Network Diversity – Implementation and Realization

Rishi Pal Bangarh[1] Dr. K.K. Jain[2]

[1]Research Scholar of Singhania University

[2]Asst. Prof. , P.G.V. College Gwalior (M.P.)

----------♦----------

THE SOFTWARE DIVERSITY COMPROMISE

It is time for parents to teach young people early on that in diversity there is beauty and there is strength.

GENERATING DIVERSITY

In order to counteract the lack of diversity in the Internet, researchers have focused on the method of diversifying pre-existing architectures, source code, and binaries in order to artificially generate a diversity of software packages. In general, we can classify the points at which diversity can be applied into the following categories: Requirements, Architecture, Implementation, and Realization. While other classification schemes of diversity techniques have been presented, we are less interested in the managerial aspect of applying diversity to entire business processes, and more concerned with diversity implementation schemes.

During the Requirements phase, early design considerations which provide diverse methods of interacting with networked devices, processing information, and interacting with the user can be factored into the initial requirements document. Schemes which generate a loose functional equivalence between different binaries would be applied during this stage . In a similar vein, the Architecture of the software architecture can be varied to allow for different data flows and process interaction, while still maintaining a standardized software interface.

The majority of the diversity schemes present in the literature consider how diversification can be applied during the Implementation and Realization phases of the software development cycle. The Implementation phase allows for source code to be modified in an algorithmic fashion, for the software to be built using different programming languages, and for the software to be built by independent teams of developers using the same language.

As proposed by Forrest, Somayaji, and Ackley, automated techniques which manipulate source code by reordering source code, adding and removing non-functional code, or changing the linking order of dynamic libraries can be utilized. Researchers working on preventing reverse engineering of binaries have developed code obfuscation techniques which can also be used to diversify software packages. A technique for obfuscating Java source code, which uses similar code reordering techniques proposed by Forrest, is presented in After code implementation, the final Realization, or build and execution, of the software can be modified through a wide variety of techniques, including the compiler-driven randomization techniques . In fact, many of the code reordering techniques which provide memory randomization functionality can be applied at runtime after a binary has been created . At the final stage of development, the instruction set used can be diversified without a wholesale switch of system architectures Both systems serve the same purpose by converting maliciously injected code into binary strings which have little meaning for the processor. Additionally, both techniques are not without practical precedent, as a similar technique was proposed by Cowan et al. for protecting pointers in memory . Both forms of artificial instruction set randomization appear to be broken , however, due to irregularities in the byte size of each opcode present in the x86 platform.

The code reordering and reforming techniques are expanded upon in for the purpose of obfuscating Java code against reverse engineering. Wang and her coauthors describe code modification techniques for use in protecting high-availability mechanisms which are currently employed in server systems. The compile-time techniques discussed are readily available for download, and have found their way into open source operating system distributions. Address space randomization is implemented in the Linux PaX toolkit and compile time randomization of stack offsets has been implemented in GCC . It has been

pointed out that address space randomization doesn't work as well as predicted in architectures with smaller address spaces due to the fact that large portions of the address space are reserved by the operating system, and are not accessible for user-land memory addressing .

THE CASE FOR ASSIGNING DIVERSITY

The attacks discussed against the publicly available diversity generation techniques undermines the assumption that a diverse pool of software can be created at a low cost. Furthermore, an analysis of POSIX-compliant operating systems showed that faults were highly correlated across different vendor's platforms, with the majority of common faults existing in upper-level functionality, such as C libraries. In general, as we descend from the high level components of a system through the core and into the original architecture specifications, software diversity becomes both more expensive to implement, and more effective against common faults. We are forced to conclude that the cost of generating a set of truly diverse software packages makes diversity a scarce resource which must be carefully and consciously allocated in order for it to be maximally effective against attackers. For a single host, choosing the optimal set of diversity techniques and diverse software packages resolves down to a problem of economics. The benefit side of the equation consists of creating a system which is different enough from the global population of computers that an attack against any one system would be difficult to port to be effective against the diversified system. Each of the diverse software packages, source level, and compilerdriven diversity techniques have a associated cost figure, as they either cost money to purchase, decrease computing speed, or increase the amount of administration time required for patching and general system maintenance.

The burden of creating a host which is considered to be diversified as compared to all other hosts on the Internet is massive, but it is not one faced by a network administrator who has control over a large pool of systems. The network administrator's diversification task is not equivalent to solving the single host diversification problem for every machine on their network. Unlike the single host's administrator, a network administrator is able to leverage the restrictions placed on an attacker by the network topology in order to reduce the number of diverse software packages necessary. This is the fundamental thesis of our work: by taking the topology presented to an attacker into account, an assignment of a small number of diverse software systems can be formulated which can slow or stop an attacker in their tracks.

