

An Autonomic Security Monitor for Distributed Operating Systems

Alvaro Arenas

Department of Information Systems and Technologies

Instituto de Empresa Business School

Madrid, Spain



Objective of the Work

- Develop the infrastructure for monitoring security properties in a Grid-based OS (XtreemOS)
 - Open distributed systems
 - Scalability? Dependability? Performance?
 - What to monitor in relation to security?

Outline

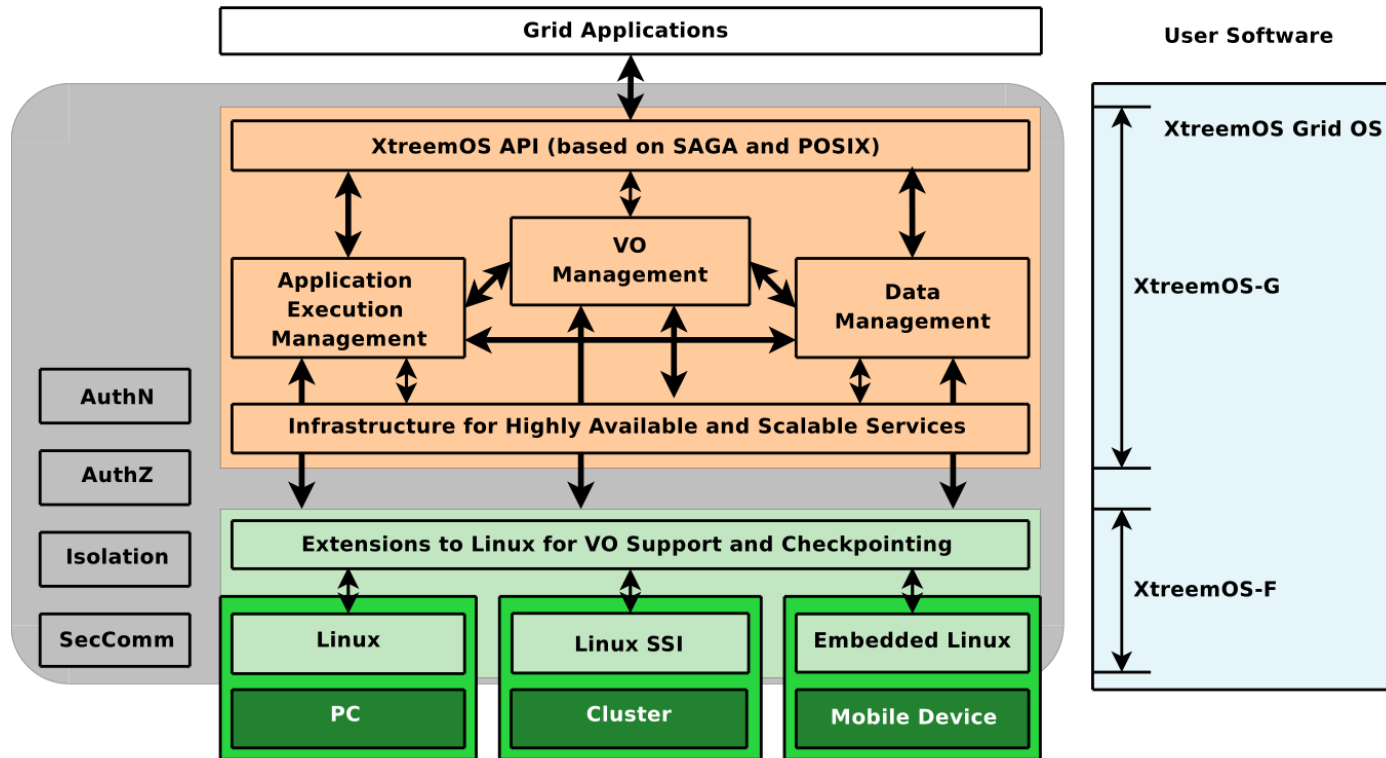
- Overview of XtremOS
- Monitoring in XtremOS
- Monitoring Security Properties
- Exploiting Monitoring Information

