Crumple Zones: Absorbing Attack Effects Before They Become a Problem

Article · April 2011

CITATIONS
6

READS
215

9 authors, including:

Michael Atighetchi
Raytheon BBN Technologies
90 PUBLICATIONS **998** CITATIONS
SEE PROFILE

Partha Pal
Raytheon BBN Technologies
123 PUBLICATIONS **1,155** CITATIONS
SEE PROFILE

Fusun Yaman
Raytheon BBN Technologies
52 PUBLICATIONS **1,360** CITATIONS
SEE PROFILE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
AUG 2011	**Journal Article** (Post Print)	SEP 2009 – JAN 2011

4. TITLE AND SUBTITLE

CRUMPLE ZONES: ABSORBING ATTACK EFFECTS BEFORE THEY BECOME A PROBLEM

5a. CONTRACT NUMBER
FA8750-09-C-0216

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62788F

6. AUTHOR(S)

(AFRL) Asher Sinclair, (BBN) Joseph P. Loyall, Michael Atighetchi, Partha Pal, Aaron Adler, Jonathan Webb, Andrew Gronosky, Fusun Yaman (Adventium Labs) Charles Payne.

5d. PROJECT NUMBER
558S

5e. TASK NUMBER
AP

5f. WORK UNIT NUMBER
SV

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

BBN Technologies; 10Moulton Street, Cambridge, MA 02138
Adventium Labs; 111 Third Avenue South, Suite 100, Minneapolis, MN 55401

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RISE
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TP-2011-29

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA #: 88ABW-2011-0352
DATE CLEARED: 27 JAN 2011

13. SUPPLEMENTARY NOTES
Publication in CrossTalk, The Journal of Defense Software Engineering March/April 2011 Vol. 24 No. 2 pp 4 -11. One or more of the authors is a U.S. Government employee working within the scope of their Government job; therefore, the U.S. Government is joint owner of the work and has the right to copy, distribute, and use the work.

14. ABSTRACT
A specific and currently relevant issue motivating the notion of ruggedized software is the confluence of the threat of cyber attacks and our increased dependence on software systems in enterprise as well as tactical situations. Software services that are essential for mission success must not only withstand normal wear and tear, stresses and accidental failures, they also much endure the stresses and failures caused by malicious activities and continue to remain usable. The Crumple Zone (CZ), a software shock absorber that absorbs attack effects before they cause significant system failures, is an architectural construct that we have developed and are maturing iteratively. We argue that the CZ is an important building block for constructing ruggedized software for supporting network-centric operations. In this paper we discuss the CZ in the context of Service-Oriented Architecture (SOA) and describe a configuration that has been realized and demonstrated.

15. SUBJECT TERMS
Crumple Zone, Service-Oriented Architecture (SOA), Cyber Attacks, SOA Survivability, Secure SOA

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			ASHER D. SINCLAIR
U	U	U	UU	9	19b. TELEPHONE NUMBER (*Include area code*) N/A

Crumple Zones

Absorbing Attack Effects Before They Become a Problem

Michael Atighetchi, Raytheon BBN Technologies
Partha Pal, Raytheon BBN Technologies
Aaron Adler, Raytheon BBN Technologies
Andrew Gronosky, Raytheon BBN Technologies
Fusun Yaman, Raytheon BBN Technologies
Jonathan Webb, Raytheon BBN Technologies
Joe Loyall, Raytheon BBN Technologies
Asher Sinclair, US Air Force Research Laboratory
Charles Payne, Adventium Labs

Abstract. A specific and currently relevant issue motivating the notion of ruggedized software is the confluence of the threat of cyber attacks and our increased dependence on software systems in enterprise as well as tactical situations. Software services that are essential for mission success must not only withstand normal wear and tear, stresses and accidental failures, they also must endure the stresses and failures caused by malicious activities and continue to remain usable. The Crumple Zone (CZ), a software shock absorber that absorbs attack effects before they cause significant system failures, is an architectural construct that we have developed and are maturing iteratively. We argue that the CZ is an important building block for constructing ruggedized software for supporting network-centric operations. In this paper we discuss the CZ in the context of Service-Oriented Architecture (SOA) and describe a configuration that has been realized and demonstrated.

