Safely Harnessing Wide Area Surrogate Computing -or- How to Avoid Building the Perfect Platform for Network Attacks

Sachin Goyal and John Carter
School of Computing, University of Utah
{*sgoyal, retrac*}*@cs.utah.edu*

Abstract

We are building a wide area surrogate computing platform, called WASCo, that allows users to dynamically locate, allocate, and exploit resources on surrogate computers spread around the Internet. WASCo allows clients to install and run arbitrary code on surrogates. Uses of WASCo include offloading energy-intensive operations from energy-constrained devices, executing bandwidth-intensive queries near a large data source, creating a dynamic resilient overlay network (RON) to route around internet problems, or instantiating a new web server near underserved clients in response to a spike in traffic.

While a surrogate infrastructure like WASCo has a huge number of beneficial uses, the ability to run arbitrary code on surrogates distributed across the internet could make it the perfect platform for spammers, hackers, content thieves, and other nefarious individuals. In this paper, we discuss the network security issues that must be addressed before wide area surrogate systems can be safely deployed and present our solutions. We show how a combination of virtual machine technology for local resource controls, network security implemented at the virtual machine monitor level, and a trust infrastructure can address the key security problems.

1 Introduction

We envision a future where the compute, storage, and networking resources of myriad cheap network-connected computers distributed around the world will be made available to arbitrary clients or devices by a mix of open p2p sharing and commercial companies. Users will be able to dynamically locate, allocate, and exploit these resources on-demand when and where they are needed. Towards this goal, we are building a wide area surrogate computing infrastructure, called WASCo, that allows users to allocate complete virtual servers, including `root` access, on surrogate machines distributed across the internet. We believe that WASCo will be a powerful platform for enabling a wide variety of distributed or pervasive services. For example, a PDA could offload compute- and energy-intensive operations to a nearby tethered server to reduce the load on its battery. A scientist could create a dynamic compute grid of servers located around the world to solve a particularly compute-intensive problem. Another scientist might wish to move a computation to a surrogate located near a large data source to avoid shipping an immense amount of data over the WAN. Or, a webmaster might witness a surge of web requests from a currently underserved part of the world and want to dynamically create a site mirror on a server near the new traffic.

Unfortunately, while the potential uses of an open surrogate infrastructure are almost endless, the potential misuses are equally endless if there are insufficient security measures to keep spammers, hackers, content thieves, and other nefarious individuals at bay. For example, a spammer could use the surrogate infrastructure to create a new spambot in response to a system administrator adding the old spambot's IP address to a blacklist. A hacker could instantiate a cracking tool inside of a company's firewall or create an army of bots from which to launch a distributed denial of service attack. A content thief could create an open relay on a machine owned by an organization with a subscription to some data source and use it to access the data. In general, giving miscreants unfettered access to surrogate computers at strategic locations in the Internet would create serious security threats [15]. Even unfettered access to systems with restricted capabilities, e.g., the CoDeeN open proxy infrastructure [13], can introduce myriad security problems. For a wide area surrogate infrastructure to attain wide acceptability, it must contain sufficient security mechanisms to make misuses infeasible.

The Spectra/Chroma [4] and Xenoservers [14] projects, along with research on the computational grid [10], have demonstrated the benefits of using dis-

