Cooperative Red Teaming of a Prototype Survivable Service-Oriented System

Partha Pal, Michael Atighetchi, Andrew Gronosky, Joseph Loyall
Raytheon BBN Technologies
Cambridge, MA
{ppal, matighet, agronosk, jloyall}@bbn.com

Charles Payne
Adventium Labs
Minneapolis, MN
charles.payne@adventiumlabs.com

Asher Sinclair, Brandon Froberg, Robert Grant
Air Force Research Laboratory
Rome, NY
{Asher.Sinclair, Brandon.Froberg, Robert.Grant}@rl.af.mil

***Abstract*—An increasing number of military systems are being developed using service orientation. Some of the features that make service orientation appealing, like loose coupling, dynamism and composition-oriented system construction, make securing service-based systems more complicated. We have been developing technologies for *Advanced Protected Services (APS)* to improve the resilience and survival of services under cyber attack. These technologies introduce a layer to absorb, contain, and adapt to cyber attacks before attacks reach critical services. This paper describes an evaluation of these advanced protection technologies using *cooperative red teaming.* In cooperative red teaming, an independent red team launches attacks on a protected enclave in order to evaluate the efficacy and efficiency of the protection technologies, but the red team is provided full knowledge of the system under test and its protections, and is given escalating levels of access to the system. The red team also operates within agreed upon rules of engagement designed to focus their effort on useful evaluation results. Apart from presenting the evaluation results, we also discuss cooperative red teaming as an effective means of evaluating cyber security.**

Keywords-component; Service-Oriented Architecture, Survivability, Adaptive Security, Red Team Evaluation

I. INTRODUCTION

Service-orientation is a software engineering methodology gaining popularity in military and civilian information systems, many of which play important roles in national security. Some of the very features that make service-orientation appealing, like loose coupling, dynamism and composition-oriented system construction, make securing service-based systems more complicated. These features ease system development, but introduce additional vulnerabilities and points of entry beyond those that exist in self contained, static, or stove-piped systems.

We have been developing the following technologies for *Advanced Protected Services* (APS) to improve the resilience and survival of services under cyber attack:

- A *crumple zone*, analogous to the crumple zone in an automobile, which is a protective layer that absorbs the effects of attacks by localizing or eliminating the damage they can cause and leaving the critical components intact and unaffected.
- *Containment regions* which ensure that attackers have to penetrate layers of defenses before they can do damage, and that multiple flows of access to critical services are isolated, so that successful attacks that deny service to some users do not affect others.
- *Adaptive response*, so that attacks whose effects might be absorbed by the crumple zone are logged, analyzed, and blocked on subsequent attempts. This helps protect against novel, zero day, and repeated attacks.

We have described the APS crumple zone and other concepts and prototype in [1]. In this paper, we describe the experimental evaluation we performed with the US Air Force Research Laboratory (AFRL) to assess the efficacy of the APS protections. The evaluation consisted of a set of *red team exercises*, in which an independent set of penetration testers attacked an APS-protected service enclave in a test bed at AFRL. Provided full access to the APS source code, documentation, and developers, and provided with ever-increasing levels of access to the system (including insider access), the red team launched a variety of attacks over the span of several days. The APS system held up against these attacks well; succumbing only to a few attacks that had significant insider access and privilege. The results provide insight not only about the effectiveness of the APS survivability approach but also where additional research and development is required.

This paper provides the following contributions:

- It provides important information about the effectiveness of the APS approach to developing survivable systems.
- It motivates and documents the use of cooperative red teaming as an important part of security evaluations. This is especially important because system security is notoriously difficult to measure quantitatively and objectively.
- It provides a development and evaluation roadmap for others aspiring to develop resilient and survivable systems, and further research directions.

