OWASP AppSensor - CrossTalk

Article submission for the "Protecting Against Predatory Practices" themed September/October 2011 edition of CrossTalk, The Journal of Defense Software Engineering.

Draft v0.4 (22nd March 2011)

Title

Building Real-Time Defenses into Software Applications

Abstract

Attack-aware software applications respond in real time to suspicious and malicious events with a very low false positive detection rate. The approach is especially suited to software applications with high information assurance requirements such as in the defense, critical national infrastructure and financial service sectors to protect against cyber espionage, tampering, fraud and theft. The OWASP AppSensor Project has developed a methodology, documentation, example code and pilot demonstration of this method of application-specific attack detection and response.

Building Real-Time Defenses into Software Applications

Introduction

Information systems and data are being targetted by advanced attackers. They are skilled, motivated and have excellent tools. They may be backed by organized crime, governments or commercial enterprises. The attackers will relentlessly attempt to access the sensitive and secret data available to software applications by finding and exploiting vulnerabilities in the applications themselves.

In this article we discuss why traditional defenses are not a solution to these types of advanced attack, and explain an initiative from the Open Web Application Security Project (OWASP)[1] which addresses the problem.

Traditional Defensive Measures

A fundamental starting point for secure software applications is to build security in at all stages of the development lifecycle. We need robust code, designed, developed, configured and operated on hardened operating systems. This does not mean the software is impregnable; unknown vulnerabilities almost certainly exist, and attackers examine and probe applications to find them. Once they have identified one or more vulnerabilities, they need to discover ways to exploit them singly, or in combination. It is very hard to detect these custom attacks using traditional defensive measures.

Some techniques commonly stated to defend applications include using transport layer security (TLS/SSL), network firewalls, application gateways (e.g. web application firewalls, XML appliances and cross domain guards) and network/host intrusion detection and prevention systems (IDPS).

TLS protects transmitted traffic between two points to provide confidentiality and integrity. It also provides some degree of assurance in the identity of one or more of the parties. However there is no inspection of the data itself — it simply passes on what was sent. As a defensive measure for a software application, it has no impact on attackers. An attacker's payloads are sent to the server in the same way as normal traffic. Network firewalls are a vital component for network defense, but they allow or deny traffic using a particular port. By necessity web applications need to allow traffic through specified ports (often TCP ports 80 or 443). Attackers of course send their payloads using the same ports. Application gateways are generic solutions to typical problems. For example, even with careful configuration and self-learning, most web application firewalls are operated in detection-only mode, and they have no insight into business logic type of attacks. Like network firewalls, IDPS has no real concept of good and bad application usage.

Real-Time and Proactive Defensive Measures

Consider the analogy of control instrumentation in an industrial production process. Temperature, pressure, mass, velocity and volumetric, etc measurement points are added to quantify the state of key aspects over time. These data may be used in localised control (e.g. a float switch in a liquid storage vessel), but more often they are integrated into an overall process control system, which aggregates information from many sensors, and uses feed-back and feed-forward control mechanisms to react to changes and maintain a desired state.

Most current software applications are like industrial processes without this control instrumentation — there are no sensors and no intelligence about what is going on — they are blind to what is happening internally. Yet applications have access to powerful processing capabilities in their supporting hardware and software systems.

The right way to detect advanced attacks is within applications themselves, by adding application-specific security controls which detect malicious activity. The application has full knowledge about the business logic and the roles & permissions of users—it can make informed decisions about mis-use, and identify and stop attackers with a very low false positive rate. It also does this in real time. This is not a generic approach because the controls are related to the actual application, the operational processes

it enables and the risk level.

Within the Application is Best

By building measures within software applications, they become attack-aware, By being attack-aware it is possible undertake pro-active defense. Within the application:

- data is fully available (e.g. decrypted)
- the context is known
- the business logic is known (e.g. purposes, paths and sequencing)
- information about the user is fully accessible (e.g. role, status, previous actions)
- program flow can be altered in real time

Similar ideas already exist in some software, but these are usually implemented as isolated processes and some may be undertaken reactively to events, or performed largely in a manual way. Some examples of these include:

- counting multiple failed authentication attempts to lock a user account
- detecting use of the TRACE HTTP method to block requests
- checking the IP address during a session and terminating the session if the IP address changes
- logging unexpected requests
- investigating suspicious events at a later date.

