An Adaptive and Dependable Distributed Monitoring Framework

Teemu Kanstrén, Reijo Savola
VTT Technical Research Centre of Finland
Oulu, Finland
{teemu.kanstren,reijo.savola}@vtt.fi

Sammy Haddad, Artur Hecker
Telecom ParisTech
Paris, France
{sammy.haddad,artur.hecker}@enst.fr

Abstract — **This paper discusses several relevant aspects of performing monitoring in the context of software-intensive systems. The focus is especially on cases where the observed system is distributed, and the monitoring system needs to be secure, dependable and capable of adapting to a number of dynamic scenarios during the system evolution. Based on the analysis of monitoring needs in this type of domain, a set of core requirements for providing a monitoring framework for these domains is defined. To address these requirements, a high-level reference architecture for a monitoring framework is presented. These requirements and reference architecture provide a basis for designing different monitoring systems.**

Keywords – monitoring, framework, security assurance, adaptation, dependability.

I. INTRODUCTION

Collecting data about different aspects relevant to the behaviour, configuration and deployment of a software-intensive system during its lifetime is important for many different purposes. Together with the different partners in the CELTIC BUGYO Beyond project, we have collected and analysed a set of core requirements from the viewpoint of building a monitoring framework (MFW) for continuous monitoring of security assurance-related information. Additionally, we have analysed a set of existing MFWs for different domains with similar requirements. Based on this set of requirements, we present a high-level architecture for a MFW and discuss how it addresses the different requirements. This paper is an extension of our previous work [1], updated with the latest evolution in our research.

There are many potential application domains of operational measurement, including supporting software (SW) quality assurance activities such as testing and debugging that require collecting data about system behaviour for analysis [2]. Quality assurance can also be associated with monitoring different aspects of an operational system, such as the quality of service in a telecommunications network [3], compliance of dynamic systems with their requirements [4] and the security compliance of the system [5]. The monitoring functionality can also be used to provide automated actions such as restarting failed services and sending failure notifications [6]. In the case of some systems, the data collection itself is the main purpose and goal of the system. For example, scientific experiments can require collecting large amounts of data for research purposes [6,7].

Although the measurement systems target different domains, they all share the goal of capturing information (monitoring) about different properties that are important to the functionality and security of the observed system. From the information monitoring perspective, they all thus share the same set of core requirements. In addition, each application of a MFW can also have its own set of specific requirements such as optimization for real-time processing [2] or high-performance capture of large data sets over large networks [6]. In this paper, we focus on the core set of requirements and classify these to five different categories: intrusiveness, security, dependability, scalability and runtime-adaptation. These categories of requirements represent different viewpoints of the monitoring functionality.

First, monitoring typically disturbs the observed system to some extent (commonly referred to as a *probe effect*), affecting its reliability and dependability. Intrusiveness needs to be minimized. Second, in many cases the collected information provides sensitive information about the observed system and its behaviour and thus this information needs to be protected from any unauthorized access (security). Third, to enable best possible use of the information, the MFW design should be highly dependable in order to assure that the data is available even in case of failures in the observed system in order to allow for analysis of the issues based on the captured information. Fourth, it is important for the MFW to be scalable for use in different types of observed systems, where the scale of distribution can vary greatly. As many modern systems evolve during their lifetime or exhibit high dynamics in their structures, the MFW should also support runtime adaptation to account for the measurement needs, the evolution of the observed system and the MFW itself.

We start by defining a core set of requirements that we have synthesized for these types of systems and present high-level reference architecture as a basis for a MFW to address these requirements. The main contribution of this paper is the identification and analysis of core requirements relevant for building a dependable, secure and adaptive distributed monitoring framework, and providing a reference architecture that addresses these requirements.

The rest of this paper is structured as follows. In Section II, we present a number of existing MFWs for different purposes and domains. In Section III, we synthesize a common set of requirements based on the review of requirements in the domains these frameworks have been applied to as well as the requirements we have collected and analysed together with different partners in the BUGYO Beyond project. In Section IV, we present a high-level MFW design to address the requirements. Finally, conclusions end the paper.

II. Background and Related work

In this section, we present the background concepts related to the discussion in this paper and a number of existing MFW designs for specific domains including a discussion on their relation to our work presented in this paper.