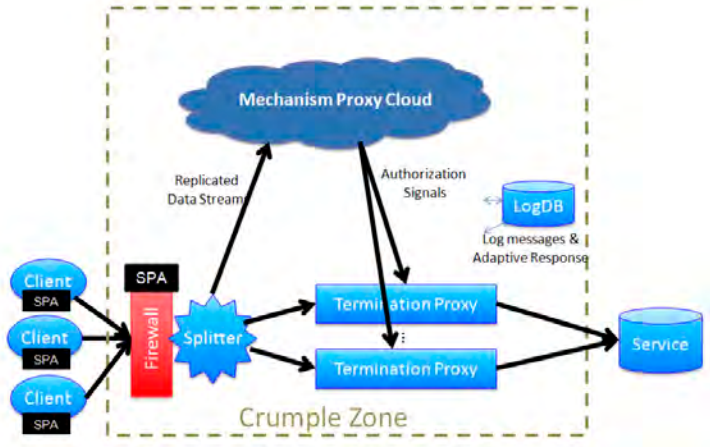
1. Introduction

A higher level of structural and operational endurance and ruggedness can be achieved in software systems by strategically introducing CZs in the system architecture. Analogous to the crumple zone in an automobile, a CZ stands before critical components and "absorbs" the effects of attacks by localizing or eliminating the damage they can cause and leaving the critical components intact and unaffected. The concept of software CZ is broadly applicable; in this paper we discuss CZs for SOA.

SOA is an architecture paradigm gaining popularity in military and civilian information systems, many of which play important roles in national security. Mission critical systems face a highly contested and hostile environment in real-world operations, and must endure and withstand malicious attacks. Potential threats against critical SOA-based systems range from automated network worms targeting SOA platform and supporting services to individual vandals to well-motivated and expert foreign intelligence apparatus that aim to subvert operations in the DoD enterprise and critical missions. The adversary objective may range from denying access to the system, to using the system without authorization, to tampering with or fabricating data in storage or in transit. But all indications, including our own assessment [1], point to serious lapses in the state of the art in SOA security. As a technology, SOA is still maturing and various aspects of SOA, including security features, are still being standardized. Furthermore, available SOA infrastructure and platforms do not always implement all of the available and specified standards. The complexity of SOA platforms combined with their rapid evolution can lead to implementers under-using or misusing available security features due to lack of expertise. Security of SOA systems is often limited to perimeter and network level [2] security.

Some of the very features that make SOA appealing, like loose coupling, dynamism, and composition-oriented system construction, make securing service-based systems more complicated. These features ease the development of systems, but also introduce additional vulnerabilities and points of entry than in self-contained, static, or stove-piped systems. In SOA, services are advertised and are looked up by potential users, many of which might not have the proper authorization to access or use the requested services. It is difficult to predict at design time exactly which actors will attempt to consume a given service and whether they will be authorized to do so. There are various system boundaries with a trust differential—one side is more trustworthy than the other side. Network and perimeter security only reinforce the "crunchy on the outside, chewy inside" view of software systems, and is utterly insufficient for developing rugged SOA systems.

Figure 1: Architectural Elements of the CZ



We argue that CZs can absorb attacks and minimize damage. CZs can be deployed at any trust boundary in the system. One key place we have experimented with and will describe in this paper is in the DMZ between the services enclave and the public network from which clients access the services.

The rest of the paper is organized as follows. Section 2 provides an overview of the CZ architecture. Sections 3-7 describe various key features of the CZ and the components and mechanisms responsible for them. Section 8 describes Related Work, Section 9 provides performance metrics and a cost/benefit analysis. Section 10 concludes the paper.

2. CZ Architecture

The CZ is, in basic terms, a layer of intelligent service proxies that work together to present a high barrier to entry to the adversary, to increase the chance of detection of malicious activities, and to contain and recover from failures and undesired conditions caused by malicious attacks. These proxies collectively implement the service's consumer-facing application programming interface. Different proxies help contain malicious activity by applying security checks and controls, then approving data for release if it passes those checks. A key principle of the CZ's design is that only data that has been inspected and approved by one or more proxies is passed along to the service. Because the CZ inspects and processes untrusted data, it is expected to fail occasionally. Automatic monitoring and re-start of the proxies inside the CZ is another key design feature.

Effectiveness of the CZ depends on three requirements:

- The CZ must be non-bypassable. All consumer requests to the service must be mediated through the CZ.
- The CZ must cover both known and unknown attacks. It should be configurable so defenses can be tailored to the system's operational requirements and the potential threat environment.
- The CZ must preserve the integrity of data that flows through it to prevent man-in-the-middle scenarios run by corrupted CZ components.

To meet the first requirement, making the CZ non-bypassable, conventional network level protections such as firewalls and routers can be used. To make it difficult for adversaries to discover and access protected services, CZ presents a very small exploitable surface to untrusted service consumers. This is accomplished by placing the CZ behind a firewall that uses single packet authorization (SPA). On the CZ's side of the firewall, termination proxies (TPs) are used as the entry point for all incoming client connections.