The remainder of this paper is structured as follows. Section II provides a brief introduction to the APS survivability prototype system. In Section III, we describe the cooperative red teaming approach. In Section IV, we describe the specifics of the APS red team exercise, including the system under test, the rules of engagement, and the levels of access granted to the

This work was supported by the US Air Force Research Laboratory (AFRL) under contract FA8750-09-C-0216. This paper is approved for public release dated 27 Apr 2012, case number 88ABW-2012-2507.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
NOV 2012		**00-00-2012 to 00-00-2012**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Cooperative Red Teaming of a Prototype Survivable Service-Oriented System	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Raytheon BBN Technologies,10 Moulton Street,Cambridge,MA,02138**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Military Communications Conference (MILCOM), Orlando, FL, October 29-November 1, 2012.

14. ABSTRACT
An increasing number of military systems are being developed using service orientation. Some of the features that make service orientation appealing, like loose coupling, dyna-mism and composition-oriented system construction, make secur-ing service-based systems more complicated. We have been de-veloping technologies for Advanced Protected Services (APS) to improve the resilience and survival of services under cyber at-tack. These technologies introduce a layer to absorb, contain, and adapt to cyber attacks before attacks reach critical services. This paper describes an evaluation of these advanced protection tech-nologies using cooperative red teaming. In cooperative red team-ing, an independent red team launches attacks on a protected enclave in order to evaluate the efficacy and efficiency of the pro-tection technologies, but the red team is provided full knowledge of the system under test and its protections, and is given escalat-ing levels of access to the system. The red team also operates within agreed upon rules of engagement designed to focus their effort on useful evaluation results. Apart from presenting the evaluation results, we also discuss cooperative red teaming as an effective means of evaluating cyber security.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**6**	

red team. In Section V, we summarize the results of the red team exercises. In Section VI, we compare to related research. Finally, in Section VII, we present some concluding remarks.

II. THE APS SURVIVABILITY ARCHITECTURE

The APS protection techniques address the vulnerabilities introduced by service-orientation into critical systems. While the full threat model that we target is beyond the scope that can be covered in the space available in this paper, we focus on the threats enabled by the open discovery, multiple entry points, and dynamic binding. In service-oriented environments, 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. Network and perimeter security only reinforces the "crunchy on the outside, chewy inside" view of software systems, and are utterly insufficient for developing survivable service-oriented systems.

We introduce a new architectural feature, a *crumple zone (CZ)* that, analogous to the crumple zone in an automobile, is positioned in front of critical services and *absorbs* the effects of attacks by localizing or eliminating the damage they can cause and leaving the critical service intact and unaffected.

The CZ, shown in Figure 1 and described in more detail in [1] is a layer of intelligent service proxies that work together to present a high barrier of 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 interface. Various proxies contain malicious activity by applying security checks and controls, then approving data for release if it passes those checks. 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. Therefore, we include monitoring and restart of CZ components.

To be effective, all client interactions with the protected service must be intercepted and routed through the CZ. To make the CZ non-bypassable, we use several techniques. First, we use firewalls and routers to route and control traffic through the CZ. Second, to make it difficult for adversaries to discover and access protected services, CZ presents a very small exploitable surface to untrusted service consumers by placing the crumple zone behind a firewall that uses *single packet authorization* (SPA) [2]. Service discovery requests that make it through the firewall are delivered an endpoint reference to a *termination proxy (TP)* that is used as the entry point for all incoming client connections.

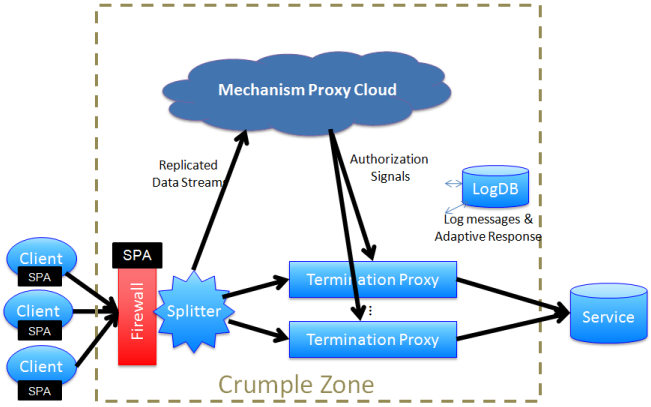

Figure 1. Architectural Elements of the Crumple Zone

All incoming requests are escrowed in a TP that terminates the SSL connection. A copy of the request is forwarded to a set of proxies that implement specific checks and are organized in a *mechanism proxy cloud* (MPC). 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 Splitter component replicates SSL streams between clients and TPs to the MPC without breaking cryptographic envelopes. Key management components of the CZ selectively share keys from the TPs to the MPC so that the new streams can be decrypted for inspection.