An Overview on XtremOS

- An open source Linux-based Grid Operating System with native VO support
- **Grid Operating System:** a comprehensive set of cooperating system services providing a stable interface for a large-scale wide-area dynamic distributed infrastructure
- **Transparency:** Grid like Linux
 - Users may be unaware of Grid issues
 - If you know Linux, you know Grid
 - Grid used like any other interactive system



XtreemOS Software Architecture



Jobs in XtremOS

- Jobs are central to Grid systems
- Processes are to jobs as threads to processes
 - Job – Job unit – process - threads
- Jobs are managed via Linux signals
 - Including new Grid signals

Nodes and Resources in XtremOS

- A resource is where a job (and their processes) is executed
 - The job can use more than one resource
- Depending on the set of services appearing in a node, XtremOS classifies nodes in three categories
 - **Clients** simply access the XtremOS system
 - **Resource nodes** provide execution and storage capabilities.
 - **Core nodes** offer VO administration, job management, ...

Outline

- Overview of XtremOS
- Monitoring in XtremOS
- Monitoring Security Properties
- Exploiting Monitoring Information

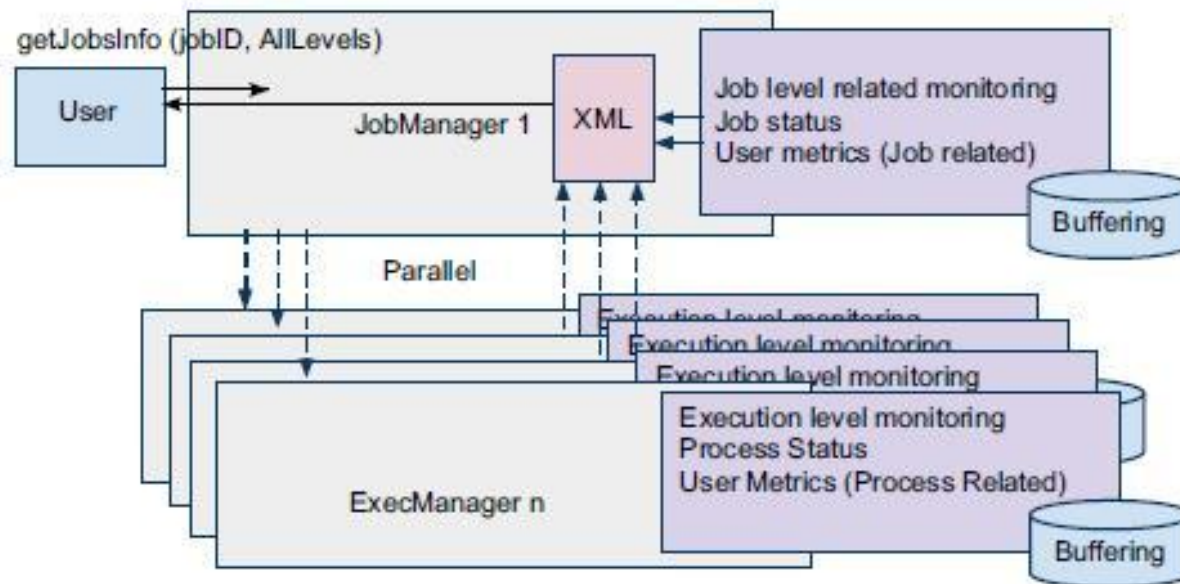
Monitoring in XtreamOS

- Aim at answering the question “what is my job doing?”
- Performed by the Application Execution Management (AEM) component
 - Monitor the execution of jobs and the status of resources
 - System events
 - User/application events
- Monitoring at the level of jobs, processes and threads
 - Events are communicated using the XtreamOS publish/subscribe subsystem (Scalaris)