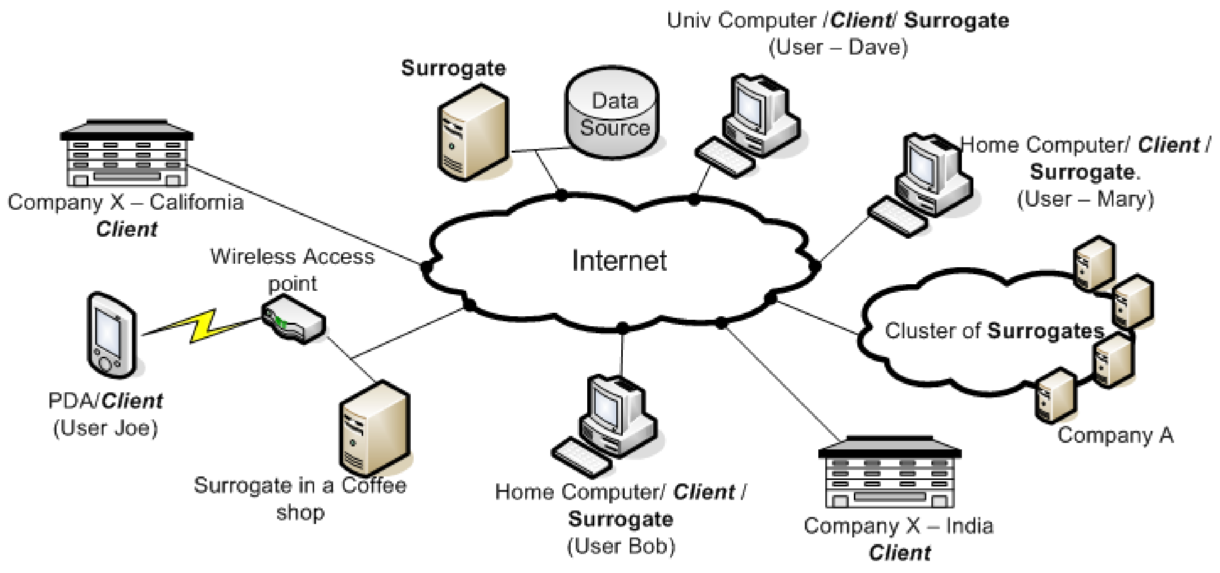

Figure 1: A Wide Area Surrogate Computing Scene

tributed compute resources. However, unlike Spectra/Chroma and the grid, WASCo is an open system that allows potentially untrusted clients to install and run arbitrary code on surrogates. Unlike Xenoservers, WASCo does not assume a trust relationship between clients and surrogates and thus needs to guard against network attacks initiated by surrogate clients.

In this paper, we describe the security mechanisms that we are including in WASCo to prevent its misuse. WASCo employs Xen virtual machine technology to allocate complete virtual servers to clients [5] and to control a client's access to the surrogate's local resources. We enforce network security by controlling the network access of surrogate VMs at the virtual machine monitor (VMM) layer. We show that we can address the security concerns outlined above without negating the value of a general surrogate infrastructure by selectively restricting surrogate VM network access, employing IP tunneling, and rate limiting surrogate network traffic. The degree to which a surrogate's network access is restricted is governed by the trust relationship between a client and the surrogate server. For example, a surrogate running on a user's home machine will have no restrictions, while a surrogate running code on behalf of a completely untrusted third party will have severely restricted network access so that it cannot be used for nefarious purposes.

2 WASCo

In this section, we outline how WASCo can be used and differentiate it from related work. We envision people making their home/office computers available as surrogates, perhaps only for trusted friends, and commercial entities making surrogates available for a fee. In WASCo, each surrogate computer runs a virtual machine monitor and allocates entire virtual servers to clients, complete with `root` access and a unique IP address. The utility of a virtual server may derive from its computing and storage resources and/or its location in the Internet. Figure 1 illustrates four example uses of WASCo.

Joe is relaxing in a cafe. His PDA is connected to the Internet via the cafe's Wifi hub. The cafe has installed WASCo on a PC so that its customers also have access to compute resources. Joe likes to voice-control his PDA, but running voice recognition on the PDA is far slower than real-time and quickly drains the battery. Instead, upon locating the available surrogate, his PDA instantiates a voice recognition service on the cafe's PC and sends it raw audio streams for interpretation. Offloading work to a tethered PC enables real-time voice recognition without draining the PDA's battery [11].

Dave is an astronomer who wants to data mine the multi-terabyte Sloan Digital Skyserver database to prove a provocative new theory. Rather than stream terabytes of data across the Internet or ask Jim Gray to FedEx a TeraScale SneakerNet box [2] with the data, Dave accesses a WASCo surrogate located on the SkyServer SAN. From that surrogate, his analysis tool can mine terabytes of data without impacting either his or the Skyserver's limited Internet bandwidth. Taking this example one step farther, after using the SkyServer surrogate to extract the relevant data, Dave might need to perform substantial computation to determine if the data matches his theory. Rather than perform the entire computation on the SkyServer surrogate, Dave could instantiate a computational grid [10] on surrogate computers located around the internet by downloading and installing grid middleware on available surrogates.

Bob and Mary are playing a network game. Rather than using a public game server, which are often overrun by hormone-drenched teenagers, or running a private game server on one of their home PCs, which might give an advantage to the player who is running the game server, they instantiate a private game server on an intermediate node roughly equidistant from both of them.

Company X has offices in India and California. Their system administrator has been instructed to ensure good connectivity between the two sites to ensure that network problems do not cause them to miss a crucial deadline later today. In response, the sysadmin rents cycles on a surrogate machine provided by Company A and twenty other surrogates not shown in Figure 1, and uses them to build a resilient overlay network (RON) [3].