The CZ provides both *horizontal* and *vertical containment* of attacks. Horizontal containment is provided by the layering of multiple and varied defenses (i.e., providing *defense in depth*). If an attack gets through one defense, e.g., the SPA firewall, there is another defense there to block it, e.g., SELinux policies and mechanism proxy checks. Vertical containment is provided by multiple MPCs, each running in their own VM, process, or dedicated host. Client requests are spread over these MPCs to balance and contain the spread of attacks. If an attack crashes the MPC VM or process, its effects will be localized to that MPC and will not affect client interactions routed through other MPCs.

Lastly, the CZ includes adaptive defense for attack containment. CZ components store audit events in a *Log Database*. A *Log Analysis* component watches for violation events, which can occur when a mechanism proxy flags a suspected attack, when the CZ takes an attack remediation action, or when an adverse effect of an attack happens (and is absorbed by the CZ). For violations that are difficult to spoof, the Log Analysis implements a simple "three strikes and you are out" policy by eventually adding source IPs of connections to a block list.

III. THE COOPERATIVE RED TEAM EVALUATION PROCESS

Quantitative evaluation of security and survivability is difficult. There is no well-defined and generally accepted methodology. We have successfully employed red team experimentation in previous survivability projects [3][4][5][6] .

A Red Team exercise is a controlled experiment to assess security and survivability of a defended system through rigorous testing, performed by a Red Team consisting of computer specialists that play the role of cyber adversaries, a Blue Team, consisting of system operators and cyber defenders, and a White Team that plays the role of referee.

Red Team exercises can be structured to follow cooperative or adversarial paradigms. In *cooperative* arrangements, both the Red and Blue Teams work together to focus attacks on the parts of the system under test that have the highest chance of yielding useful insights. Also, both teams work together to identify the right mix of depth vs. breadth of attacks to maximize the knowledge that can be learned from the exercise, which in turn feeds into the next technology improvement cycle. An advantage of cooperative exercises is that the developers of a

system can participate as sophisticated inside attackers, as they already know everything there is to know about the system, complementing and enhancing the Red Team's knowledge and expertise with the latest attacks and techniques. In *adversarial* exercises, teams do not cooperate during experiment execution. Instead, each team follows their objectives to either defend the system under test or attack it. While adversarial exercises are considered to provide more realistic and independent assessment, previous exercises [7][8][9][10][11] have taught us that a cooperative methodology is more valuable and cost-effective for evaluating R&D prototypes.

In APS, we conducted a cooperative red team exercise. The Red Team consisted of members of an independent penetration testing team, other AFRL personnel, and "traitors" from the development team. The Blue Team consisted of developers from BBN and Adventium. Finally, the Project Managers from AFRL and BBN formed the White Team. Metrics and analysis from the red team exercise complement the internal laboratory-based experimentation conducted under the APS project.

The exercise was conducted over the course of a week at AFRL, after several months of cooperative planning, attack generation, and white board sessions. During the preparation, BBN provided copies of the software with documentation, assisted AFRL in setting up the system under test, and provided in-depth details about the software's architecture and design.

