The Journal of Systems and Software 000 (2017) 1–18

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

On early detection of application-level resource exhaustion and starvation

Mohamed Elsabagh*, Daniel Barbará, Dan Fleck, Angelos Stavrou

George Mason University, Fairfax, VA 22030, USA

ARTICLE INFO

Article history:
Received 29 March 2016
Revised 10 November 2016
Accepted 24 February 2017
Available online xxx

Keywords:
Resource exhaustion
Starvation
Early detection
Probabilistic finite automata
Kernel tracing

ABSTRACT

Software systems are often engineered and tested for functionality under normal rather than worst-case conditions. This makes the systems vulnerable to denial-of-service attacks, where attackers engineer conditions that result in overconsumption of resources or starvation and stalling of execution. While the security community is well familiar with volumetric resource exhaustion attacks at the network and transport layers, application-specific attacks pose a challenging threat. In this paper, we present Radmin, a novel system for early detection of application-level resource exhaustion and starvation attacks. Radmin works directly on compiled binaries. It learns and executes multiple probabilistic finite automata from benign runs of target programs. Radmin confines the resource usage of target programs to the learned automata and detects resource usage anomalies at their early stages. We demonstrate the effectiveness of Radmin by testing it using a variety of synthetic and in-the-wild attacks. We provide a theoretical analysis of the attacker's knowledge of Radmin and provide a metric for the degree of vulnerability of a program that is protected by Radmin. Finally, we also compare the accuracy and effectiveness of two different architectures, Radmin which works in both the user and kernel spaces, and URadmin which works solely in user space.

1. Introduction

Availability of services plays a major – if not the greatest – role in the survivability and success of businesses. Recent surveys[1,2] have shown that IT managers and customers alike tend to prefer systems that are more often in an operable state, than systems that may offer higher levels of security at the expense of more failures. This means that any disruption to the availability of a service directly translates into a loss of productivity and profit. Businesses invest in deploying redundant hardware and replicas to increase the availability of the services they offer. However, as software designers often overlook Saltzer-Schroeder's "conservative design" principle (Saltzer and Schroeder, 1975), systems are often engineered and tested for functionality under normal rather than worst-case conditions. As a result, worst-case scenarios are often engineered by the attackers to over-consume needed resources (resource exhaustion), or to starve target processes of resources (resource starvation), effectively resulting in a partial or complete denial-of-service (DoS) to legitimate users.

A system is exposed to resource exhaustion and starvation if it fails to adequately restrict the amount of resources used or influenced by an actor[3]. This includes, but is not limited to, infrastructure resources, such as bandwidth and connection pools, and computational resources such as memory and CPU time. The attacks can operate at the network and transport layers (Zargar et al., 2013), or at the application layer such as algorithmic and starvation attacks (Crosby and Wallach, 2011; Chee and Brennan, 2010). The asymmetric nature of communication protocols, design, and coding mistakes, and inherently expensive tasks all contribute to the susceptibility of programs to resource exhaustion and starvation attacks. Attacks targeting the network and transport layers have attracted considerable research attention (Groza and Minea, 2011; Rutar and Hollingsworth, 2011; Fu, 2011). Meanwhile, attacks have become more sophisticated, and attackers have moved to higher

* Corresponding author.
E-mail addresses: melsabag@gmu.edu (M. Elsabagh), dbarbara@gmu.edu (D. Barbará), dfleck@gmu.edu (D. Fleck), astavrou@gmu.edu (A. Stavrou).

[1] Availability overrides security concerns, http://www.hrfuture.net/performance-and-productivity/availability-over-rides-cloudsecurity- concerns.php?Itemid=169.

[2] Mobile users favor productivity over security, http://www.infoworld.com/article/2686762/security/mobile-users-favor-productivityover- security-as-they-should.html.

[3] CWE-400: uncontrolled resource consumption, http://cwe.mitre.org/data/definitions/400.html.

http://dx.doi.org/10.1016/j.jss.2017.02.043
0164-1212/

layers of the protocol stack. Since 2010, resource exhaustion attacks that target the application layer have become more prevalent[4] (Chee and Brennan, 2010) than attacks at the network layer and transport layer.

In this paper, we present Radmin, a system for automatic *early* detection of application-level resource exhaustion and starvation attacks. By application-level attacks, we refer to the classes of DoS attacks that utilize small, specially crafted malicious inputs that cause uncontrolled resource consumption in victim applications. To this end, Radmin traces the resource consumption of a target program in both the user and kernel spaces (see Section 3), builds and executes multiple state machines that model the consumption of the target program.

The key observation is that attacks result in *abnormal* sequences of transitions between the different resource consumption levels of a program when compared to normal conditions. By modeling the resource consumption levels as multiple realizations of a random variable, one can estimate a conditional distribution of the current consumption level given the history (context) of measurements. Consequently, the statistical properties of the resulting stochastic process can be used to detect anomalous sequences.[5]

Radmin operates in two phases: offline and online. In the offline phase, the monitored programs are executed on benign inputs, and Radmin builds multiple probabilistic finite automata (PFA) models that capture the temporal and spatial information in the measurements. The PFA model is a finite state machine model with a probabilistic transition function (see Section 4). Both the time of holding a resource and the amount used of that resource are mapped to states in the PFA, while changes in the states over the time are mapped to transitions.