Monitoring Terminology

- **Events** (also called **metrics**) is a kind of monitoring data
 - It can be created by the user or by the system
 - E.g. CPU utilization, network traffic, job status, ...
- An **event/metric** has a description, a **value type** (Boolean, integer, timestamp, . . .) and a **scope**
- A **scope** is the place where the info belongs
 - JOB, JOBUNIT or PROCESS

Components in the General Monitor in XtremOS



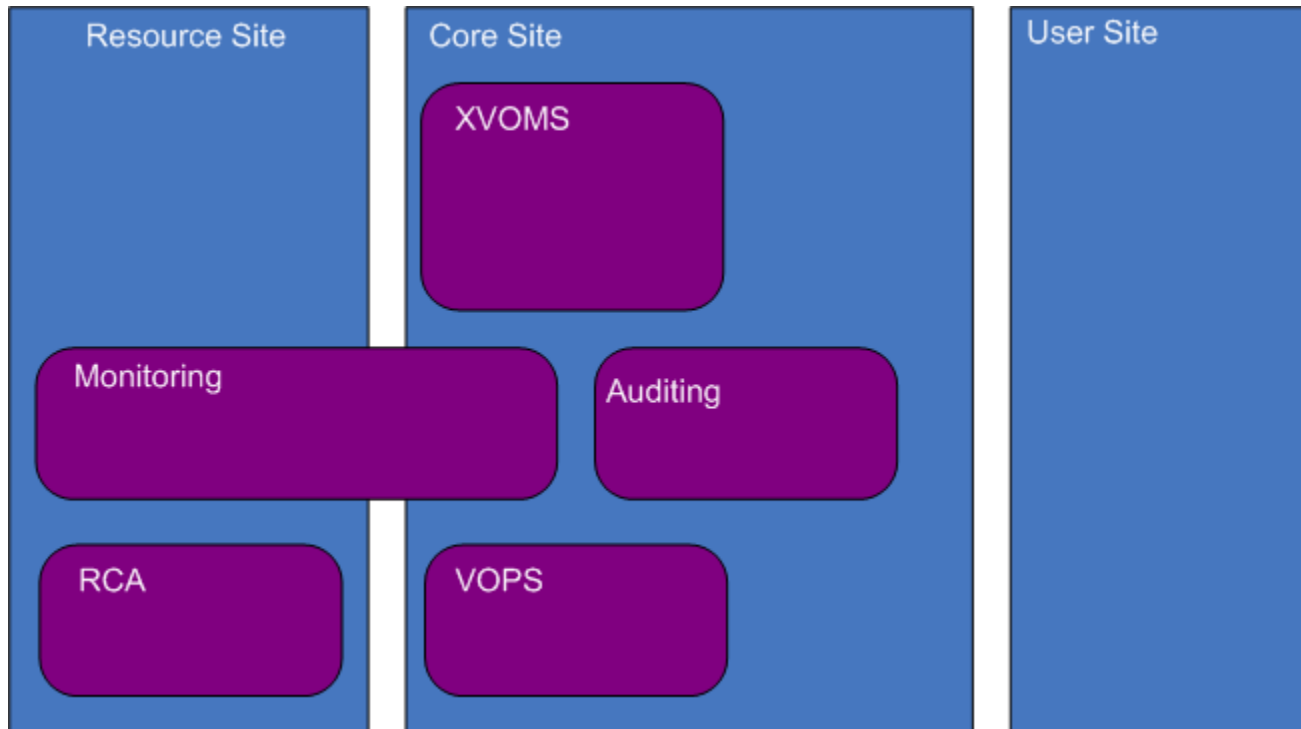
Implementation

- Two alternatives for implementation
 - LTTng: Linux Trace Toolkit new generation
 - Monitors kernel events and implements buffering
 - E.g.: monitor thread/process status changes
 - Obtain process status from a daemon in the nodes reading `/proc/pid` and storing the resulting values
- Interface: *getJobsInfo*, *getJobsMetrics*, *addJobMetric*, *setMetricValue*, *removeJobMetric*, ...