The concept described in this article focuses and formalizes this approach. It is about implementing proactive measures to add instrumentation and controls directly into an application in advance so that all these events (and more) are centrally analysed and responded to.

It also means there is no need to attempt to replicate application logic in other devices.

Normal vs. Malicious Behavior

The concept relies on being able to differentiate between normal and malicious behavior. That is one of the biggest hurdles when security controls area added outside the application. When detection is undertaken in the application itself, the application itself has access to the complete context of a request and information about the user. This means the application can identify attempts to bypass access control, or bypass client-side input validation, access processes in the wrong order, etc; none of these can happen by accident. And this knowledge, is not available to protection devices on the network.

This fine-grained ability to differentiate between normal, suspicious behavior and attacks can also be used to improve the usability of human interfaces with the software applications. A pure whitelist imposed by a gateway device might block a password which has a trailing line feed character appended; an application might consider this invalid, but trim it since it could easily be the result of a copy & paste action by the user.

Evasion and Unknown Attacks

Input data are decrypted and canonicalised within the application and therefore detection within the application is less susceptible to advanced evasion techniques such as obfuscation.

Since attackers have to probe applications, even unknown attacks can be discivered which would otherwise not be known about.

AppSensor Project[2]

The concept described above was initially developed under the OWASP Season of Code 2008[3] by

Michael Coates. The output was a document defining the AppSensor conceptual framework, available as a printed book[4], or a free PDF[5]. Over the last two years, a reference implementation in Java[6], further guidance[7], demonstration pilot[8], presentations and videos[9] have been created to support the initiative.

Like in an industrial process control systems, a whole range of sensors can be considered for incorporation by the instrumentation engineer. Most of the equivalent application sensors are located within the application code itself (like process control sensors in vessels and pipes), but AppSensor also has the notion of inputs from other "external" devices.

In application software, these external inputs could come from diverse sources such as:

- user profiling
- reputational services (e.g. credit checking, risk tolerance, threat level)
- fraud monitoring system
- CRM system
- file system integrity monitoring
- network security devices (e.g. network and application firewalls).

As an example, the software could alter its security posture based on an external threat level rating such as the Defense Condition Level (DEFCON)[10].

Some detection points identify specific events defined as a generic signature; others consider behavior over time. Furthermore, it is possible to detect and differentiate between both suspicious and attack events, providing additional insight into how an application is being used, and abused. Some detection points also look at the results, or output, rather than user inputs. For example the number of records returned by a database or XPath query, or the integrity of relational database tables or log files.

AppSensor has defined over 50 detection points in twelve categories (see Table 1). These range from signature type detection points which typically look for particular events within a user's single request/response cycle, to behavioral type detection points which assess events over longer time periods (e.g. a single user session, a finite time period, the life of the application).

Table 1: AppSensor Detection Point Categories

Type	Code	Name
Signature	RE	Request Exceptions
	AE	Authentication Exceptions
	SE	Session Exceptions
	ACE	Access Control Exceptions
	IE	Input Exceptions
	EE	Encoding Exceptions
	CIE	Command Injection Exceptions
	FIO	File IO Exceptions
	HT	Honey Trap
Behavioral	UTE	User Trend Exceptions
	STE	System Trend Exceptions
	RP	Reputation

Full descriptions, considerations to maintain a low false positive detection rate and examples of each detection point have been documented[11].

Thresholds are set for each detection point, or group of detection points. These then act like tripwires which determine when the application undertakes a response.

Rich Responses

Traditional defenses offer binary response options such as allow/deny or log/block (see Table 2). They are often limited to a single dimension - the particular network connection contravening a "rule".

Table 2: Traditional Binary Responses

Response	Examples
Allow	Log events only
Deny	Block IP address

The AppSensor concept describes a wide spectrum of possible responses[12], and when combined with the very low false positive detection rate, defenses do not need to be left in logging-only detection mode for fear of preventing valid traffic. We can also move to a two-dimensional view, where responses may be directed against a single connection, a single user, but now also a group of either of these (e.g. based on location, role, or behavior), or all users. Table 3 defines the responses possible together with software application examples.