WASCo lets clients run arbitrary programs on surrogates and does not restrict clients to using only machines on which they are completely trusted. Without appropriate security precautions, WASCo could be used by hackers to bypass firewalls or launch denial of service attacks, by spammers to create spambots, or by content thieves to access and distribute private or copyrighted data. Staniford et. al. [15] discuss the risks associated with attackers who are able to gain control of vast number of Internet hosts. Pai et. al.'s experience with the CoDeeN open content distribution proxy [13] demonstrates the security risks that arise even when the functionality available on an open surrogate is restricted. In contrast, most systems that enable users to harness the widely distributed compute and storage resources of a large number of organizations, e.g., PlanetLab [6] or the computational grid [10], are made available only to trusted users and/or only run trusted programs. Their security model is based on a simple access control model. In WASCo, however, we must address the problems that arise when untrusted users can run untrusted code.

The Xenoservers project [14] is probably the closest in spirit to our work. However, Xenoserver requires a trust relationship between clients and surrogates [7], whereas WASCo supports unknown and untrusted users. Like Xenoservers, we build upon Xen virtual machine (VM) technology for single-node resource controls [5], and thus can allow clients to install and run arbitrary code without fear that it will interfere with other clients sharing the surrogate host. Virtual machine technology provides a simple, yet powerful, substrate on which to build surrogates [9]. In this paper, we discuss how to extend the basic resource encapsulation of VMs to avoid the network security risks that a truly open surrogate infrastructure would create.

3 WASCo Network Security

The open nature of WASCo requires us to pay special attention to security. We use VM sandboxing to restrict what local resources a client can consume, but a malicious client could misuse the virtual server to launch network attacks as described earlier. The goal of WASCo's network security system is to prevent such misuse.

WASCo enforces network security at the virtual machine monitor (VMM) layer where it can monitor and control the network traffic going to/from a virtual server. Our guiding principle is to restrict network access from/to a virtual server depending on client-specified needs and the degree to which the surrogate trusts the client. The less a surrogate trusts a particular client, the more highly it restricts the client's network access.

From the perspective of a WASCo surrogate, a client falls into one of three trust categories: (i) *completely trusted*, (ii) *semi-trusted*, and (iii) *untrusted*. Trust derives from a variety of factors. Administrative relationships (e.g., machines belonging to a common organization), financial relationships (e.g., machines for which a user has rented access), and trust chains can all impact the trust assigned to a particular surrogate request.

A user's home/office machine will typically *completely trust* surrogate requests from the user. Virtual servers owned by completely trusted users can communicate with any node without restriction.

A university surrogate might consider any request originating from within the university to be *semi-trusted*; similarly, a commercial surrogate may consider any request for which a client is paying to be semi-trusted. The basic security policy for semi-trusted users is "trust, but verify". Semi-trusted users' network access is based on policies defined by the surrogate's administrator. For example, a SkySurvey surrogate server might allow rate-controlled access only to the IP address and port on which the database server is listening and the client device. The surrogate manager can log network traffic by semi-trusted clients to analyze potentially malicious behaviors.

Finally, requests from unknown clients, e.g., somebody using an open surrogate provided by a p2p system, will typically be completely *untrusted*. By default, untrusted surrogates are allowed direct access only to the client node from which the surrogate was requested. This design is similar to the restrictions imposed on Java Applets. For many applications, e.g., the surrogate used to offload speech recognition from a PDA, this highly restricted network model is sufficient once the surrogate has downloaded and instantiated the server software from a specified data source [11]. Untrusted users can get access to arbitrary other nodes by tunneling through the client or by being given explicit permission to access a particular remote node, as described below.

Tunneling

To limit the potential damage of an untrusted surrogate, but let it access arbitrary internet nodes, WASCo supports IP tunneling from the virtual server to the client computer. Clients can use software network address translation (NAT) to masquerade as the client from the perspective of other nodes, and forward packets from/to the virtual server via the allowed direct connection. This mechanism is akin to a virtual private network (VPN) or mobile IP. Because all packets sent between the surrogate and nodes other than the client are routed via the client, the client is unable to use the surrogate to circumvent firewall access controls or launch anonymous network attacks. VNET employs a similar technique for grid computing [16].