Outline

- Overview of XtremOS
- Monitoring in XtremOS
- **Monitoring Security Properties**
- **Exploiting Monitoring Information**

XtreemOS Security Services

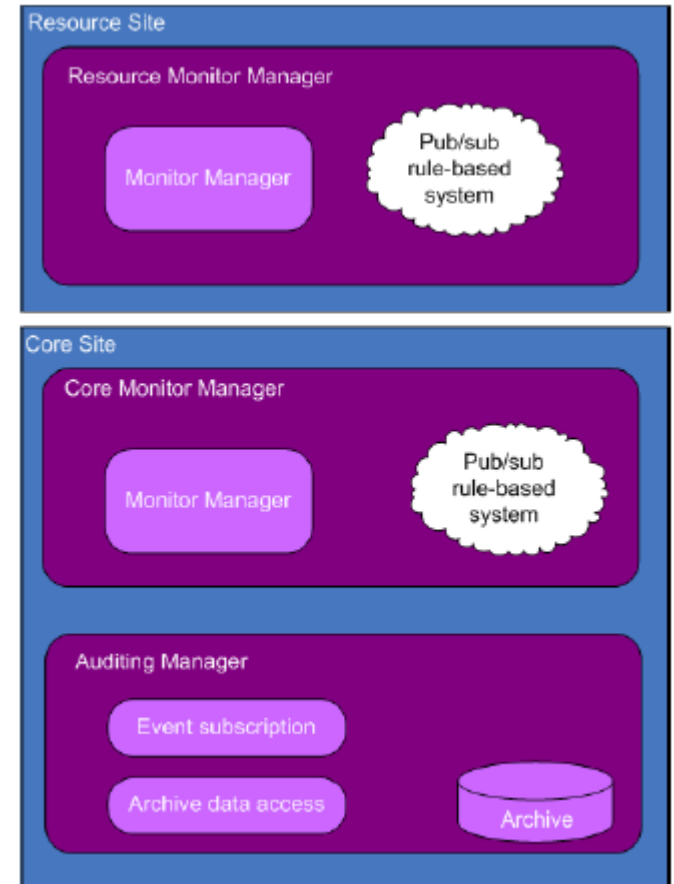


What Security Information Should We Monitor?

- CPU/Memory usage
 - Useful for **controlling the usage of resources** according to CPU/memory usage
- Network traffic
 - Incoming network packets are analysed locally by a resource and such info is aggregated and compared with information from other resources
 - Accumulated knowledge enables **detection and prevention of DDoS attacks**
- Resource availability/usage
 - Determines trends on resource availability and usage
 - Time periods during which the **resources are scarce** and times when **resources are abundant**
- Access to resources
 - Access to resources may be governed by dynamically changing attributes
 - Monitoring of the attributed may be used to implement **Separation of Duty or Chinese Wall** policies

Secure Monitoring and Auditing

- Resource monitor is responsible for monitoring resource-related events
 - Such as CPU utilisation, memory usage, network traffic, ...
- Core monitor monitors events related to the Grid and VOs, as well as info from core services such as VO management
- Auditing service stores monitoring information into databases for historical analysis



Implementation

- Both secure monitoring and auditing services implemented in Java
- Interface allows users to define monitoring rules which describes what to monitor, to whom to notify events
- Interface Monitor: *saveEvent, saveMetric, addNotification, subscribe, unsubscribe*
- Interface Auditing: *addArchiveRule, cancelArchiveRule, query*