The second requirement, varied and configurable defenses, is achieved through a set of proxies that implement specific checks and are organized in a mechanism proxy cloud (MPC). The MPC monitors observable behavior of requests. We have implemented proxies that check assertions on application data, e.g., by checking serialization fields, as well as canary proxies that consume application data and thereby absorb attacks, e.g., by crashing or getting corrupted.

The third requirement, preserving data integrity within the CZ, is achieved by service layer virtual private groups (sIVPG). The Splitter component replicates SSL streams between clients and TPs to the MPC without breaking cryptographic envelopes. Key management components that are also part of the sIVPG selectively share keys from the TPs to the MPC so that the new streams can be decrypted for inspection.

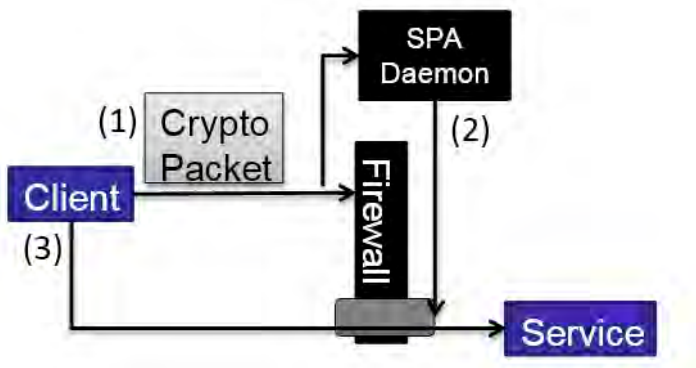
3. SPA

The first layer of defense an attacker coming from the outside needs to overcome is the CZ's firewall. In addition to standard restrictions on open ports and IP ranges, we use SPA [3] to implement a least-privilege policy that allows access to listening ports only to authenticated clients.

Figure 2 illustrates the general concept behind SPA using a client (on the left) trying to access the service (on the right) through the firewall (in the middle). The firewall starts out by blocking all traffic to the service. A legitimate client starts the interaction sequence (in step 1) by sending a cryptographic-based credential that is encoded within a single packet to the firewall. After verifying client identity and authorizing the client's connection request, the SPA server side component grants the client the right to establish a single TCP connection (for a limited amount of time) by adding specific firewall rules (step 2). Finally, the client establishes a normal TCP connection in step 3. A client without the proper credential is denied access.

SPA limits exposure of the protected enclave to port scans, remote OS fingerprinting, and low-level network stack exploits (such as TCP connection flooding). Port scan or OS fingerprinting attempts for reconnaissance will return no information unless the adversary has stolen or forged cryptographic credentials.

Figure 2: SPA



4. TP

TPs are advertised as service endpoints for the client, while the actual service is accessible only from the TP. The client believes it is connecting directly to the service, but the TP provides a barrier between the service and the client. The TP escrows client-server data until it is analyzed and determined to be safe to release.

One key design decision for constructing the TP was to keep its logic minimal and therefore making it less prone to exploits. For that reason, the TP does not itself analyze any client data because the analysis process might introduce corruption or crash faults. Instead, data analysis is performed in the MPC (see Section 5). If traffic passes all checks, the MPC sends authorization messages to the TP stating how many bytes of client data have been approved for release. The TP requires active approval of client data by the MPC within a certain amount of time. If the MPC detects anything wrong with the data or if the MPC fails to send a timely approval message, the connection to the client is closed by the TP and the escrowed data is discarded. Alternatively, when the MPC approves a certain number of bytes for release, the TP releases that amount of data from

escrow and sends it to the service. One key benefit of the split check-escrow model is that corrupted nodes in the MPC cannot directly affect the integrity of the application stream since MPC nodes only operate on a copy of the data and cannot alter the data that is released from the TP's escrow buffer. On the other hand, corrupted nodes in the MPC can incorrectly approve or disapprove release of escrowed data because the TP only receives instructions to release a certain number of bytes. This issue is dealt with by using voting on the release instruction, described in Section 5.

Crashes in the MPC will prevent approval messages from reaching the TP and will then result in the TP closing the connection to the client. All incoming client connections are routed through the TP–if the TP were to crash, many client connections would be terminated. Isolating the possible crashes in the MPC limits the number of clients affected by any crashes. Watchdogs help the system recover from crashes and are discussed in more detail in Section 7.

A single TP would be a single-point-of-failure in the CZ. This can be addressed by incorporating multiple TPs in the CZ, deployed in a manner analogous to load balancing. This provides isolation and replication to this critical part of the CZ. Additionally, in conjunction with the watchdog for the TP, the TPs can be moved and restarted to provide additional fault tolerance.