Accessing Authorized Nodes

In addition to tunneling, virtual servers can negotiate for network access to nodes with whom their user has a trust relationship. This scheme can be used to bootstrap interesting applications, e.g., Company X in Figure 1 can use this to authorize traffic between the rented virtual servers over which it wishes to establish the RON overlay. To do so, a user provides an authorization certificate (*authcert*), which the surrogate manager uses to determine if the virtual server should be allowed access to the remote node. We assume the existence of a public key infrastructure (PKI) that binds a node's IP address(es) to a public key. The same public key can be bound to multiple IP addresses, which we use to allow surrogate managers to sign certificates on behalf of their virtual servers.

Suppose A is a normal node. In this case, a *authcert* that allows the user open access to A has the form: $\langle K_{pubA}, \langle A_{ip}, K_{user} \rangle_{K_{privA}} \rangle$, where A_{ip} is the IP address of node A, K_{user} is the user's public key, and K_{pubA}/K_{privA} are node A's public and private keys, respectively. By signing the authcert, node A indicates that it is willing to accept connections from surrogates belonging to K_{user}. In response to a request by a client to allow communication with node A, the surrogate manager first verifies that K_{pubA} is indeed the public key of node A. Once the surrogate manager has validated K_{pubA}, it verifies that the authcert is signed by node A before allowing the untrusted client access to node A.

Now suppose node A is a virtual server running on a surrogate S_A. The PKI will contain information about S_A, but not A, so a surrogate manager cannot verify an authcert issued by (virtual) node A. In this case, the authcert must include K_{pubA} signed by S_A. The surrogate manager can use this information and the PKI to verify the identity of S_A, to verify that S_A has signed K_{pubA}, and to verify that A_{ip} is an IP address that server S_A can allocate to its surrogates. Because virtual server's have limited lifespans, an authcert signed by a surrogate manager also needs to have a limited lifespan. This can be accomplished in a number of ways, e.g., by adding an expiration time on the signed certificate (which requires loosely synchronized clocks), by having the surrogate manager on the node requesting access periodically poll S_A to verify that A is still a valid virtual server, or by having the requesting node register a callback with S_A so that it will be informed when virtual node A terminates.

Traffic Shaping and Rate Limiting

In addition to denying semi-trusted and untrusted users access to arbitrary remote nodes, we limit the maximum bandwidth available to untrusted virtual servers. The surrogate manager can fairly allocate the local node's bandwidth between the surrogate host and various client virtual servers. Thus, a rogue virtual server cannot interfere with other virtual server's network access, nor can it effectively launch a DoS attack on the surrogate node's LAN/WAN connection by flooding packets to authorized nodes (e.g., the client). It also limits the extent to which semi-trusted users, who are allowed open access to a limited number of nodes, can misuse the surrogate. WASCo enables the administrator to set different bandwidth limits to different flows. For example, in the SkyServer surrogate example the administrator might allow very high bandwidth between the surrogate and the SkyServer database, but limited bandwidth out the SkyServer WAN connection.

Attack Detection and Logging

The surrogate VMM can monitor all network traffic to/from a virtual server and perform sophisticated attack detection, e.g., ReVirt [8]. In addition, the surrogate manager can log which nodes a particular surrogate connects to and its source client, which can be used to identify the true culprit should a surrogate launch a network attack. Finally, should a virtual server be identified as launching a network attack, the VMM can remove its network access without destroying the virtual server, there enabling postmortem analysis of the attack software [17].

4 Network Security Implementation

The current WASCo prototype employs the Xen [5] virtual machine system to support multiple virtual servers per surrogate node. Xen para-virtualizes an x86 platform, which allows multiple independent OS kernels (currently either Linux or FreeBSD) to run on a single machine. In the latest version of Xen ($>$1.2), the network driver runs in the controller virtual machine domain, and is responsible for creating virtual interfaces for the other virtual machines and connecting them and the physical network card via a software bridge. Because the network driver runs in the controller virtual machine domain, clients cannot circumvent WASCo's network access controls even if they patch or reconfigure their guest OS kernel.

WASCo's basic network security mechanisms are built using off the shelf Linux tools. Linux supports sophisticated network packet filtering and traffic shaping. We use Linux's Netfilter/iptable [1] mechanism to filter packets as they pass through the network stack. Each virtual server is configured with its own ruleset, which allows the surrogate manager to restrict which nodes a particular virtual server can contact based on trust level and authorization certificates.