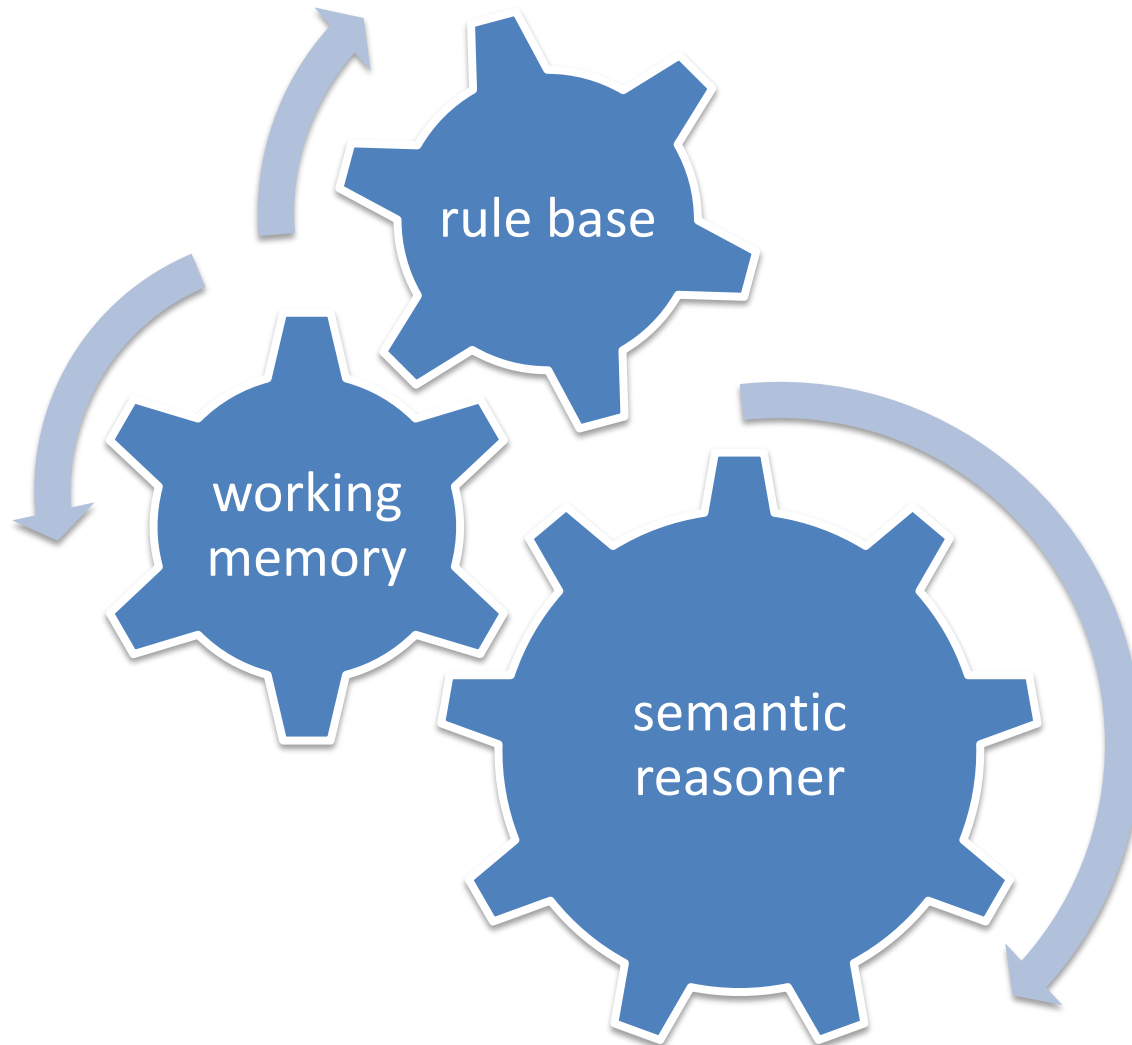
Outline

- Overview of XtremOS
- Monitoring in XtremOS
- Monitoring Security Properties
- **Exploiting Monitoring Information**