While it may be argued that the network topology traversed by an attacker is a complete graph, and every machine must be made diverse and separate from every other machine on the network, this statement is not true even for IP-level connectivity. The prevalence of firewalls and private address spaces prevent any machine from connecting to any other machine on the Internet. Furthermore, not every attack exploits IP-level connectivity for propagation. Worms which spread by traversing individual e-mail address books move through a network topology which is emarkably sparse, and client-server file sharing worms inhabit graphs which are largely bipartite.

EXAMPLES OF NETWORK DIVERSITY ASSIGNMENTS

E-Mail Topologies: Any individual that utilizes e-mail has become a target of selfpropagating code. Vulnerabilities associated with the default configurations of MIME han14 Client . The effect of optimally distributing two software packages on a bipartite network is clear in (a) and (b). Bipartite networks such as these are often found in client-server file sharing topologies. dlers have given rise to client-side computer viruses . Errors in the parsing code in major mail transfer agents have resulted in server-side attacks that are also propagated via e-mail traffic [55]. Secure diversity can be implemented in the stated situation through the utilization of interchangeable MIME and e-mail header parsers which are selected by the application based upon a topology-sensitive algorithm. Replacing one parser library with another would have no user-discernible impact on the software's behavior and performance.

Client-Server File Shares: Network-accessible file shares have become a popular target for platform-dependent worm propagation . In many office environments, the file shares are partitioned into the client and server groups . A random network topology clearly benefits from a random distribution of three heterogeneous software packages as compared to a uniform distribution of a single package . While the assignment is sub-optimal, the number of edges which exist between nodes running similar software packages is clearly reduced.

Communication links between similar systems are represented by a solid line. This partitioning can be enforced using firewalls and ACLs. A worm infection on a client system would be able to self-propagate to any machine in the file-sharing topology by first attacking a server machine; likewise, a worm infection on a server would have to first attack a client before propagating further. The secure diversity principle can be quite effectively applied to such a network with only two different software packages. All previous communication links between similar systems are replaced by links between

dissimilar computers, represented by the dotted lines . By utilizing a second software package for file sharing on the server systems, it is possible to prevent a client system from propagating a worm that attacks a vulnerability in the file sharing subsystem.

Sensor Networks: The networking field that would benefit greatly from the secure diversity principle is sensor networks . Enforcing a diversity policy in a sensor network is less of an administrative challenge, since these large networks of relatively simple computational and environmental monitoring nodes are usually controlled by a single entity, be it a military commander or a building supervisor. Because the hardware is characterized as being relatively simple, it is not a major technical challenge to recreate their comparatively small software suite for the purposes of introducing variation between individuals in the population.

Consider the possibility of a system-wide vulnerability that allows for an attacker to take over a single networked sensor. A single attack can be used to leap-frog from node to node across the entire network, as indicated by the bidirectional links.

Sensor networks can be distributed with multiple operating systems in ROM. After being dropped into the operational location, a node can load up one of a multiple set of OSes. By constructing a network that contains a multiplicity of operating systems, a single operating system-specific attack will not be able to propagate across the entire breadth of the network. Such a randomized distribution of software packages can reduce the number of possible node-to-node movements by an attacker.

Reasoning about Diversity While the concept of diversity assignment schemes may be philosophically appealing, currently there is no formal system available for reasoning about diversity assignments. In the following chapter, we provide a framework that abstracts both the generation and attacking of diverse software packages.

REFERENCES

[1] WLAN Association , “Introduction to Wireless LANs ” , WLANA Resource Center , 1999 , http:// www.wlana.com/learn/intro.pdf

[2] John Vollbrecht , David Rago , and Robert Moskowitz. “Wireless LAN Access Control and Authentication”,White Papers at Interlink Networks Resource Library, 2001.http://www.interlinknetworks.com/resource/WLAN_Access_Control.pdf

[3] WLAN Association , “Wireless Networking Standards and Organizations ” WLANA Resource Center, April 17, 2002.

http://www.wlana.com/pdf/wlan_standards_orgs.pdf

[4] Interlink Networks , “Wireless LAN Security using Interlink Networks RAD Series AAA Server and Cisco EAP-LEAP ” , Application Notes at Interlink Networks Resource Library, 2002.

http://interlinknetworks.com/images/resource/wireless_lan_security.pdf

[5] Jesse R. Walker , “Unsafe at any key size ; An analysis of the WEP encapsulation ” , 802.11 Security Papers at NetSys.com , Oct 27 , 2000.

http://www.netsys.com/library/papers/walker-2000-10-27.pdf.

[6] Interlink Networks , “Introduction to 802.1 x for Wireless Local Area Networks ” , White Papers at Interlink Networks Resource Library , 2002.

http://www.interlinknetworks.com/resource/Wireless_LAN.pdf

[7] Pierre Trudeau , “Building Secure Wireless Local Area Networks ” , White Papers at Colubris.com,2001.

http:// download.colubris.com/library/WP-010712-EN-0100.pdf.

[8] Jean – Paul Saindon , “Techniques to resolve 802.11 and wireless LAN technology in outdoor environments” , News Article at SecurityMagazine.com, Aug 08, 2002