A. *Background Concepts*

Runtime monitoring is defined here as the act of collecting information about a system during its operation. A MFW is a system that performs the collection (and possible processing) of this information and provides it to interested clients who make use of this information. The raw information is typically detailed in nature and thus is mainly used by other SW components to perform further processing of the data, such as visualizations for human users or issuing system control commands for automated adaptation based on algorithmic analysis. Additionally, a MFW can also provide higher-level events describing observations made of the observed system by local processing nodes (e.g. service failure). These events can also describe information about the MFW itself, such as the evolution of the system infrastructure (e.g. the addition, removal or reconfiguration of a MFW element). These basic concepts apply on different scales, from embedded SW to networked systems of systems, where only the scale of the component and its interconnection mechanisms change.

The basic (measurement) information about a system is captured by a *probe*. A probe is defined here as a SW or hardware (HW) component capable of performing a specific measurement on the observed system. These can be either commercial-off-the-shelf (COTS) or custom-made components. These probes are linked to the MFW to provide the monitoring data to be processed by the MFW components and its client applications. The MFW implements the infrastructure to capture the data over the different observed system elements. In different types of systems, the MFW infrastructure is thus also different, such as spread over the network in distributed systems or distributed over the components of an embedded SW. In this paper, we focus on monitoring of distributed systems, although we note that the principles can for the most part be applied at different scales provided that the system is designed using proper architectural properties (e.g. see [8] for embedded SW).

Specific types of probes can be identified and used depending on the needs of systems. For example, Wang et al. [4] define four types of probes:

- Instrumented probes with analysis – probes embedded in a system which process raw data before outputting it.
- Instrumented probes without analysis – probes embedded in a system which directly provide captured raw data.
- Intercepting probes with analysis – external probes (e.g. network analyzer HW) that provide some analysis of the raw data as output.
- Intercepting probes without analysis – external probes that provide the raw data as captured.

Together these probes provide the raw measurement data to fulfil the system measurement needs. A mediator component is then used to provide information access to clients.

B. *Related Work*

In this section we describe a number of existing measurement frameworks presented in the literature, and discuss their relation to our work presented in this paper.

The current study presents a part of the CELTIC BUGYO Beyond project results. This project follows up on the CELTIC BUGYO Project. During the BUGYO project, a preliminary version of MFW dedicated to security assurance was designed and implemented [5]. The main challenge of this framework was to provide a solution to the problem of data collection related to the security-enforcing mechanisms of the observed system. In this previous work, the measurement architecture was not deeply investigated and it was expected to be installed in an ad hoc manner. We partly extend this work but take a new and more systematic approach to defining the requirements for a MFW and its practical deployment, with a particular focus on addressing the dynamic aspects present in real systems. This MFW was described to consist of three types of components with the following roles. A component called *probe agent* controls a specific type of a probe, a *MUX agent* provides multiplexing of data over subnet boundaries, and a *server agent* handles centralized processing of the monitoring data and interfacing with a client system. The observed data is not sampled at a high frequency or in great amounts and thus they do not optimize the communication infrastructure but rather focus on using XML-based protocol formats. One main weakness of this MFW was the fact that it was not adaptive and could only run on a fixed architecture. For the rest of this paper, we adopt the agent terminology from this previous work and the associated agent types (basic MFW components).

A MFW for capturing large amounts of scientific experimentation data is presented in [7]. It is aimed at capturing massive amounts of data from sensor electronics located next to the Large Hadron Collider (LHC) detectors with high-performance, scalability and dynamic monitoring requirements controlled by a flexible and configurable GUI. The fundamental feature here is the routing of many sorts of data (from simple parameters to histograms or event fragments). They use Common Object Request Broker Architecture (CORBA) [9] as the protocol between the MFW components to support standards-based implementation on different platforms. A separate data stream is used to pass the high-volume data through the MFW system, and a separate channel is used to pass control requests and events related to the MFW. This aims to provide high-performance data capture with specialized data streams.

A similar MFW (called MonALISA) for capturing data about scientific experiments is described in [6]. Its first purpose was the analysis and processing of large-scale data-intensive grid applications. Their targeted challenge is to provide a MFW able to manage monitoring aspects related to storage, networks and a large number of running applications in near real time. The design of MonALISA is inspired by the Jini [10] architecture where each agent in the framework is described and registered as dynamic services. The MonALISA design is divided into three layers, each with a specific functionality. The first layer provides dynamic registration and discovery of all MFW components. The second layer consists of the monitoring services, where each provides data and events of a defined type, and the others can subscribe to these data types. The third layer is called the proxy layer and it handles the communication of the MFW components over network filters (e.g. firewalls), providing access control to the monitoring data and to the management of the MFW components.