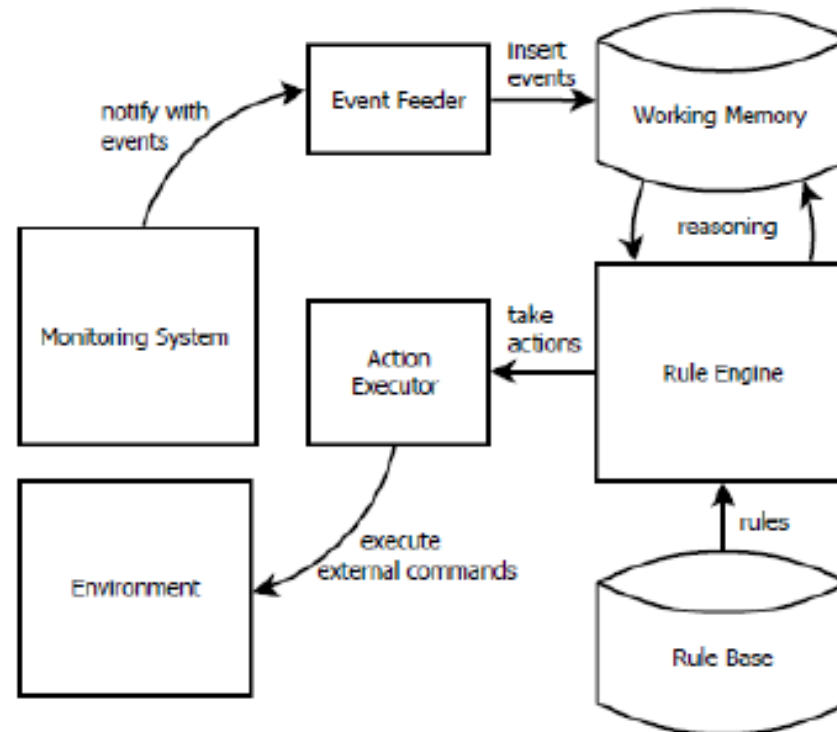
Autonomic Management of Security Events

- Exploit the information collected by the security monitor in order to have a more autonomous behaviour
- Include a rule-based system able to analyse monitoring information in real-time and take corrective actions accordingly

Rule-Based System



Architecture of Autonomic Part



Implementation

- Use Java Drools technology, a platform for defining rules and event processing
- Manager Class encapsulates and hides Java Drools interface in order to simplify rule management
- The Drool rule engine is not thread-safe, but Manager Class synchronises all necessary methods, this the whole system may be used by multiple threads, as its methods are non-blocking

Example of Monitoring Events

```
public class FloatMetric extends Metric {  
    private float value;  
    (...)  
}
```

```
public class Metric extends MonDbRecord {  
    private String metricName;  
    private ValueUnit valueUnit;  
    (...)  
}
```

```
public abstract class MonDbRecord implements Serializable {  
    private long id;  
    private Date timeStamp;  
    private Domains domains;  
    (...)  
}
```

```
public class Domains implements Serializable{  
    private long id;  
    private String vo;  
    private String userGroup;  
    private String user;  
    private String job;  
    private String resource;  
    (...)  
}
```



Some Rules

```
rule "Track users CPU usage"  
no-loop  
when  
    UserCpuMetric( $user : userid, $newvalue : value ) over window:length( 1 )  
    $fact : UserCpuUsage( userid == $user, value != $newvalue )  
then  
    modify( $fact ) {  
        setValue( $newvalue )  
    };  
end
```

```
rule "Limit CPU usage"  
when  
    CpuQuota( $limit : quota )  
    UserCpuUsage( $user : userid, value >= $limit )  
then  
    UserJobBlocked block = new UserJobBlocked();  
    block.setUserid( $user );  
    insertLogical( block );  
end
```



Conclusions

- Monitoring security is seen as a particular case of XtremOS monitoring
 - Relevant events and user metrics are monitored and aggregated in order to determine potential security problems
- Rule-based system allows one to manage dynamically the monitoring information collected from resources, jobs and Vos
- Future work: exploit the monitoring information for more complex security rules
 - Runtime detection of malicious job signatures that could imply viral behaviour

An Autonomic Security Monitor for Distributed Operating Systems

Alvaro Arenas

Department of Information Systems and Technologies
Instituto de Empresa Business School
Madrid, Spain



Thank you !