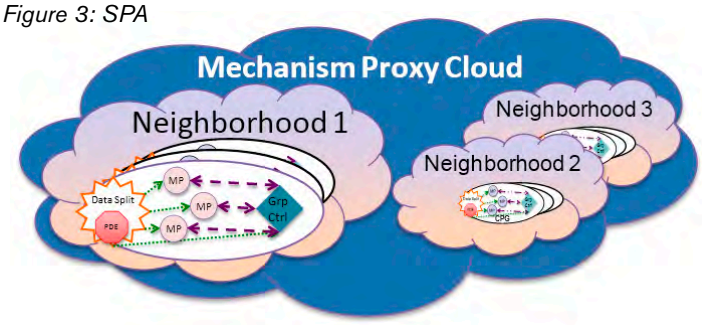
5. MPC

The MPC is a metaphor for a loosely coupled set of proxies that perform checks on application data. Figure 3 shows a detailed version of the MPC, which has a hierarchical structure. At the bottom of the hierarchy there are individual mechanism proxies (MPs) implementing check functionality, the next level up are the proxy groups (PGs), and finally the neighborhoods.

MPs inspect the content of the incoming traffic for attacks. For example, a rate proxy will raise a flag if the session has an unusually high message rate. Similarly a size proxy will reject a message with huge user data. Such proxies are useful for detecting known attacks, i.e., high message rate leading to denial of service, and big objects leading to heap overflow. To protect against novel attacks we utilize MPs that simulate (to a certain extent) the behavior of the protected service. If the simulated behavior is close enough to the actual behavior the effects of the novel attack can then be detected, absorbed, and managed by the proxy. The Canary proxy is an example based on this technique. Like the historical canary in a coalmine, a canary proxy will be affected by the attack in the same way the protected entity would. Canary is designed to parse the incoming stream the same way the server would thus protecting the deployed service against attacks that might be caused by arbitrarily malformed streams or arbitrary attack commands encoded in serialized data (for example, serialized instances of Java classes).

PGs represent a coordinated collection of MPs that together perform checks on application traffic. PGs are associated with SSL connections; each SSL connection between clients and TPs will be forwarded (through the sIVPG) to a dedicated PG. This assignment can be controlled at runtime based on available resources. The proxies within a group coordinate with a group controller (one controller per group), which regulates the control flow between the proxies in the group. Intuitively, the group controller enforces an order of execution on the proxies for improved protection. For example, to prevent unnecessary deaths of the canary proxy, we can chain a blacklist proxy, which screens for instances of known malicious classes, before the canary. The group controller is also responsible for communicating with the TP to notify it of the number of bytes cleared by all of the proxies in the group.

Figure 3: SPA



Neighborhoods represent fault isolation boundaries and are associated with processes in the current implementation model. For example, a corrupted MP running in Neighborhood 1 cannot directly access or spread to other MPs running in Neighborhood 2. A neighborhood can host multiple groups for load balancing purposes. Neighborhoods can be distributed within the MPC on different physical hosts and virtual machines.

In most cases, the crash of a canary like proxy also implies the crash of all components in the same neighborhood. This means that sessions of all clients sharing the same neighborhood will terminate. However, clients connecting through other neighborhoods will not be affected and future connections will go through the remaining neighborhoods.

To address the issue of a malicious MP incorrectly instructing the TP about escrow release mentioned earlier, one needs to assign redundant PGs to a single SSL connection and vote on the group's release decision. If the PGs are sufficiently independent, known fault tolerance schemes can be employed to detect and tolerate the desired number of corrupt PGs.

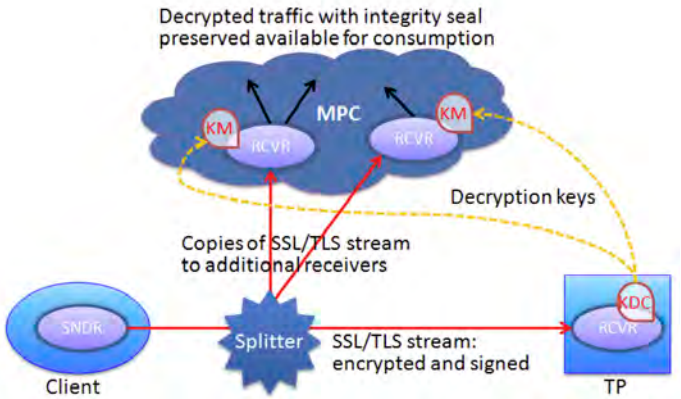
6. sIVPGs

At a high level, the function performed by the sIVPG is to a) replicate the encrypted stream without losing any application data, b) share keys so that the receiving end points (RCVRs) in the MPCs can decrypt and verify the integrity of the replicated SSL packets, and c) make the decrypted stream available to the MPs. We explored various implementation options including libpcap-based packet sniffers [4] to replicate the traffic stream, and settled on a netfilter-based approach [5] because the latter provides more robustness against packet loss.