To manage the amount of bandwidth that each virtual server has between arbitrary nodes, we use the Linux traffic control system (*tc*) [12]. The surrogate manager can specify for each virtual server the maximum aggregate outgoing bandwidth that it can send. Using classful queuing disciplines and `iptable` together, the surrogate manager can classify different traffic flows from a single virtual server and assign different bandwidth limits to each flow, e.g., in the SkyServer example it can allow unlimited bandwidth to the SkyServer database, but limited bandwidth out via the WAN connection.

To implement tunneling, we use GRE tunneling [12], available as part of Linux, to establish IP-in-IP tunneling back to the client. The virtual server sees two network interfaces, one virtual interface with an IP address in the surrogate domain and a tunnel interface with an IP address specified by the client. IP routes are configured to use the tunnel interface by default and the surrogate interface for specific IP addresses (e.g., to the local SkyServer database).

WASCo currently does not employ a logging/intrusion detection mechanism in the VMM, but we are considering adding one in the future.

5 Conclusion and Future Work

In this paper, we first identify a variety of ways that a wide-area open surrogate architecture could be misused and then describe how to build a flexible network security system that mitigates these problems via selective packet filtering, IP tunneling, and bandwidth limiting. Our solution uses a mix of available tools, including the Xen virtual machine system and a variety of network filtering tools available in recent Linux releases.

To the best of our knowledge, WASCo is the first wide area surrogate computing infrastructure that supports fully trusted, semi-trusted, and untrusted clients running arbitrary code on surrogates. Basing WASCo on virtual machine technology protects WASCo hosts from untrusted client code and allows WASCo to implement a sophisticated network security system by monitoring and controlling all traffic to/from each virtual server. We show that while a naive wide-area open surrogate computing could be used to launch devastating network attacks that are not possible in closed domains like grid systems, the problems can be solved and we believe the benefits of openness and generality outweigh the risks.

A medium term goal of our work is to release a publicly usable surrogate computing system and deploy surrogates nodes around the world. We plan to more clearly define what limitations should be applied to semi-trusted users and develop a simple yet comprehensive set of security options for system administrators based on different trust levels. We will consider reputation systems as a means for deriving trust levels [7]. We plan to investigate effective ways to integrate traffic logging and mechanisms to automatically detect malicious network usage patterns so that we can reduce the network access restrictions imposed on semi-trusted and untrusted users. Finally, we will evaluate and update WASCo's security infrastructure based on the results of real-life usage.

References

[1] The netfilter/iptables project, http://www.netfilter.org.

[2] A conversation with Jim Gray. *ACM Queue*, June 2003.

[3] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, October 2001.

[4] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.-I. Yang. The case for cyber foraging. In *10th ACM SIGOPS European Workshop*, 2002.

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, 2003.

[6] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Computer Communication Review*, 33(3), 2003.

[7] B. Dragovic, S. Hand, T. Harris, E. Kotsovinos, and A. Twigg. Managing trust and reputation in the xenoserver open platform. In *Proceedings of the 1st International Conference on Trust Management*, May 2003.

[8] G. W. Dunlap, S. T. King, S. Cinar, M. Basrai, and P. M. Chen. Revirt: Enabling intrusion analysis through virtual-machine logging and replay. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[9] R. Figueiredo, P. Dinda, and J. Fortes. A case for grid computing on virtual machines. In *Proceedings of the 23rd Internatinal Conference on Distributed Computing Systems*, 2003.

[10] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid - enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15(3), 2001.

[11] S. Goyal and J. Carter. A lightweight secure cyber foraging infrastructure for resource-constrained devices. In *Sixth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 2004.

[12] B. Hubert. Linux advanced routing and traffic control howto, http://lartc.org.

[13] V. S. Pai, L. Wang, K. Park, R. Pang, and L. Peterson. The dark side of the web: An open proxy's view. In *Proceedings of the Second Workshop on Hot Topics in Networking (HotNets-II)*, 2003.

[14] D. Reed, I. Pratt, P. Menage, S. Early, and N. Stratford. Xenoservers: Accounted execution of untrusted code. In *IEEE Hot Topics in Operating Systems (HotOS) VII*, March 1999.

[15] A. Sundararaj and P. Dinda. Towards virtual networks for virtual machine grid computing. In *Proceedings of the third USENIX Virtual Machine Research and Technology Symposium (VM 04)*, 2004.

[16] N. Weaver, S. Staniford, and V. Paxson. Very fast containment of scanning worms. In *Proceedings of the 13th USENIX Security Symposium*, 2004.