The crux of the approach is the network witness: a trusted third party entity comprised of tamper-resistant hardware and software that resides on the platform at network end-points. In order for a platform component to fulfill the role of a network witness, certain criteria must be met. We assume network witnesses provide:

- *Reliable introspection*: The network witness must be capable of reliably measuring the state of the host system including its network usage.
- *Attestation*: The network witness must be able to report its measurements in an authenticated manner to other witnesses in the network.
- *Isolation*: The network witness must be isolated from the host processor to ensure that actions taken by the host do not influence the witness. This is required to prevent an attacker in possession of the system from circumventing or disabling the network witness through software.
- *Trusted execution*: The network witness must only execute code that is cryptographically signed by a trusted third party such as the IETF or the manufacturer of the hardware.
- *Tamper-resistance*: The network witness cannot be tampered with by a network-based adversary or the platform owner. In reality, a tamper-proof platform is unattainable, hence, this requirement translates into a cost of circumvention requirement, that is, the cost of tampering with the network witness must exceed the value of the services it protects.

Network witnesses represent an extension of what has been deployed by industry. Specifically, one candidate witness is the Manageability Engine (ME) component of Intel's Active Management Technology [1]. The ME is a hardware component that currently resides in the chipset on many Intel motherboards providing manageability services to IT administrators. Its capabilities go further than simple manageability though. The ME can be used to report authentic integrity checks on the running state of the operating system and applications in a tamper-resistant manner [2]. The ME also has the capability to monitor network traffic going to and from the host and can filter traffic known to be malicious [3]. Figure 1 shows a conceptual picture of AMT in its current form. As the figure shows, the ME sits along the path between the host and the network interface and has direct access to the memory image of the target host. As a result of this placement, it can perform automatic worm containment as well as network access control based on the software state of the host. Specifically, AMT can use heuristics to detect the scanning behavior of worms that are attempting to propagate from the host. To implement network access control based on the host's security state, AMT can be used in conjunction with Cisco's Network Admission Control (NAC) to create "virtual enterprise

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0627752 and by the generous donations of Intel Corporation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or Intel.

978-1-4244-2652-2/08/$25.00 ©2008 IEEE

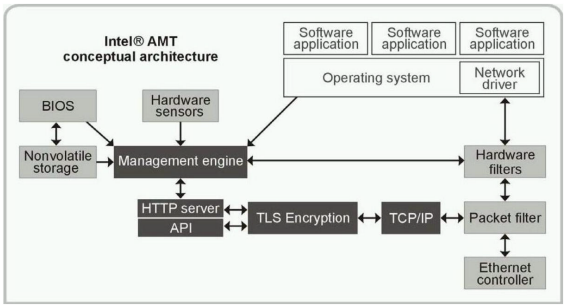

Fig. 1. Architecture of Intel's Active Management Technology (AMT)

networks" [4]. Based on AMT measurements of host integrity, network traffic from compromised hosts can be isolated and quarantined automatically.

III. Protocol Attestation and Validation

Network witnesses can simplify the design of network protocols and algorithms much like trusted third parties can greatly simplify complex security protocols. One straight-forward use for network witnesses is to have them authentically attest to "ground truth" measurements on the host in order to provide situational awareness in the network. In the solution described previously using Intel's AMT and Cisco's NAC [4], the Manageability Engine measures the "security posture" of the untrusted software running on the host it resides on. This security posture includes the patch-level of the applications and operating system and the presence of anti-virus agents. This attested posture is then sent into the network where network routers implementing Cisco's NAC decide what network resources to allow the host to access. The attestation can be done using methods such as TCG's TNC [5]. In that model the network witness would be an Integrity Measurement Collector, and the decision point in the network would be an Integrity Measurement Verifier. If the posture reported by the witness is poor (e.g., the host is not running anti-virus programs), then its traffic is automatically isolated onto a virtual network that keeps it from communicating with the rest of the hosts on the enterprise network. In order to ensure proper operation, the network mutually authenticates itself with the AMTs on all of the attached hosts. While the market success of this product demonstrates the strength of the idea, the notion of using attested host measurements in designing robust networks of the future represents a significant and fundamental advance that requires a more extensive look. In the future, actions taken by network witness enabled systems could be verified by network infrastructure, such as in the NAC scenario, or by other network endpoints participating in a joint networked application.