In this approach, as soon as a client connection is initiated, the splitter component, as shown in Figure 4, starts to buffer traffic from that connection using a netfilter module. When the SSL handshake is completed and the PG in a MPC neighbor-

hood has been initialized to handle the new connection, the Key Distribution Center at the TP and Key Managers in the neighborhoods communicate to exchange the SSL keys. In parallel, the splitter starts to forward the buffered data to the RCVRs. The RCVRs buffer data until the key exchange step is completed, and make the decrypted data available through stream interface as soon as the necessary keys are available. Note that if the client-server messages are signed and encrypted at the application level, an additional level of key sharing is needed to make the decrypted data available for inspection and processing to the proxies.

Figure 4: sIVPGs



7. Recovery Focused Adaptation

The CZ is equipped with adaptation mechanisms that enable recovery and containment of attack effects. TPs and each MPC neighborhood have a watchdog that monitors the respective components and automatically restarts them when a crash is detected. A restarted component reconnects itself to its peers and begins handling new client connections. The watchdogs poll the components in a configurable interval (one second in our test configuration). Component restart time is dependent on configuration and load details. In our test configuration, components start in less than one second.

The CZ also maintains a database of log messages with database permissions set so that CZ components can write to the database (but not read) and only designated analysis components can read from the database (but not write). The logging mechanism collects data that will help the system prevent or minimize future attacks. For example, each time a check does something that might cause the neighborhood to crash (such as checking a serialized object through the canary proxy), it enters a log message. When it finishes executing the code that may cause a crash, it enters another log message. These log messages contain timestamps as well as the IP information about the connection under analysis.

The log analysis component analyzes the data collected in the log database. In particular it looks for indications that a particular client connection caused a crash. For example, a neighborhood that crashed might have a log message indicating that a block of potentially-crash-producing code was entered, but was never exited. The log analyzer can take proactive actions—either by modifying the firewall to prevent connections from a particular IP address or by assigning connections from an IP address to a high-risk neighborhood to further protect other client connections from potential crashes.

The watchdogs and logging insure that the CZ remains available, is resilient to attacks, and proactive in preventing or minimizing the effects of future attacks.

8. Related Work

Port Knocking [6] is similar to SPA, but SPA has the following advantages over Port Knocking: SPA is based on strong cryptographic ciphers, making spoofing more difficult, SPA packets are non-replayable, and SPA is robust against trivial sequence busting attacks.

SPA Implementations take different approaches; we explored two open-source implementations, Fwknop [7] and knockknock [8]. These implementations differ in ways that might impact which one is chosen for a specific deployment. For packet encoding, Fwknop uses dedicated UDP packets while Knockknock encodes requests in TCP headers. This implies that Knockknock requires admin privileges on the client to generate customer TCP headers. For packet capturing, Fwknop uses libpcap (a large C library) to passively sniff SPA packets. Knockknock reads packet information from kern.log through a daemon that restricts root privileges to only ~15 lines of Python code. In our view, this makes the Knockknock daemon less likely to be subverted. Regarding functionality, Fwknop provides feature rich support for service ghosting and port randomization, while Knockknock follows a minimalistic approach.

Web Application Firewalls (WAFs) are designed to protect J2EE applications and web services (WS) against common vulnerabilities listed in the OWASP top 10 list, e.g., SQL injection. While most WAFs are deployed at DMZ boundaries only and are hosted on hardened appliances, CZs are based on a lightweight distributed software paradigm that allows us to surround a selected set of services with fine-grained defenses. WAFs support only WS-related interaction models and lack support, for example, for other distributed protocols such as Java RMI.

Application Server Clustering ensures availability of services by transparently rerouting traffic to redundant application servers in the presence of attacks that affect service availability. Load-balancers and clusters can work in conjunction with CZ to implement voting.