A monitoring framework for Voice-over-IP (VoIP) traffic is presented in [3]. The main goal of this framework is to allow high-speed, real-time and efficient (in terms of CPU use) analysis of communications traffic. In this framework, data, i.e. packets, are collected and processed at the kernel level using a plugin-based server architecture with Session Initiation Protocol (SIP) [11] and Real-Time Transport Protocol (RTP) [12]. A set of filters is defined for the different plugins to define which data has to be processed by each plugin. The plugins can then in turn be configured to process the filtered data according to their functionality. A library of functions is provided to implement common data processing functions and an easy configuration and update mechanism is provided for the plugins.

A MFW for web service requirements is presented in [4]. The data to be monitored is inferred from the web-service specification (WSDL) of the observed system. A set of constraints over the observed data and events is used to describe the allowed values and event sequences. This includes frequency, interval and ordering of events. Some of their probes provide basic values while others process values before passing them forward. Probes are generated based on the WSDL.

A MFW focusing on embedded real-time systems is introduced in [2]. It focuses on high-performance requirements in a single (non-distributed) embedded system. It defines a custom binary protocol and a set of optimized services for capturing high-frequency data from a running system, focusing on minimizing its use of the observed system resources (e.g. memory and CPU load). It provides a set of services to enable the implementation of features to monitor different aspects of the system (e.g. task scheduling).

The GEMOM (Genetic Message Oriented Secure Middleware) EU FP7 Project has developed an adaptive security, resilience and Quality-of-Service (QoS) MFW [13,14] for distributed information systems. The data communication is arranged by a publish/subscribe mechanism, and separate modules handle authentication and authorization functionality. Specific QoS, resilience and security properties can be set for the different data processed by the GEMOM system, allowing for customization of authorization and authentication for different elements.

From the design factors of the abovementioned measurement frameworks, we apply the following MFW design patterns for the purposes of the framework discussed in this study:

- Specific adaptation layer for measurement targets [5].
- Repository of available probes for specific measurement targets [2].
- Simple interface to integrate custom probes into the overall measurement system [2, 3].
- Usage of widely supported protocols [7].
- Configurability of operational probes [7].
- Optimized communications related to expected measurement data communication patterns [5,7].
- Usage of separated dedicated communication channels [7].
- Customization of different aspects of the MFW based on module composition [3].
- Provide a registry that is dynamically updated to reflect available measurements and probes [6].
- Use specific components to handle connection and processing over network subdomains [5,6].
- Describe the measurements with a common set of properties [4,6].
- Optimize the availability and reliability of measurement infrastructure [13].
- Secure the communication data [13].

In the following sections, we describe why these design patterns are relevant for a measurement framework. We also present a reference architecture that takes these and other MFW requirements into account.

III. REQUIREMENTS

This section describes a core set of issues we have identified for the type of monitoring addressed in this paper. Based on the analysis of these issues, a set of requirements for the design of a MFW is presented. These issues have been divided into five different categories: scalability, runtime adaptation, correctness, intrusiveness and security. An overview of these categories and their properties is shown in Figure 1. These categories are described in the following subsections.

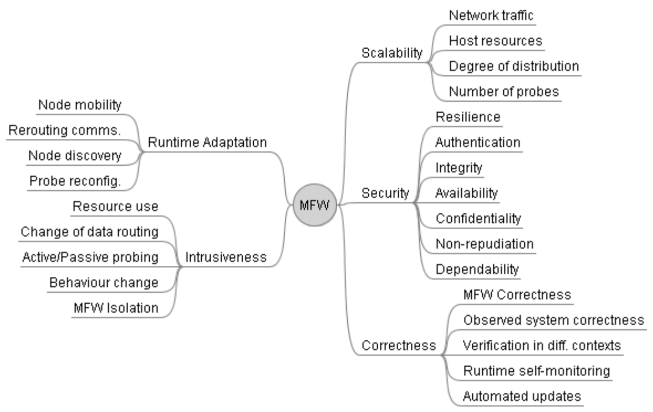

Figure 1. Requirements overview.