A. Attesting human input

Perhaps the most important problem in networks today is automated attack. Such attacks include automated web account signup, comment and e-mail spam, denial of service attacks, ticket purchasing robots, click fraud, and bots in on-line games [6], [7], [8]. Several kludges exist to tackle automation. One is to just tell the bot to go away and trust that it will (i.e., the use of robots.txt to keep search engines from indexing certain web sites). This is easily circumvented. Another is to use CAPTCHAs to try and prove the presence of human on the other end. This makes networked applications much harder to use and is becoming less tenable as improved decoding algorithms have been developed [9].

A more elegant approach for doing so is to have the network witness attest to the fact that a human is present by measuring pieces of the system that are the result of physical events (i.e., keyboard and mouse events). Such a facility is the basis of an extension to Intel's AMT [10] where the Manageability Engine attests that the events coming from keyboard and mice on the USB bus matches those the software sees. While the system focuses on detecting automation in on-line games, it clearly has relevance to a range of other networked applications that could benefit from knowledge of human use.

For example, attested measurements of keyboard and mouse events could clearly aid the e-mail spam problem. A client SMTP server could have the sending host attest to whether the keyboard and mouse was used in the last minute upon receiving an e-mail from it. Combined with content-based analysis, if the e-mail looks like spam and the sender's keyboard and mouse has been idle, the SMTP server would have a more accurate indicator that the source is spamming. Comment spam in blogs could be treated the same way with web servers asking network witnesses at clients to attest to keyboard and mouse activity before accepting comments.

B. Attesting prior actions

While attesting human use mitigates the ability for hackers to remotely launch automated attacks, it would still be possible to launch one when the user of the compromised machine is actively using the keyboard or mouse. To address this, another useful attestation would be for the network witness at the client to attest to selected prior network history. For example, in the case of spam, the witness could attest to the number of e-mail messages sent by its host in the last hour. Combined with attestation of human input, attesting e-mail transmission history can provide a significant improvement in today's handling of e-mail.

In general, there are many attacks in networks today that could benefit if attested prior history were attached.

Problem	Problem description	Witness measurement
Scanning	Adversary performs a brute-force scan on random locations to find vulnerable hosts	Witness measures and attests TCP/IP fan-out and work-weight statistics of host for easy detection [11]
Flooding	Adversary floods packets into network core forcing core routers to keep track of per-flow state in order to rate-limit effectively	Witness attaches bandwidth usage information to traffic allowing core routers to obtain a trusted measure of per-flow consumption without keeping per-flow state.
Colluding	Sybil attack used to inflate reputations or to bypass incentive-driven protocols [12]	Witness attests the number of unique IDs being used by host to trackers.
Spoofing	Adversary spoofs packets from IP addresses it does not own on the witness' host	Witness measures and attests the number of source IP addresses being used by its host to indicate abuse to others.

TABLE I
PROBLEMS MITIGATED USING ATTESTED CLIENT MEASUREMENTS FROM NETWORK WITNESSES.

For example, Sybil attacks that create multiple bogus identities, on-line poll rigging that consist of clicking a voting link multiple times, and network scanning attacks that probe target networks over time could all be thwarted if the network witness at the host were forced to attest to prior network history. Note that currently, the way to address this problem is to force services in the network to track the history of individual clients. This is problematic since doing so can result in a significant amount of state that must be kept. Thus, the use of network witnesses puts the onus on the client to attest to its relevant history when interacting with the network.