Table 3: AppSensor's Range of Rich Responses

Response	Examples
Logging Change	Capture sanitised request headers and response bodies
	Full stack trace of error messages logged
	Record DNS data on user's IP address
	Security logging level changed to include 'informational' messages
Administrator Notification	Email alert sent to everyone in the administration team
	SMS alert sent to the on-call administrator
	Visual indicator displayed on an application monitoring dashboard
	Audible alarm in the control room
Other Notification	Broadcast event to SIEM
	Signal sent to upstream network firewall, application firewall (e.g. XML, web) or load balancer
	Alert sent to fraud protection department
	Record added to server event log
	Event highlighted in a daily management report
	Email alert to staff member's manager
	Proactive entry added to customer support system
User Status Change	Internal trustworthiness scoring about the user changed
	Reduce payment transfer limit for the customer before additional out-of-band verification is required
	Reduce maximum file size limit for each file upload by the forum user
	Increase data validation strictness for all form submissions by this citizen
	Reduce the number of failed authentication attempts allowed before the user's account is locked (ASR-K below)
Proxy	Requests from the user invisibly (from the user's perspective) passed through to a hardened system
	Request are proxied to a special honeypot system which closely mimics or has identical user functionality
User Notification	On-screen message about mandatory form fields
	On-screen message about data validation issues
	Message sent by email to the registered email address to inform them their password has been changed
Timing Change	Extend response time for each failed authentication attempt
	File upload process duration extended artificially
	Add fixed time delay into every response
	Order flagged for manual checking
	Goods despatch put on hold (e.g. despatch status changed)
Process Terminated	Discard data, display message and force user to begin business process from start
	Redirection of an unauthenticated user to the log-in page
	Redirection to home page
	Display other content (i.e. terminate process but display the output of some other page without redirect)
	Redirection to a page on another website
Function Amended	Limit on feature usage rate imposed
	Reduce number of times/day the user can submit a review
	Additional registration identity validation steps
	Additional anti-automation measures (e.g. out-of-band verification activated, CAPTCHA introduced)
	Static rather than dynamic content returned
	Additional validation requirements for delivery address

	Watermarks added to pages, images and other content
	Additional human interactive proof challenges added due to the number of incorrect guesses of CAPTCHAs outnumbering the correct guesses by more than a certain factor (e.g. Token bucket idea)
Function Disabled	Add friend' feature inactivated
	'Recommend to a colleague' feature links removed and disabled
	Document library search disabled
	Prevent new site registrations
	Web service inactivated
	Content syndication stopped
	Automated Direct Debit system turned off and manual form offered instead
Account Logout	Session terminated and user redirected to logged-out message page
	Session terminated only (no redirect)
Account Lockout	User account locked for 10 minutes
	User account locked permanently until an Administrator resets it
	One user's IP address range blocked
	Unauthenticated user's session terminated
Application Disabled	Website shut down and replaced with temporary static page
	Application taken offline
Collect Data from User	Deploy additional browser fingerprinting using JavaScript in responses
	Deploy a Java applet to collect remote IP address
	Deploy JavaScript to collect information about the user's network

Thus for example, when a small number of detection points have been activated in an application providing access to secret information could alert supervisors and either decrease response times or proxy requests to a less sensitive system. A smart meter might disable a diagnostic function which appears to have been compromised, whilst continuing to allow electricity supply and billing. A banking application might reduce the payment transfer limits when suspicious behavior is detected.

When behavioral aspects over time (a request/response, a session, a time period) are included, the application becomes capable of rich three-dimensional responses. The behavioral aspects can span further than application restarts or even application life cycles from deployment to disposal, since data from other systems which pre-date the application may be incorporated.

Cross Integration

As mentioned above, AppSensor can potentially make better use of information from other security devices to contribute to its pool of information for attack detection, increasing the value of those other systems.

Apart from consuming data from other systems and devices, AppSensor can pass back intelligence to other systems. This might be used as part of the real-time response:

- locking a user's single-sign-on account
- removing some permissions for a user from a physical access control system
- setting a warning flag about a customer in a CRM
- blocking a session ID or IP address at a network firewall

Equally, data can be federated to other systems such as SIEM, IDS and fraud monitoring systems.

Information Insight and Value

Applications contain considerable knowledge about users and their activities. Without AppSensor so much intelligence is being wasted. System owners are left with tools and devices which have to guess about what is going on. AppSensor provides insight into whether, and how, attacks are occurring.

And of course, if this is a public-facing commercial application, some of these data would be valuable to those interested in understanding and improving usability, reducing drop-out rates, etc.