A. Scalability

From the basic scalability perspective, the MFW infrastructure needs to be able to handle varying amounts of observed events and data captured from systems with different degrees of distribution. In this case, scalability issues include supporting varying amounts of probes and resource limitations in processing large amounts of data (e.g. CPU and memory use, network bandwidth). For the MFW infrastructure this can require use of alternate communication paths in order to avoid causing network congestion. This can require use of specific MFW components to be deployed at different locations, such as localized processing and multiplexing nodes, routing nodes and probe control nodes.

The scale of distribution of a MFW depends on the nature of the target of measurement. In some cases, it may be possible to perform all measurements from a single centralized location when the measurement nodes already have the ability to provide the required measurement access and functionality remotely, such as over remote Secure Shell (SSH) connections. In more highly distributed systems, a more distributed MFW architecture is needed. In this case, measurements need to be performed for a large number of nodes (possibly divided into subnets), not all of which are reachable from a single centralized (monitoring) location. This requires specific router elements to be deployed to handle connections over the different nodes and subnets as divided by filters (e.g. firewalls).

With regards to the distribution aspect, the addressing scheme used for specifying which measurements can be supported also needs to be considered. Depending on the type of measurement being performed, one may wish to address more than one target in a measurement request. For example, in the security assurance domain, one may request information on the encryption strength of all routers deployed and visible to the MFW. The task of the MFW is then to automatically scale these requests without overloading the network.

As the observed system evolves, the need for such specialized MFW elements may increase, meaning that the addition of these elements needs to be supported in an evolving manner through runtime adaptation.

B. Runtime Adaptation

Modern distributed systems are rarely static in their formation. Even if the system itself supports scaling to different sizes of network infrastructure, it is not enough to simply support deploying these different types of architectures. The system must be able to evolve to support changes in scale while in operation. In today's distributed systems, nodes come and go, and the same nodes can move to different locations in the network. In addition to physical changes in the infrastructure, system reconfigurations can also change the available communication channels. To address these scenarios, the MFW must be able to keep the mapping of its elements up to date and in synch with the changes in the observed system (i.e. which element does a probe measure and which measures it provides), and adapt to changes in the available communication channels between the MFW infrastructure nodes. In order for client applications to build functionality on top of the monitoring data, the MFW must also abstract the changes so that a client can request a specific measure and does not need to worry about the dynamic aspects of evolution such as node mobility.

For routing of data through the MFW infrastructure, various issues need to be considered. If the used communication paths become unavailable, communication channels need to be rerouted. Yet, in many situations and to fulfil the non-intrusiveness objective, the routing in the observed system should not be changed to accommodate the needs of the MFW. For this reason, the MFW must provide adaptation capabilities to address these needs in an optimally adaptive and non-intrusive way.

When a measure has become unavailable but later becomes available again, the MFW needs to reconnect to the corresponding probe and to notify any client applications making use of this information that it has again become available. Similar notifications must be given for all changes in MFW evolution.

As new MFW infrastructure elements are installed, the MFW needs to link them to enable applications to use the data they provide, or for the MFW to use them for the control of the measurement (e.g. data routing). A basic means is manual configuration of all added nodes, but runtime reconfiguration by the MFW also needs to be configured. The optimal case is that the discovery and integration of new deployed MFW elements are automated.

Changes (evolution) in the observed system's infrastructure also affect the monitoring needs. As a basic feature, a MFW must enable the user to deploy changes to the active measurement requests. Additionally, depending on the types of probes deployed, different types of actions are necessary. Embedded probes are assumed to come with the deployed components of the observed system, and require the deployment of a matching MFW component to integrate them with the MFW infrastructure. External probes, on the other hand, can be used to monitor more than one target, and thus, in this case, the MFW needs to reconfigure the (existing) external probe to also monitor the new element that was introduced. For all new probes, there must exist a suitable deployed MFW element to integrate them with the MFW infrastructure.

C. Intrusiveness

Monitoring always has some effect on the target of observation – this is often referred to as the *probe effect*. In the case of the MFW, this translates to the effects that the installation of the MFW components and their use have on the observed system. Installation of the components themselves alone has an effect on the use of some system resources if the components are installed within the observed system itself and/or share some resources with it. In the case of external probes, installation should have only a minor effect on the target of observation unless the installation itself somehow changes the behaviour of the system, such as the routing of messages.