To mitigate the problem of Sybils, the witness could attest to the number of identities the end host has created on a particular site before being allowed to create additional ones. To mitigate duplicate votes in polls, the witness would attest that a particular voting link has not been visited in the past. Finally, to help intrusion detection systems identify stealthy port scans, the witness could attest to the number of unanswered connection attempts it has seen the host send to a target network over time. By shifting the burden of responsibility in tracking history from the services to the clients, a number of other attacks can be mitigated. Table I summarizes situations where attested measurements performed by a network witness could be used to aid network management.

C. Attestation-assisted congestion control

In the network core, in order to perform resource allocation in the presence of a greedy adversary, routers are often forced to keep state on the bandwidth usage history of active flows. This can be prohibitively expensive given the sheer number of flows that need to be tracked. Rather than have the network or target destination keep this information on a per-flow basis, one could instead rely on a network witness to track bandwidth usage history and to attest to its value by periodically attaching it in an authenticated manner to packets it sends in the network. For example, consider the Core-Stateless Fair-Queuing scheme in which edge routers label flows based on their bandwidth usage and core routers prioritize packets based on the labels [13]. In this scheme, the edge router is forced to keep per-flow state of all flows it sends into the core of the network and label each flow accordingly while the core network routers are effectively stateless: providing strict priority ordering based on the labels. With a network witness built into the architecture, network queuing and scheduling algorithms could instead rely on witnesses at the end host to attach such state in an authentic fashion. In the CSFQ case, the network witness at the end host would be entrusted to directly label packets with their appropriate values.

D. Validating protocol use

One way to exploit protocols is to use them in unintended ways. For example, a typical TCP session starts with a 3-way handshake using TCP SYN packets. An adversary wishing to avoid detection may bypass this mechanism by constructing fake TCP segments for connections it has not set up a priori and use the responses to determine if a service is running on a particular port [16]. One way to use a network witness running on an end host is to either report such inconsistencies or to automatically drop segments that are violating protocol rules. In the case of bogus TCP segments, it can ensure that each TCP segment it sources is consistent with and can be associated with a valid TCP session running on the host. Another interesting application of the network witness is in the detection of IP spoofing attempts. The witness could either enforce that a client only sends IP packets with a source address previously acquired via DHCP or that a client only use IP addresses that correspond to those the witness has previously seen the client receive packets using. Finally, a network witness could also be used to ensure proper behavior in application protocols. Consider the ratio cheating exploit in BitTorrent [17]. As part of the BitTorrent protocol, trackers depend on clients reporting their true upload and download statistics in order to build proper incentives for sharing. Savvy programmers have exploited this vulnerability by developing software that falsifies these reports to the tracker. In the network witness approach, one could instead use the witness to

Exploit	Exploit description	Witness validation action
Fragroute	Adversary intentionally fragments segments in order to evade or disable security mechanisms [14]	Witness at adversary disallows fragmentation unless link MTU requires it.
Xmas scan	Adversary sets invalid combinations of TCP flags to fingerprint a remote operating system based on its response	Witness at adversary drops invalid or malformed TCP segments
RST spoofing	Adversary forges a TCP RST segment in order to tear down a connection between two other hosts	Witness at adversary drops TCP segments not part of a valid local TCP connection
Optimistic ACKs [15]	Adversary acknowledges data it has not received to quickly ramp up the connection's rate	Witness at adversary drops TCP segments that violate TCP state machine
DNS poisoning	Adversary spoofs DNS replies to victim in response to its DNS requests	Witness at adversary drops DNS replies that were not preceded by a valid request to itself

TABLE II
EXPLOITS MITIGATED BY HAVING NETWORK WITNESSES ENFORCE PROTOCOL RULES.

validate the ratios being reported or to measure and report ratios directly in order to prevent cheating in the system. Table II lists some other ways that witnesses could be used to enforce proper protocol usage.

IV. EXPLORING NEW PROTOCOLS

While measuring and validating existing protocols are useful applications of network witnesses, ultimately one would like to construct new protocols that can fundamentally change how networks operate. Towards this end, we examine potential new protocols based on network witnesses that could be used in a clean-slate network design.