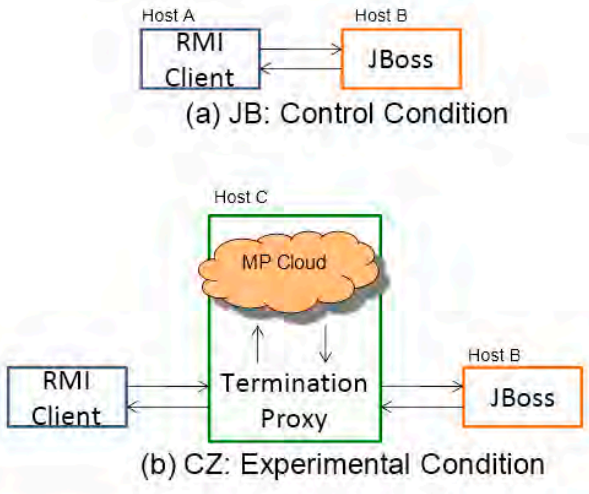
XML Appliances provide security through schema validation, WS-security functions, and assured transformation of content using standards like XSLT. While there is some similarity between CZ MPs and functionality provided by XML appliances, XML appliances are based on a single hardened platform and don't provide advanced features such as canary proxies, diverse proxy implementation, and automatic restart.

Cross Domain Guards mitigate information exchange risks between different classified networks. New generation SOA-based guards [9][10] have started separating filter functionality into services that can be hosted outside of appliances, similar to the MPC. Compared to the work described here, existing certifi-

cation and accreditation requirements play a more important role in guards, preventing the use of advanced techniques that don't fit current practices, e.g., use of canary proxies and probabilistic design algorithms.

9. Experimental Validation

Figure 5: Experiment Configurations



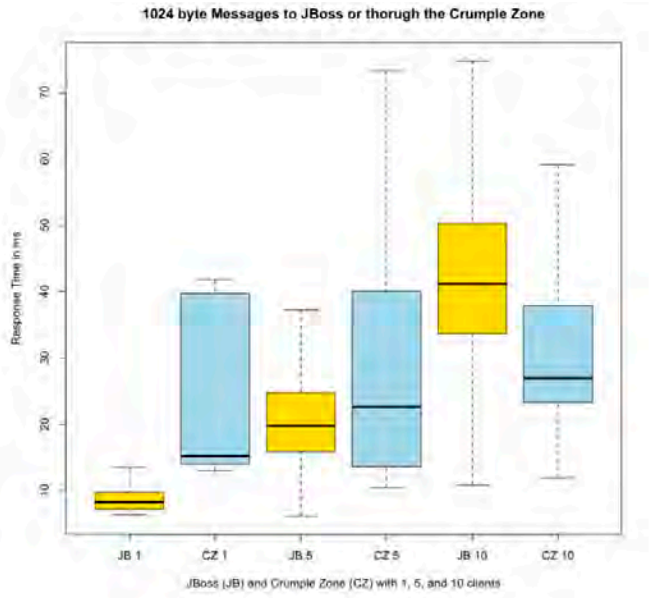
To evaluate the performance and robustness of the current proof-of-concept prototype CZ, we conducted multiple internal experiments. The system under test consisted of a Java RMI service and a MPC with four MPs, including rate, size, white list, and canary checks. Figure 5 shows the base and control conditions used.

We experimented with two categories of client messages: computation intensive and data intensive. The compute intensive messages are short but require the JBoss server to perform a mathematical calculation. The data messages were 1KiB, 10KiB, and 100KiB in size and required the JBoss server to process the data. As one might expect, the overhead of the CZ increases as the messages increase in size. This overhead ranges from 18% for the computation message to 84%, 803%, and 4040% for 1KiB, 10KiB, and 100KiB data messages respectively.

Our future work includes investigating and optimizing our code to handle large messages more efficiently. We suspect that the extra I/O load analyzing the data could be responsible for the slower processing.

Additionally we investigated server response time when multiple clients make requests simultaneously. The results for testing one, five, and 10 clients are shown in the box plot in Figure 6. Interestingly, the response times for the CZ improve relative to the control condition as more clients are added. In fact, the median response time for the CZ is less than the median response time for the control condition when 10 clients connect at once. We believe that this improvement is due to the CZ sharing a connection to the JBoss server for all of the clients versus a separate connection to the JBoss server for each client in the control condition. This experiment shows that although there is overhead for a single client using the CZ this may be mitigated when multiple clients connect through the CZ.

Figure 6: Experiment results for multiple client connections



As shown in Figure 5, all of the CZ functions, including the termination proxy and the MPC, were hosted on a single host. While this configuration introduces minimal cost overhead in terms of additional hardware costs, IO operations on the single machine will become a choke point given enough load. We plan to investigate other deployments in the future in which MPs are distributed across a set of machines in a load-balanced way. The expectation is that load-balanced configuration will decrease round trip times under heavy loads although increasing hosting costs.

10. Discussion and Next Steps

The CZ design and prototype described in this paper provides a promising foundation for protecting critical services from malicious attacks that succeed to a degree, i.e., get past the traditional access control and authentication services. This means that the CZ should be effective against novel, zero-day, and insider attacks.