Probes that share resources with the target of observation have an impact on its available resources such as CPU load and memory consumption. These probes are typically embedded probes, running as part of the observed system or on the same HW. The operation of external probes can also affect the observed system in a similar way. For example, routing data through an external probe can add network latency to the observed system. For another example, activating features and tools, such as SFlow [15] and Netflow protocol-based tools [16], on router ports can be a drain on router CPU cycles. Some of these effects can be minimal yet still cause unwanted effects to, for example, the timing of the system, changing it from the original design specifications. When a probe is more active, for example, sending packets to different network elements, it can invoke untested or unintended behaviour in a system, causing side-effects such as a complete crash or incorrect results.

In the scope of the BUGYO Beyond project, we have identified the following possible probe effects:

- Consumption of hosting device resources (CPU, memory, disk).
- Consumption of network bandwidth.
- Deployment of conflicting SW components, such as shared libraries.
- A probe accessing restricted resources in the system, causing the system to enter a different state. For example, triggering an alarm and a countermeasure.
- A probe accessing system resources (e.g. a file) and causing a lock, while another part of the system is trying to use the same resource.
- Change of privileges required for an agent (or probe) to be able to perform its tasks.
- Opened gates (e.g. ports) to access information in the observed system.
- Changes performed by the probe on the measured system, e.g. by some eXtensible Configuration Checklist Description Format (XCCDF) [17] configurations or benchmarks.
- Active probing of a certain interface of a target of measurement can trigger some unintended features or even crash the whole system, if it has some unintended features linked to this interface.

This set is largely related to our target domain of security assurance, where numerous measurements are done through the file system and with shell scripting. Although most of these are very generic and we see them as being applicable across many domains, some different probe effects may apply in different domains.

The predictability of probe effects is important in assessing which probes to take into use and how. However, combining any arbitrary probes and systems produces unpredictable effects. Even with tested systems and probes, the exact available resources and system configurations can vary and thus it is not possible to provide exact guidelines for any possible probe effects for probe-system combinations. Providing a large-scale analysis of possible probe effects is out of the scope of this paper but it should still be stressed that it is important to take into account the possible probe effects and where possible test for them. Depending on the potential impact, in some cases it may be unfeasible to deploy probes on operational systems with which they have not been tested previously. Knowing the possible probe effects in this case helps in studying them for a specific system and probes. Overall, it can be said that an active probe is more likely to have a higher probe effect than a passive probe.

However, even if the intrusiveness of probes cannot be fully predicted, the MFW should be able to monitor its impact on the observed system and take any measures it can to address these impacts (e.g. re-route communication or start load-sharing components). This requires providing self-monitoring and adaptation features based on specific probes and analysis of the monitoring system properties (e.g. data-stream latency). It also requires providing self-adaptation features such as tuning the probe measurement frequency where observed to be needed.

Even if the implementation and use of probes cannot be constrained, the MFW infrastructure itself, however, can and needs to be designed to be minimally intrusive (i.e. to minimize its probe effect). To achieve this, the different constraints described above need to be considered, as does how they can be minimized in the design and implementation of the different components of the MFW infrastructure. One main goal in avoiding intrusiveness for a system should be the isolation of the MFW from the observed system as much as possible (system independence). Any problems in the MFW (e.g. SW crash) should not affect the observed system, and ideally switching the MFW on or off should have no effect on the observed system. Similarly, the network traffic and any other shared resources of the two entities should be separated as much as possible. Where full separation is not possible, the goal should be to aim for as much separation as possible. Virtual separation should be applied where physical separation is not applicable. When separation is applied, it needs to be applied in both functional and non-functional domains, e.g. for control.

D. *Correctness*

During the service lifecycle various problems may arise. The correctness viewpoint needs to consider the correctness of both the MFW components and their behaviour. It must also consider their effects on the observed system and how this may affect its correct behaviour. In this regard, the correctness aspect of the MFW is closely related to the intrusiveness viewpoint. Addressing this includes both providing for verification of the behaviour of the whole MFW and all its components before deployment, and monitoring and addressing any problems found in its operational use. Generally, the following three main approaches can be identified from the correctness viewpoint:

- Simplicity in the design in order to minimize the possibility of new problems
- Update mechanisms for the MFW infrastructure
- Testing and verification mechanisms to assure the correctness of the implementations

In practice aiming for simplicity would mean encapsulating more complicated processing of the measurements, control structures and similar properties at a higher level in a separate measurement processing and control layer. However, the choice of how much functionality is incorporated in the MFW infrastructure components (probes and agents) is a choice of tradeoffs in the extent to which it is in the interests of the user for processing to be performed locally (to save on resources such as network bandwidth) or on a dedicated server-agent component that can be hosted on dedicated HW.