A. Public work

Over the last two decades, a large amount of research has been devoted to solving the problem of unwanted traffic. One approach for combating this problem is to use proof-of-work or client puzzles. With client puzzles, a client is forced to solve a computational puzzle before being given access to a resource. While many schemes currently exist for doing so [18], [19], [20], [21], most face a fundamental problem in that the issuer and the verifier of puzzles must reside at the service being protected. Such a design still allows unwanted traffic to reach most of the way to the destination.

One approach for solving this problem is the use of publicly verifiable proof-of-work or "public work" functions [22], [23]. In such schemes, as part of advertising its location, a service also supplies a work function that the client must solve in order to reach the service. While the work function is easy to generate and verify, it requires resources from the client in order to calculate a correct answer. The key to the function is that it is publicly verifiable. That is, any network device that has recorded the previous advertisement can verify whether the subsequent requests are wanted by the service and can then drop them long before they reach their destination. In our architecture, this verification can occur *at the client itself* using the network witness at the end-host. The witness can record advertisements from services that the client requests and validate that subsequent requests have valid solutions *before* being allowed into the network, allowing unwanted traffic to be dropped at the client.

B. Scheduled transmissions

While public work provides an analog knob to dial back unwanted traffic, another compelling way to stop unwanted traffic is to have destinations send a binary "cease and desist" order to particular clients which would then be enforced by the network witness. A web server under sustained attack from a botnet could advertise to each network witness running on known botnet machines a signal that indicates it no longer wants to hear from them until a particular time in the future. The witness would then enforce this by dropping the host's communication with the target destination until the advertised time.

C. Data exfiltration

One of the key problems in industry is that of data exfiltration: the unauthorized leaking of sensitive data. The ability to send a machine a piece of data and ensure that it never leaves that host is an extremely important one to most enterprises and sensitive government agencies. A network witness provides a logical place for preventing data exfiltration. As part of a protocol for transferring sensitive data, it could receive sensitive data and its exfiltration requirements, monitor system activity to track how the data is accessed, and shut down any attempts at exporting such data over the network.

D. Execute-once protocols

In the previous section, an on-line polling example is augmented with attested measurements in order to detect multiple votes from the same system. Going a step further, one interesting use of the network witness is to implement "execute-once" protocols. Consider the case of an electronic voting protocol that self-destructs upon its first successful execution. Such a property could effectively be implemented using a network witness that would detect and deny all traffic generated by a protocol after its first use.

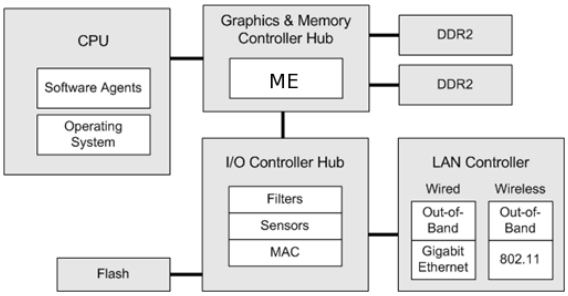

Fig. 2. Functional location of one instance of Intel AMT's Manageability Engine (ME)

V. Exploring Limits

It is important to understand the limits of the attestation approach and the consequences that such limits have on the ability to support the end scenarios described earlier.

A. Limits on threat models

The placement of the network witness within the host completely changes the threat model that can be mitigated. Figure 2 shows where the Manageability Engine of Intel's AMT platform currently sits in one generation of the PC architecture. As a result of its placement in the memory controller, it can measure the contents of memory, all network traffic going to and from the host, and all I/O going to and from peripherals. Consider the threat model addressed with this architecture in attempting to detect automation cheats in a game client [10]. In this case, the Manageability Engine monitors the electrical signals over the USB bus and ensures the keyboard and mouse input that is driving application behavior was, in fact, a result of what it measured over the USB bus (i.e., that no keystrokes were fabricated in software by the device driver or operating system). While such an approach is useful in identifying botnet and spam activity where the exploits are software-only, it can be problematic when the end user *is* the adversary. Specifically, for on-line games, the cheater can move the automation out of the target host's software stack and onto the *other side* of the I/O bus by building a USB device that generates the desired electronic signals across the bus to pull off the automation attack. Consider how the threat model would change if a data authenticity capability were placed in the keyboard/mouse itself (i.e., secure peripherals). Using this approach of peripherals authenticating the physical motion of the device, even if the end user was the adversary, he/she would then be forced to physically move the peripheral with a robot to pull off the automation attack undetected.