IV. THE APS RED TEAM PROCESS

A. System Under Test(SuT) and Claims

For a successful assessment, it is important to succinctly describe the system to be evaluated in terms of the technologies used, component layout, and network topology and the claims the technology is making. Figure 2 shows the components that were evaluated as part of the exercise together with Layer 4 connections. Figure 3 shows the network topology used during the experimentation. To assess the effectiveness and overhead of the technology being evaluated the SuT needs to protect a vulnerable application, which serves as the baseline undefended system. We picked a baseline undefended application based on the following criteria. First, that it should exhibit vulnerabilities that are commonly found in DoD systems, e.g., threats listed in DoD specific threat assessment reports. Second, that it is susceptible to vulnerabilities of Information Management Systems (IMSs) evaluated in previous exercises [8][9][10]. We compared project metrics of the defended system (i.e., the SuT configured to protect the vulnerable application) and the baseline (i.e., a configuration containing only the vulnerable application), to validate the APS survivability claims.

Claims enable us to focus the evaluation on specific properties of specific parts of the system and avoid testing the space of known limitation. We defined the following six claims:

1. The CZ improves application survivability, including confidentiality, integrity, and availability (CIA)
2. The CZ is non-bypassable, meaning all traffic between clientIP and jbossIP is intercepted by either czIP1 or czIP2
3. The CZ tolerates crashes of a single instance of a number of components, including TPs, Splitter, RmiReg, and MPC
4. The CZ tolerates corruption of the Splitter component aimed at impacting message integrity and confidentiality

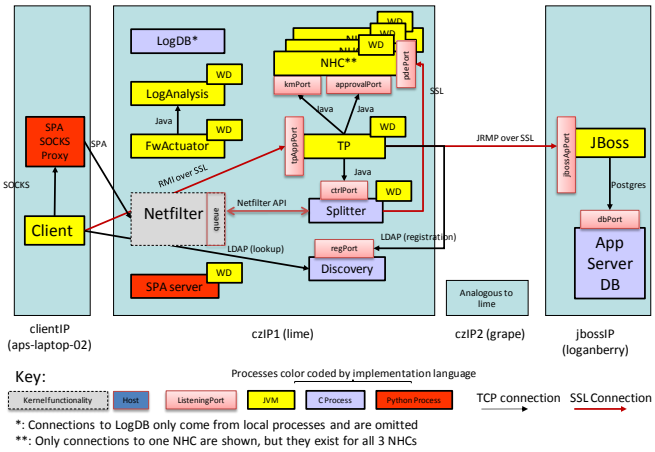

Figure 2. Component View

5. The CZ provides controlled inspection of application-level messages while maintaining confidentiality and integrity of data (in the absence of corruption of CZ components)
6. The CZ minimizes information disclosure at the network layer by hiding listening sockets to unauthorized clients.

B. Rules of Engagement (RoE)

Rules of engagement are used to guide the behavior of both the Red and Blue Teams. We used RoE to avoid testing what is already known about the SuT and to focus on gaining new insights instead. For cooperative exercise, it is less important to precisely capture the RoE than it is to document known limitations, starting points, and objectives of the exercise.

Known Limitations: The following list contains limitations of the SuT that were known going into the exercise and hence did not need to be assessed:

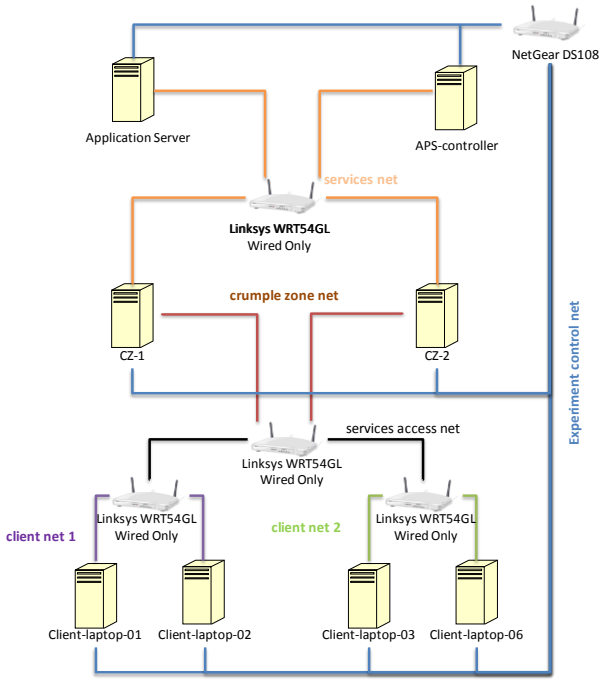

Figure 3. Network Topology View

1. No redundant network paths. Since we were using Single Small Office Home Office (SOHO) routers, we ruled out attacks on network bandwidth availability.
2. No specific router settings. Since the routers were not hardened, e.g., by configuring static ARP, we ruled out low-level network attacks, e.g., ARP spoofing.
3. No specific OS hardening or redundant host configuration. We ruled out attacks targeted at corrupting or crashing the CZ TCP/IP stack.
4. No defenses for the client machine. We ruled out attacks against the client. In fact, a corrupt client was an attacker starting point.

These limitations were due to the fact that we were evaluating a R&D prototype system rather than a transition-ready, matured and hardened system. None of the limitations listed here require new R&D efforts, and can be addressed by simply taking existing technologies and spending the time, effort, and budget to harden and configure them appropriately.

Adversary Starting Points: To perform a thorough assessment of a survivable system, it is important to avoid situations where the Red Team spends a large amount of time trying to overcome and evaluate only a specific defense as opposed to an overall evaluation of the system, especially for a system that is designed with defense-in-depth in mind. For instance, in the OASIS Dem/Val Red Team exercise [4], the Red Teams spent days working against network-level protections. Since the attacker laptop hooked up to the WAN (analogous to the starting point 1 below) was their only starting point, this necessary first step took time away from other useful testing.

For the purpose of the APS exercise, the Red Team launched attacks from the following starting points with varying privileges, enabling us to assess the multiple levels of defense provided by APS:

1. Attacker Machine: The adversary's machine (i.e., the adversary has root access) is connected to one of the client enclaves.
2. Corrupted Client: The adversary has root access on a client machine.
3. Corrupted TP: The adversary can execute arbitrary code as part of the TP process, realized by allowing the adversary to add custom classes to the TP's classpath. Note that this does not equate to root privilege on the TP machine.
4. Corrupted MPC process: The adversary can execute arbitrary code as part of a constituent MPC process.

Threat Model: The threat model was based on the notion of sophisticated adversaries. We assume that the adversary has complete knowledge about the system, including access to network topology diagrams, architecture and design documents, and all source code. However, the adversary does not have access to locally installed crypto material on end-systems, e.g., private keys used to establish TLS connections or pass SPA authorization. The overall objective of the Red Team is to assess the effectiveness of the CZ by trying to cause loss of availability, integrity, or confidentiality of the protected service. The objective of the Blue Team is to keep up service availability, integrity, and confidentiality in the presence of sustained attacks by participating in the operations of the CZ via administrative interfaces.

V. Results of the Red Team Exercises

Over the course of one week at the Air Force Research Laboratory in Rome, NY, the red team launched over 40 attacks on the APS protected service. While the attacks that were launched represent a specific attack scenario, the attacks provided coverage over commonly used threat models for web applications, e.g., the OWASP Top 10 list. In many cases, when an attack did not succeed, we provided more access (e.g., turning off a firewall or launching the attack from inside the CZ host and tried the attack again). In addition, we conducted white board sessions for a dozen additional attacks, i.e., we discussed what the effects would be and analyzed the effectiveness of the attack, the defense against it, and decided whether the attack should be launched.

The test environment included automated client-bots making continuous calls on the protected service through the CZ. During the attack, blue team members looked over system logs in real time, but did not take any defensive action.

The APS protected system exhibited significant resilience and adaptability at multiple layers. No attack without privilege or only client privilege succeeded. Two attacks succeeded in causing damage as planned without additional privilege, one that corrupted the splitter and one that launched a *fork bomb*, i.e., an attack that forks new processes recursively until all CPU resources are exhausted. Both attacks started inside and with considerable privilege. Three other attacks eventually succeeded when granted deeper access and additional privilege.