Applicability

AppSensor is a comprehensive proactive approach, rather than reactive,which can be applied to applications throughout the enterprise. It reduces the risk of unknown vulnerabilities being exploited. It greatly increases the visibility of suspicious events and actual attacks. This can provide additional information assurance benefits:

- reduced security risks to data and information systems
- improved compliance
- reduction in the consequences of data breaches.

In turn, these can provide improved service levels and resilience and, for commercial organizations, competitive advantage.

The recent announcement[13] from the U.S. Defense Department that it is planning to spend $500 million to research new cyber security technologies including "active defenses" - technologies that detect attacks and probes as they occur - suggests there is growing interest in this area.

???

Implementation

Existing AppSensor documentation already includes guidance on detection point considerations, determining the malicious intent, monitoring system trend events and implementation guidance. However, the planning stages are probably the most time-consuming aspect of implementing AppSensor and with the recent addition of further detection points and response actions, this document provides more information on the process and considerations to take.

AppSensor has a very low false positive detection rate. The implementation must ensure that this is not compromised by adding inappropriate detection points, or implementing them in a way which leads to a greater number of false positives. The method presented also tries to build in consideration of business operations and usability, so that not only is a low false positive rate maintained, but processes are not unduly disrupted and the users are not subjected to difficulties through simple human error. In other words, building in a degree of human fault tolerance.

Although AppSensor works best within the authenticated portion of an application, it is also possible to apply the principles to other areas. In the latter, the range of "normal behavior" may be wider, the identity and location of users may be harder to pinpoint and some detection points may no longer be necessary. But the same benefits are possible.

???

Conclusions

???

About the Authors

Michael Coates

Michael Coates has extensive experience in application security, security code review and penetration assessments. He holds an MSc in Computer Security from DePaul University and a BSc in Computer Science from the University of Illinois. Michael is the creator and leader of the AppSensor project, a contributor to the OWASP Top 10 project and a frequent speaker at security conferences. As web security lead at Mozilla, Michael protects web applications used by millions of users each day.

Email michael.coates@owasp.org

John Melton

John Melton is a software architect and designer with particular knowledge of security for online systems. He is the AppSensor project's Development Lead, and has been responsible for most of the example code developed. John studied at the University of North Carolina at Charlotte. John is a Senior Information Security Engineer at Wells Fargo and lectures at UNC Charlotte.

Email jtmelton@gmail.com

Colin Watson

Colin Watson's work involves the management of application risk, building security and privacy into systems development and keeping abreast of relevant international legislation and standards. He holds a BSc in Chemical Engineering from Heriot-Watt University in Edinburgh, and an MSc in Computation from the University of Oxford. He contributes to a number of OWASP projects and is a member of the OWASP Global Industry Committee. Colin is a consultant and co-founder of Watson Hall Ltd.

Email colin.watson@owasp.org

1 About the Open Web Application Security Project, OWASP, http://www.owasp.org/index.php/About_OWASP

2 OWASP AppSensor Project, http://www.owasp.org/index.php/Category:OWASP_AppSensor_Project

3 OWASP Summer of Code 2008, http://www.owasp.org/index.php/OWASP_Summer_of_Code_2008

4 OWASP AppSensor - Detect and Respond to Attacks from Within the Application, v1.1, Michael Coates, Paperback, 48 pages, Lulu, http://www.lulu.com/product/paperback/owasp-appsensor/4520003

5 OWASP AppSensor - Detect and Respond to Attacks from Within the Application, v1.1, Michael Coates, https://www.owasp.org/images/2/2f/OWASP_AppSensor_Beta_1.1.pdf

6 OWASP AppSensor Developer Guide, http://www.owasp.org/index.php/AppSensor_Developer_Guide

7 OWASP AppSensor Implementation Planning Workbook, http://www.owasp.org/index.php/File:Appsensor-planning.zip

8 OWASP AppSensor Live Tutorial, http://www.defendtheapp.com/

9 OWASP AppSensor Media, http://www.owasp.org/index.php/OWASP_AppSensor_Project#tab=Media

10 Defense Readiness Condition (DEFCON), United States Armed Forces

11 OWASP AppSensor Detection Points, http://www.owasp.org/index.php/AppSensor_DetectionPoints

12 OWASP AppSensor Response Actions, http://www.owasp.org/index.php/AppSensor_ResponseActions

13 Pentagon Seeks $500 Million for Cyber Technologies, 15 February 2011, Bloomberg, http://www.bloomberg.com/news/2011-02-15/pentagon-seeks-500-million-for-cyber-research-cloud-computing.html