Another limiting aspect of the witness approach is its placement at network endpoints. As a result of this, adversaries in the network can selectively drop attestations or use a reflector attack to force a flood of attested measurements into the network to disrupt the system. For the approach to work, it is important that such attacks on witnesses be effectively mitigated.

B. Limits on accuracy

While treating network witnesses as an omniscient entity on the client is convenient for research purposes, in practice, existing solutions have physical limitations that prevent certain types of measurement. For example, although Intel's Manageability Engine sits off of the front-side bus of the main CPU, it cannot measure all memory transactions going through the FSB since the ME runs at a *fraction* of the speed to keep costs low. This is a key limitation in its current use, since it is unable to reliably measure what is effectively a moving target (i.e., the memory image of the running applications and operating systems).

It is essential that the witness can guarantee that its attested measurements are correct since an adversary will always identify and exploit any limits on a system to his/her benefit. For example, consider an attack where an adversary floods the network witness with events it must measure and attempts to slip a request through undetected. Being able to understand how attested measurements might degrade under such attack is important in determining the quality and correctness of what is being attested. For the approach to work, the witness *must not* provide an incorrect attestation otherwise all protocols built around such attestations are then rendered useless.

C. Limits on storing attested data

Another important limit is the amount of attested data a network witness can and is willing to store. Attesting to every packet that has ever been sent by the end system is prohibitively expensive for the witness to do, as is keeping attested measurements an indefinite period of time. Protocols based on an attestation approach must explicitly account for situations where the end client is unable to attest to the desired measurement. Towards this end, we will consider how one might manage attested data and how to build protocols to limit the amount of attested measurements are performed and kept. For example, with a voting protocol, one might attest to a particular event occurring over a specific window (i.e., ensuring that the presidential "voting" URL is only visited once every four years).

D. Limits from the user

Because users must be able to control what is going on with their data, it is important that *what* network witnesses measure is transparent and configurable to them. Specifically, to ensure privacy protection, users must have the ability to configure what they are willing

to attest to and to "opt-out" of protocols that require attestation of information they do not want to divulge. For example, not many users would be willing to participate in a protocol that requires the attestation of every keystroke made by the user. However, some users may be willing to have the witness attest to the whether or not their keyboard has been used in the last hour if this meant reducing their level of spam. Key to the witness approach is the trade-off between a user's privacy and the level of detail given in attested measurements.

E. Limits on deployment

Legacy systems that do not possess the network witness capabilities pose an issue that may limit the motivation for deployment of systems with network witnesses enabled. This issue can be mitigated by vendors (ISPs, networked application service providers, etc) providing incentives to users who enable network witnesses on their systems. For example, since an ISP benefits from its subscribers verifiably adhering to network protocol standards, it could give users a reward such as a bandwidth increase or an extra email account for enabling witnesses. Another example incentive would be a web portal that does away with CAPTCHA tests for users that are willing to use the witness to attest that their keyboard/mouse are active.

F. Limits on extensibility

As shown in Figure 2, current hardware-isolated execution environments that fit the network witness profile sit off-chip and can only measure a range of buses attached to the main CPU. As a result, exploits involving internal hardware CPU registers, such as overwriting the interrupt descriptor table register or structured exception handling registers escape the purview of these hardware components. Any attestation approach must be able to support future measurement capabilities that can be added. For example, if a witness could eventually be implemented within the CPU, then the network protocols must eventually be able to take advantage of increased visibility into CPU state.