Table 1 shows a summary of the attacks executed during the weeklong red team exercise. It shows the attack number, variants if any, a description of the attack, which of CIA it covers, which APS claims it covers (described in Section IV), and whether the attack runs were successfully survived or not. The attacks were determined by members of both red and blue team based on the threat model and prior team-member experiences aiming to achieve coverage over the APS claims.

As Table 1 shows, we ran multiple runs for some attacks, in most cases granting increasing access and privilege. In others, the additional runs were testing different starting points. A green check mark means the attack did not succeed, i.e., the APS software absorbed the effects of the attack. The yellow triangle (a yield sign) means that the attack worked, but only when additional concessions (such as greater access or shutting off some defenses) were provided to the attackers. A red X means that the attack worked as planned.

Despite the successful defense against the attacks that had considerable access and privilege, we note that highly effective attacks that we did not cover or consider may very well exist. But overall, we have taken a significant step in the right direction improving the defense of service-oriented systems.

Time and resources did not permit running all attacks that were conceived and developed to varying degrees of maturity. Some of them were run outside the context of the red team exercise week at the blue team laboratory. Others were analyzed during white board sessions at various planning meetings leading up to and during the exercise week. Table 2 shows the summary of runs that were performed outside of the exercise week and attacks that were analyzed during white board sessions. The legends are same as Table 1, except the "NA", indicating that the attack was not applicable to the service used in the system under test (e.g., the attack exploited a vulnerability

Table 1. Summary Results of Red Team Experiments Ran during the Exercise Week.

No.	Var	Description	CIA	Claim	Survival Against Runs						
4		SQLI- quote-truncation	C	5	▼						
10		Watchdog – kill watched process (a-c)	A	3	✓	✓	✖				
14		NW Recon – sniff identify connection and data	C	1	▼						
15		Corrupt splitter performing malicious acts(a-g)	C,I,A	4	✖	✓	▼	✓	✓	✖	
16		App Rate – malicious client making too many rq	A	1	✓						
17		App Size-- trying to allocate large memory	A	1	✓						
18		FA Bomber – trying to access files from MP	C	1,5	✓						
19		SQL 123 – multiple flavors of SQL injection	C	5	✓						
20		Run without SPA (control for degraded config.)	A	1,6	✓						
21		Serialization- system exit while deserialize	A	3	✓						
22		Whitelist- sneak in unauthorized object	A,I	1,3	✓						
23		Log Analysis – 3 strikes to adaptive response	C,I,A	1	✓						
24	1	TCP Flood no knock: too many req w/ o knock	A	1	✓						
	2	TCP Flood w knock LDAP and RMI ports (a-b)	A	1	✓	✓					
	3	TCP Flood LDAP and RMI after SPA (a-b)	A	1	✓	✓					
25		Direct Connection from Client to JBoss	I,C	2	✓						
26	1	Dir Con to JBoss from MP w /wo JVM Pol.(a-d)	I,C	2	✓	✓	✓	✖			
	2	Same as above, & try to exec code NH (a-e)	I,C	2	✓	✓	✓	✓	✖		
27		Fork Bomb (a-b)	A	1, 3	✖	▼					
28		Multi user: stress CZ s with client load (a-g)	A	NA	✓	✓	✓	▼	▼	▼	▼

✓ Attack successfully absorbed

▼ Attack succeeded after concessions granted to attackers

✖ Attack not survived; i.e., the attack succeeded

Table 2. Summary Outcome of Other Attacks.