Testing and verification are needed to assure the correctness of the implementations in different contexts and configurations (in development and while in operation). Since the infrastructures that are the target of observation can vary greatly and be highly dynamic, the correct functionality of the MFW also needs to be evaluated in different contexts. It is important to verify both the correct functionality of the individual components and their interactions. To support this, the MFW design needs to provide suitable interfaces to ensure required testability features (e.g. as described in [18, 8]).

Even with the best verification and testing techniques, unanticipated situations will arise and failures in the operation of the MFW will be found during its operation in various complex contexts and interactions with different components and systems. To address any errors found during operational use, update mechanisms are needed for the different components of the MFW infrastructure.

A set of specific issues needs to be considered for updating components in an operational system. Three main issues for this are referential transparency, state transfer, and mutual references [19]. Referential transparency refers to updated components not having enough information to notify all related components that are connected to the component in question whether they need to be notified about the update. State transfer refers to the requirement of transferring the state of the component from the previous version to the new updated version. For example, a probe agent may have a state describing its current activity and the state of the connected probe, and all this information needs to be transferred to the new version of the updated component. Mutual references refer to the problem of requiring simultaneous updates on several components due to their mutual dependencies, e.g. during an interface update.

Related to these issues, the updating of the MFW components translates to different requirements for the different components of the MFW infrastructure. This includes both rerouting of communication channels and reconfiguration of any specific functionality, such as the frequency of sampling with a probe. In addition to fixing errors and vulnerabilities, implementation updates are needed in order to enable new features for the different components that are developed to address new requirements for the MFW infrastructure. As many modern systems have high availability requirements, it cannot be assumed that they can be taken down for these updates, and similarly the MFW should keep providing its functionality with minimal interruption.

As the MFW generally collects data from external tools or built-in functions in the observed system, it cannot provide generic support for updating these components external to the MFW. In this regard, the MFW must rely on the user(s) having the possibility to deploy the updates using support provided by the observed system and its components. This can take different forms, such as specific interfaces provided by the updated component (e.g. SSH scripting access) or remote management and deployment tools.

E. *Security*

Considering the information processed and the functionality provided by the MFW, security issues also need to be adequately considered. Confidentiality, Integrity and Availability (CIA) [20] are the most widely recognized basic dimensions of information security. To minimize the possible impacts of the different security threats on the MFW and the observed system, every means to guard the observed system from abuse of the MFW infrastructure should be taken.

Considering the communication of the measurement data itself, the following viewpoints need to be considered:

- Data protection: Data can be communicated through a non-secure network and should be protected, e.g. through encryption.
- Registration, authentication and non-repudiation: To protect from unauthorized access, each of the new deployed probes needs to be registered and authenticated, and every message between the probes and the aggregation server needs to be authenticated.
- Services availability: To be useful, the monitoring data needs to be available at all times, optimally even when the rest of the system is unavailable, allowing investigation of system failures. To achieve this goal, the measurement infrastructure and aggregation services should be able to communicate data at any time or keep a log file.

The MFW provides continuous monitoring data captured from the observed system. Much of the information that is being collected can describe sensitive details about the observed system, and thus protecting the confidentiality and integrity of this information is as important as the security of the observed system itself.

As the MFW needs to be closely tied to the observed system to enable making the required observations, different factors having a bearing on its impact on the observed system from the security viewpoint must also be considered. The possibility of introducing new vulnerabilities into the observed system by using the MFW and launching attacks against the observed system through the MFW needs to be minimized. In certain attack scenarios, the attacker might even gain unauthorized control of the whole target system through the MFW.

Instrumentation of a given observed system by a continuous in runtime management needs to have adequate security controls in place based on holistic risk-driven analysis. The holistic approach should cope with the resilience of the complete resulting system, including the original observed system, necessary changes to accommodate the MFW functions and the new MFW-specific parts. The ultimate goal is to achieve a resulting system that is more resilient overall than the original system.