VI. Summary

The preceding analysis shows that a hardware component possessing the properties of reliable introspection, isolation, tamper resistance, and the ability to only execute signed code can provide a fundamental new building block in the design of secure network protocols. A network witness possessing these characteristics could be used to attest to the correct use of existing protocols by attesting to human input, attesting to prior platform actions, assisting with congestion control, and validating protocol use. Additionally, the presence of a network witness in all network endpoints could allow for the development of new, more secure network protocols based on public work, transmission scheduling, and one-time execution. The use of a network witness for these purposes should be explored in more detail to fully evaluate its effectiveness in these roles.

References

[1] Intel, "Intel Active Management Technology," http://www.intel.com/technology/platform-technology/intel-amt/.

[2] J. Evers, "Taking on Rootkits with Hardware," December 2005, http://news.cnet.com/Taking-on-rootkits-with-hardware/2008-1029_3-5992309.html.

[3] "Intel System Defense Technology," http://www.intel.com/cd/ids/developer/asmo-na/eng/320960.htm.

[4] D. DeLiberato, H. Khosravi, D. Valdez, and L. Webb, "Intel and Cisco Collaborate to Improve Enterprise Security," *Technology@Intel Magazine*, September 2005.

[5] Trusted Computing Group, "Trusted Network Connect," http://www.trustedcomputinggroup.org.

[6] "DDoS on Blue Security Blog Knocks Typepad, LiveJournal Offline," May 2006, http://news.netcraft.com/archives/2006/05/03/ddos_on_blue_security_blog_knocks_typepad_livejournal_offline.html.

[7] B. Brenner, "Gmail CAPTCHA Cracking Leads to Spam Surge," March 2008, http://searchsecurity.techtarget.com/news/article/0,289142,sid14_gci1304779,00.html.

[8] G. Hoglund and G. McGraw, *Exploiting Online Games: Cheating Massively Distributed Systems*, Addison-Wesley, 2007.

[9] "PWNtcha - captcha decoder," http://sam.zoy.org/pwntcha.

[10] T. Schluessler, E. Johnson, and S. Goglin, "Is a Bot at the Controls - Detecting Input Data Attacks," in *NetGames*, October 2007.

[11] J. Binkley and S. Singh, "An Algorithm for Anomaly-based Botnet Detection," in *USENIX Conference on Steps to Reducing Unwanted Traffic on the Internet*, July 2006.

[12] B. Cohen, "BitTorrent," http://www.bitconjurer.org/BitTorrent.

[13] I. Stoica, S. Shenker, and H. Zhang, "Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks," in *ACM SIGCOMM*, September 1998.

[14] "fragroute," http://www.monkey.org/~dugsong/fragroute.

[15] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson, "TCP Congestion Control with a Misbehaving Receiver," *ACM SIGCOMM CCR*, vol. 29, no. 5, April 1999.

[16] Fyodor, "Remote os detection via tcp/ip stack fingerprinting," http://www.insecure.org/nmap/, 1998.

[17] "How to Cheat BitTorrent Ratio by Spoofing," http://www.raymond.cc/blog/archives/2006/07/27/how-to-cheat-bittorrent-ratio-by-spoofing/.

[18] C. Dwork and M. Naor, "Pricing via Processing or Combatting Junk Mail," in *CRYPTO*, August 1992.

[19] A. Juels and J. Brainard, "Client Puzzles: A Cryptographic Defense Against Connection Depletion," in *NDSS*, February 1999.

[20] X. Wang and M. Reiter, "Defending Against Denial-of-Service Attacks with Puzzle Auctions," in *IEEE Symposium on Security and Privacy (S&P)*, May 2003.

[21] W. Feng, "The Case for TCP/IP Puzzles," in *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)*, August 2003.

[22] W. Feng and E. Kaiser, "The Case for Public Work," in *IEEE Global Internet Symposium*, May 2007.

[23] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y. Hu, "Portcullis: Protecting Connection Setup from Denial-of-Capability Attacks," in *ACM SIGCOMM*, August 2007.