No.	Var	Description	CIA	Claim	
2		SQLI-a: Insert records into the services db	I	5	▼
3	1	SQLI-b: see if db content with apostrophe is denied	A	5	NA
	2	SQLI-b: deny client by db violation with spoofed credential	A	5	NA
6		LogDB-c: corrupt the log db to force a bad action/cover track	I	5	✓
7		SPA-self-auth:	C,I,A	5	✓
8		Corrupt LDAP	C,I,A	1	✓
13	1	Use backtrack and metasploit to escalate access, remote shell	C, I	1	✓
	2	Same as above + the vulnerable class is white listed	C,I	1	✓
	3	Same as above + JVM policies weakened to allow remote con	C,I	1	✓
	4	Same as above + JVM policies turned off	C,I	1	✓
	5	Same as above + SEL weakened to allow remote con	C,I	1	▼
	6	Same as above + SEL Turned off	C,I	1	✖

✓ Attack successfully absorbed

▼ Attack succeeded, but only after concessions granted to attackers

✖ Attack not survived; i.e., the attack succeeded

NA Attack not applicable to the protected service

that did not exist in the service being protected).

In addition, other attacks were not run for various reasons. Attack 1 was an example used to illustrate the red team process and is not included in any results. Attack 5 was deemed to be impossible to launch. Finally, attacks 9, 11, and 12 were deemed to be duplicates of other attacks that were run.

Combining the actual runs and white boarded attacks provide a good test and evaluation coverage over the CIA triad and the APS survivability claims. Many attacks were generic, i.e., depending on the payload they could attempt to compromise a combination of C, I or A of some aspect of the protected service or the APS defense—these runs were counted in all three (C, I, and A) categories. The total number of runs attempted to compromise each of the C, I, A attributes of some aspect of the system and how they fared are shown in the left-hand side chart of Figure 4. The right hand chart shows unique attacks and attack runs aimed at the specific individual APS survivability claims and how they fared. "F" and the green color indicate the number of attacks that failed, i.e., did not work as planned. "P/S" and the yellow color indicates the number of attacks that were partially successful, i.e., that worked after additional concession was provided to the attackers. "S" and the red color indicate the number of attacks that succeeded.

VI. Related Work

Red Teams, defined as "an organizational element comprised of trained and educated members that provide an independent capability to fully explore alternatives in plans and operations in the context of the operational environment and from the per-

attack runs* against C I A

	F	P/S	S
C	18	4	5
I	18	3	5
A	22	6	4

attack runs* against Claims

Claim		1	2	3	4	5	6
Attacks	Total	12	3	4	1	6	1
Runs	Total	20	10	7	6	6	1
	F / P / S	15 / 3 / 2	8 / 0 / 2	4 / 1 / 2	3 / 1 / 2	4 / 2 / 0	1 / 0 / 0

* Actual and whiteboarded

Figure 5. Map of Test and Evaluation Coverage Based on the Attacks Ran and Analyzed during White Board Sessions.

spective of adversaries and others" [12], are not a new concept in cyber security. Various commercial and DoD organizations as well as national labs offer Red Teaming service and training. The Sandia Information Design Assurance Red Team (IDART) [13] is fairly well known and is the process that the AFRL penetration testing team followed during the APS exercises. The U.S. Army University of Foreign Military and Cultural Studies (UFMCS) [14] also offer a course on Red Teaming.

Penetration testing is related to Red Teaming, although the exact definition and distinction from Red Teaming is murky. We see penetration testing as "security testing in which evaluators attempt to circumvent the security features of a system based on their understanding of the system design and implementation" [15]. As such, it is expected that a Red Team Exercise will include some level of penetration testing. The Open Web Application Security Project (OWASP) Testing Guide [16] illustrates a number of penetration testing techniques that are commonly used against web application to assess security.

VII. CONCLUSIONS

We described a successful application of cooperative red teaming to evaluate the R&D prototype of a survivable services architecture. We argue that cooperative red teaming is more effective for research prototypes than other red team approaches, such as an adversarial red team. The cooperative approach supports more cost-effective evaluation by sharing information that focuses on attacks that were likely to have an adverse effect on the system and discard attacks (after analysis and discussion) that would have little or no effect. The red team was able to quickly gain a deep understanding of the system under test due to full access to the software and developers. The red team had full knowledge; there was no attempt to gain security by obscurity. The red team did not have to waste time on penetration attacks to gain access to launch sophisticated attacks - privileged and insider starting points were granted to them.