The degree to which the CZ is effective against a particular attack depends greatly upon the extent to which the MPC replicates the server functionality and the kind of cross checking algorithms employed. In the extreme case, the Canary Proxy would replicate most of the server functionality, and would be susceptible to, and therefore provide protection against any attack that would be effective against the service, and the proxy group-TP processing would be made Byzantine Fault- Tolerant. The amount of redundancy and protocol overhead must be weighed against the perceived threat model. One of the next steps that we are going to undertake is to evaluate the benefits and costs of protections and simulated functionality in the MPC, and how it fits particular threat models and platform performance requirements.

Similarly, the current prototype only protects the critical server from attacks by rogue clients. However, a fully protected system will want to protect the return path also, i.e., protect a client from

a server that might have been compromised. To accomplish this, the return path from the client and server must go through a CZ. This CZ should be similar to the one on the request path, except that the functionality simulated by the Canary Proxy will involve processing of the response.

The current prototype concentrates on protecting server calls made over RMI. Although this is a valid model, representing calls made by composed clients and servers, a large class of client-server interactions in SOA are through WS interchanges, e.g., using SOAP. We are currently in the process of designing a CZ that works with WS interfaces.

Finally, to substantiate our claims that the CZ can protect against large classes of known, zero-day, novel, and insider attacks, we plan to conduct experiments and collect concrete and empirical evidence. As we have done in prior research projects [11], we plan to conduct independent red team exercises to evaluate the efficacy of the CZ to protect against attacks by motivated and determined adversaries. In these exercises, an independent red team, experienced in cyber attacks and with insider knowledge of the system being protected, but not part of the development team, will launch attacks against the system. We will evaluate the ability of the CZ to absorb the attacks and protect the service, and the extent of the class of attacks that the CZ is effective against. To the extent possible, we will measure the difference in time to effectively compromise the system with and without CZ.

Acknowledgments

The authors would like to acknowledge the support and collaboration of the US Air Force Research Laboratory (AFRL) Information Directorate. This material is based upon work supported by the Air Force Research Laboratory under Contract No. FA8750-09-C-0216.

ABOUT THE AUTHOR

Michael Atighetchi is a senior scientist at BBN's Information and Knowledge Technologies business unit. His research interests include cross-domain information sharing, security and survivability architectures, and middleware technologies. Mr. Atighetchi has published more than 35 technical papers in peer-reviewed journals and conferences, and is a senior member of the IEEE. He holds a master's degree in computer science from University of Massachusetts at Amherst, and a master's degree in IT from the University of Stuttgart, Germany.

Raytheon BBN Technologies
10 Moulton Street
Cambridge, MA 02138
Phone: (617) 873-1679
Fax: (617) 873-4328
E-mail: matighet@bbn.com

Dr. Partha Pal is a lead scientist at Raytheon BBN Technologies. He leads the survivability research thrust of at Raytheon BBN, and has served as the principal investigator in a number of DARPA, DHS and AFRL R&D projects in the areas survivability and information assurance. He has published over 65 papers in refereed journals, conferences and workshops, has been in the program committees of multiple workshops and conferences, and has been a co-organizer of the Recent Advances in Intrusion Tolerance workshop for the past two years.

Raytheon BBN Technologies
10 Moulton Street
Cambridge, MA 02138
Phone: (617) 873-2056
Fax: (617) 873-4328
E-mail: ppal@bbn.com

ABOUT THE AUTHORS cont.

Dr. Aaron Adler is a Scientist in BBN's Information and Knowledge Technologies business unit. His research interests include distributed systems, artificial intelligence, and human computer interaction, specifically sketch recognition. He has a Ph.D. in Computer Science from Massachusetts Institute of Technology (2009).

Raytheon BBN Technologies
10 Moulton Street
Cambridge, MA 02138
Phone: (617) 873-3517
Fax: (617) 873-2794
E-mail: aadler@bbn.com

Andrew Gronosky is a staff engineer at Raytheon BBN Technologies. He has experience developing a variety of software applications including data analysis and visualization, digital signal processing, and parallel and distributed systems. He holds a Master of Science degree in mathematics from Rensselaer Polytechnic Institute and is a member of the IEEE and the ACM.

Raytheon BBN Technologies
10 Moulton Street
Cambridge, MA 02138
Phone: (617) 873-3517
Fax: (617) 873-3486
E-mail: agronosk@bbn.com

Dr. Fusun Yaman is a Scientist in BBN Technologies. Her research interests are in distributed planning, spatio-temporal reasoning and machine learning specifically learning user preferences from observations. She has a Ph.D. in Computer Science from University of Maryland at College Park (2006).