In the online phase, Radmin executes the PFAs as shadow resource consumption state machines, where it uses the transition probabilities from the PFAs to detect anomalous consumption. Additionally, Radmin uses a heartbeat signal to *time out* transitions of the PFAs. Together with the transition probabilities, this enables Radmin to detect both exhaustion and starvation attacks.

Radmin aims at detecting attacks as early as possible, i.e., *before* resources are wasted either due to exhaustion or starvation. Radmin does *not* use any static resource consumption thresholds. Instead, the PFAs capture the transitions between the different consumption levels of different program states, and statistics of the PFAs are used to detect anomalies. The PFAs allow Radmin to *implicitly* map different program states, i.e., program behavior at some execution point given some input, to *dynamic* upper and lower resource consumption bounds.

We quantified the earliness of detection as the ratio of resources that Radmin can save, to the maximum amounts of resources that were consumed in benign conditions (see Section 5). This corresponds to the tightest *static* threshold that traditional defenses can set, without causing false alarms. Radmin has an advantage over all existing defenses that use static thresholds (see Section 10), since exhaustion and starvation attacks can evade those defenses. Exhaustion attacks can consume the highest amounts of resources possible, just below the static threshold (see footnote 4) (Chee and Brennan, 2010). Additionally, starvation attacks, by design, do not aim at directly consuming resources such as attacks that trigger deadlocks or livelocks (Chee and Brennan, 2010).

This paper is an extension of the work we presented in Elsabagh et al. (2015). We make the following main contributions:

1. Radmin, a novel system that can detect both resource exhaustion and starvation attacks in their early stages. Radmin employs a novel detection algorithm that uses PFAs and a heartbeat signal to detect both exhaustion and starvation attacks. Radmin takes both temporal and spatial resource consumption information into account and adds minimal overhead.
2. We implement a prototype[6] that uses kernel event tracing and user space instrumentation to efficiently and accurately monitor resource consumption of target processes.
3. We demonstrate the effectiveness of Radmin using a broad range of synthetic attacks against a number of common Linux programs. We show that Radmin can efficiently detect both types of anomalies, in their early stages, with low overhead and high accuracy.
4. (extension) We demonstrate the effectiveness of Radmin using two common in-the-wild DoS attacks against Apache, namely Apache Killer[7] and Slowloris[8]. We show that Radmin can efficiently and accurately detect both attacks in their early stages.
5. (extension) We implement a new prototype, URadmin, that operates solely in user-space. Uradmin is easier to deploy by end users without requiring kernel tracing. We compare the accuracy and overhead of Radmin to URadmin.
6. (extension) We propose a metric to measure the degree of vulnerability of a program to resource exhaustion that may be triggered using benign inputs that conform to the PFAs enforced by Radmin.

The rest of the paper is organized as follows. Section 2 discusses the assumptions and threat model. Section 3 presents the technical details of Radmin and its implementation. Section 4 describes the models used in Radmin and the detection algorithm. Section 5 evaluates Radmin on synthetic exhaustion attacks and common starvation weaknesses. In Section 6 we experiment with in-the-wild attacks. We compare the detection performance and overhead of Radmin to a user space only solution in Section 7. Section 8 analyzes the attacker's knowledge and proposes a metric for the degree of vulnerability of a program that is protected by Radmin. Section 9 provides a discussion of different aspects of Radmin and possible improvements. We discuss related work in Section 10, and conclude in Section 11.

2. Assumptions and threat model

Radmin's main goal is early detection of application-level resource exhaustion and starvation, which may result in full or partial depletion of available resources (CPU time, memory, file descriptors, threads, and processes) or starvation and stalling. We assume that actors can be local or remote, with no privilege to overwrite system binaries or modify the kernel.

We consider the following types of exhaustion and starvation attacks. First, attacks that result in a sudden surprisingly high or low consumption of resources (e.g., an attacker controlled value passed to a `malloc` call). Second, attacks that result in atypical resource consumption sequences such as algorithmic and protocol-specific attacks that aim at maximizing (flattening) the amounts of consumed resources. Third, attacks that cause stalling of execution, including triggering livelocks or prolonged locking of resources.

In our experiments, although we considered only programs running on × 86 Linux systems and following the Executable and Linkable Format (ELF), the proposed approach places no restrictions on

[4] myths of ddos attacks, http://blog.radware.com/security/2012/02/4-massive-myths-of-ddos/.

[5] Unless stated otherwise, we use "measurements" and "sequences" interchangeably in the rest of this paper.

[6] Source code available under GPLv3 at: https://github.com/melsabagh/radmin.

[7] Apache killer attack - apache server vulnerabilities, https://security. radware.com/ddos-knowledge-center/ddospedia/apache-killer/.

[8] Slowloris - apache server vulnerabilities, https://security.radware. com/ddos-knowledge-center/ddospedia/slowloris/.

Please cite this article as: M. Elsabagh et al., On early detection of application-level resource exhaustion and starvation, The Journal of Systems and Software (2017), http://dx.doi.org/10.1016/j.jss.2017.02.043