Likewise, the combination of outsider view and insider insight in the cooperative red-team setting led to better learning and discovery of nooks and crannies of the APS architecture and its defenses. One of the significant results from this exercise was how well the APS software held up against the sustained and new attacks the red team launched. This validates the APS approach and motivates further research, development, and evaluation of the crumple zone architecture for survivability. In addition, successful attacks highlighted places where additional research, development, and evaluation are needed.

As the APS prototype technology matures and becomes more transition-ready, we expect additional experimental evaluation of the technology deployed in target environments.

REFERENCES

[1] M. Atighetchi, P. Pal, A. Adler, A. Gronosky, F. Yaman, J. Webb, J. Loyall, A. Sinclair, C. Payne, "Crumple Zones: Absorbing Attack Effects Before They Become a Problem," *CrossTalk - The Journal Of Defense Software Engineering*, March/April 2011.

[2] M. Rash. (2006) "Single Packet Authorization with Fwknop." [Online]. http://www.cipherdyne.org/fwknop/docs/SPA.html

[3] M. Atighetchi, P. Pal, F. Webber, R. Schantz, C. Jones, J. Loyall, "Adaptive Cyberdefense for Survival and Intrusion Tolerance," *IEEE Internet Computing*, vol. 8, no. 6, pp. 25-33, Nov/Dec, 2004

[4] J. Chong, P. Pal, M. Atighetchi, P. Rubel, F. Webber, "Survivability Architecture of a Mission Critical System: The DPASA Example," 21st Annual Computer Security Applications Conference, 2005, pp. 495-504.

[5] P. Pal, F. Webber, M. Atighetchi, P. Rubel, P. Benjamin, "Automating Cyber Defense Management," 2nd International Workshop on Recent Advances in Intrusion Tolerant Systems, Glasgow, UK, 2008.

[6] P. Pal et al., "An architecture for adaptive intrusion-tolerant applications," *Software: Practice and Experience*, vol. 36, no. 11-12 (September - October), pp. 1331-1354, 2006.

[7] P. Pal, M. Atighetchi, F. Webber, R. Schantz, C. Jones, "Reflections On Evaluating Survivability: The APOD Experiments," 2nd IEEE International Symposium on Network Computing and Applications, Cambridge, MA, USA, April 16-18, 2003.

[8] M. Atighetchi, "Alpha Test Report to Assess 100xJBI Security Capabilities," BBN Technologies, Final Report SPO900-98-D-4000 subcontract 205651, 2007.

[9] M. Atighetchi, "Alpha Test Report to Assess Apollo V1.0 Security Capabilities Towards TRL-level 6 Readiness," BBN Technologies, Final Report of SPO900-98-D-4000 subcontract: 205344 – Mod. No. 5, 2008.

[10] M. Atighetchi, P. Pal, A. Gronosky, "Understanding the Vulnerabilities of a SOA Platform – A Case Study," 9th IEEE International Symposium on Network Computing and Applications, Cambridge, MA, USA, 2010.

[11] P. Pal, F. Webber, R. Schantz, "The DPASA Survivable JBI – A High-Water Mark in Intrusion-Tolerant Systems," EuroSys Workshop on Recent Advances in Intrusion-Tolerant Systems, Lisbon, 2007.

[12] DoD Dictionary. Joint Publication 1-02, DOD Dictionary of Military and Associated Terms 08 November 2010, as amended through 15 February 2012. [Online]. http://www.dtic.mil/doctrine/dod_dictionary/

[13] Sandia National Laboratories. (2012, Mar) IDART. [Online]. http://idart.sandia.gov/

[14] US Army. (2012, Mar) University of Foreign Military and Cultural Studies. [Online]. http://usacac.army.mil/cac2/UFMCS/

[15] DoD, "Communication Security (COMSEC) Monitoring and Information Assurance (IA) Readiness Testing," DoD Instruction, 8560.01, 2007.

[16] Open Web Application Security Project (OWASP). (2008, November) Testing Guide. [Online]. https://www.owasp.org/index.php/OWASP_Testing_Guide_v3_Table_of_Contents.