Raytheon BBN Technologies
10 Moulton Street
Cambridge, MA 02138
Phone: (617) 873-3966
Fax: (617) 873-2794
E-mail: fyaman@bbn.com

Jonathan Webb is an engineer in BBN's Information and Knowledge Technologies business unit. Over 20 years at BBN, Mr. Webb has been involved in a wide range of software development projects including simulation of dynamic systems, web based data management systems, middleware for information management, and cross domain information sharing.
Mr. Webb has a master's degree in aeronautics and astronautics from the Massachusetts Institute of Technology.

Raytheon BBN Technologies
10 Moulton Street
Cambridge, MA 02138
Phone: (617) 873-3321
Fax: (617) 873-4328
E-mail: jwebb@bbn.com

Dr. Joseph Loyall is a principal scientist at Raytheon BBN Technologies. He has been the principal investigator for Defense Advanced Research Projects Agency and AFRL research and development projects in the areas of information management, distributed middleware, adaptive applications, and quality of service. He is the author of over 75 published papers; was the program committee co-chair for the Distributed Objects and Applications conference (2002, 2005); and has been invited speaker at several conferences and workshops. Dr. Loyall has a doctorate in computer science from the University of Illinois.

Raytheon BBN Technologies
10 Moulton Street
Cambridge, MA 02138
Phone: (617) 873-4679
Fax: (617) 873-4328
E-mail: jloyall@bbn.com

Asher Sinclair is a Program Manager at AFRL's Information Directorate working in the Enterprise Information Management Branch at the Rome Research Site.
His work history includes enterprise systems management, service-oriented architectures, information-level quality of service, and network security.
He has contributed to more than 12 technical papers and conference proceeding publications. He holds a bachelor's degree in Computer Information Systems from the State University of New York and a master's degree in Information Management from Syracuse University.

AFRL
525 Brooks Road
Rome, NY 13441
Phone: (315) 330-1575
E-mail: asher.sinclair@rl.af.mil

ABOUT THE AUTHORS cont.

Charles Payne is a Member of the Technical Staff at Adventium Labs in Minneapolis, Minnesota. He has been a Principal Investigator for the Office of Naval Research in the area of virtualized cross domain support and has been a key contributor to DARPA, DHS and AFRL programs investigating high assurance security architectures. He has published more than a dozen papers and served as Program Chair for the Annual Computer Security Applications Conference (2009). Mr. Payne has a Masters of Science degree in Computer Science from The College of William and Mary in Virginia.

Adventium Labs
111 Third Avenue South, Suite 100
Minneapolis, MN 55401
Phone: (612) 817 2525
E-mail: charles.payne@adventiumlabs.org

REFERENCES

1. Michael Atighetchi, Partha Pal Andrew Gronosky, "Understanding the Vulnerabilities of a SOA Platform - A Case Study," in The 9th IEEE International Symposium on Network Computing and Applications (IEEE NCA10) , Cambridge, MA USA, 2010.
2. Eric Rescorla, SSL and TLS: Designing and Building Secure Systems. United States: Addison-Wesley Pub Co., 2001.
3. Michael Rash. (2006) Single Packet Authorization with Fwknop. [Online]. <http://www.cipherdyne.org/fwknop/docs/SPA.html>
4. (2010, September) TCPDUMP home page. [Online]. <http://www.tcpdump.org/>
5. (2010, September) Netfilter Homepage. [Online]. <http://www.netfilter.org/>
6. Martin Krzywinski. (2005) portknocking.org. [Online]. <http://www.portknocking.org/docs/portknocking_an_introduction.pdf>
7. CipherDyne. (2010, September) CipherDyne. [Online]. <http://www.cipherdyne.org/fwknop/>
8. Moxie Marlinspike. (2010, September) KnockKnock. [Online]. <http://www.thoughtcrime.org/software/knockknock/>
9. Jason Ostermann. (2009) Presentation at UCDOM Annual Conference: Raytheon DSCDS Intro. [Online]. <http://www.ucdmo.gov/conference09/Ostermann_Raytheon%20DSCDS_09022009.pdf>
10. BAH. (2009) Presentation at the UCDMO Annual Conference: BAH DSCDS Overview. [Online]. <http://www.ucdmo.gov/conference09/Morris_BAH%20DSCDSoverview_final_09022009.pdf>
11. Joe Loyall Michael Atighetchi, "Meaningful and Flexible Survivability Assessments: Approach and Practice," in CrossTalk - The Journal Of Defense Software Engineering, March/April 2010, pp. 12-18.