To address these issues, the MFW infrastructure must be resilient to malware, relevant vulnerabilities, security attacks and other faults with security effects. For example, Denial-of-Service (DoS) attacks can affect the availability of the observed system through misuse of the MFW and its resource consumption. In addition to considering external threats such as DoS from unauthorized external attackers, internal threats also need to be considered, such as message flooding attacks or even more harmful attacks by a malicious authorized node.

Overall, an attacker should not be able to use the MFW infrastructure to cause harm to the system under observation by causing it to take excessive resources, gain control of the system, install unauthorized software or otherwise change the system behaviour in an undesired way. The MFW infrastructure should not provide any means of performing attacks against the observed system. Similarly, the MFW infrastructure itself must be protected from attacks and failures. This includes authentication, integrity and availability controls for the communication between the different elements of the MFW infrastructure, proper authorization and data confidentiality and integrity countermeasures. As it is in general not possible to predict and prevent every possible failure or issue in advance, non-repudiation of the data, messages and actions should be at an adequate level to enable audit trail and forensics activities.

An important security-related aspect to be considered is resilience. Even if the observed system is experiencing problems, the MFW should continue its operation as far as possible in order to provide information about the current state of the observed system and to allow for an expert to take corrective actions based on the gathered information. This means the MFW should, if possible, be even more dependable and resilient than the rest of the system components. In some cases, only the latest information may be of interest. However, in other cases the historical data could also be important and in these cases the local nodes need to keep a historical data storage when disconnected from the overall measurement infrastructure.

Overall, the main aspects related to the security viewpoint can be summarized as follows:

- Reduce the chance of using the MFW as a tool to mount attacks against the observed system through, for example, probe registration messages as a DoS attack vector or gaining control of the system.
- Minimize the use of shared resources between the MFW and the observed system to reduce the possibility of it being used as an attack vector.
- Provide sufficient security controls for the measurement infrastructure.
- Use proper authentication and authorization mechanisms for measurement communications.
- Ensure the confidentiality of all processed data as it describes sensitive security-related measurements, both in individual probes and at the overall server level.
- Provide non-repudiation to properly ensure auditability and forensics.
- Ensure any confidential data inside the MFW such as credential and key information for probes (e.g. password).
- Isolate the MFW from the observed system as far as possible.

IV. A Reference Architecture

This section describes a MFW Reference Architecture (RA) and discusses how it addresses the requirements discussed in the previous section. Figure 2 shows a high-level overview of a potential MFW infrastructure with different components. In this architecture, the server-agent is the component that handles the high-level control and processing of the overall measurement infrastructure. The router-agents are the components that take care of transmitting the required data across the network. Probe agents are responsible for controlling different probes at local nodes.

In a practical measurement infrastructure, different compositions of these nodes are to be expected and while the general types of nodes can be described as probe, router and server agents, the exact composition of the individual nodes can also vary. For example, the probe agents need to be adapted to the exact measurement needs, the router agents to the exact routing needs and the server agent to the actual processing needs.

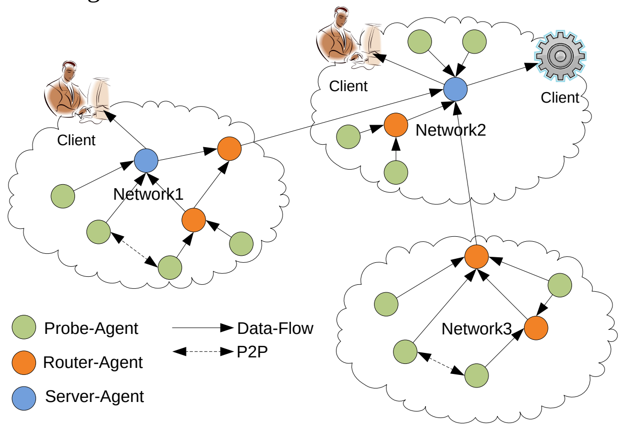

Figure 2. Overall conceptual architecture.

Figure 3 shows a layered view of the same conceptual architecture. It is shown as a set of layers, where each of the layers consists of a set of one or more separate components deployed over the network. Each component in these layers only communicates with the components in the layers directly above and below it. These form a layered architecture as described by Buschmann et al. [27]. This encapsulates the functionality of each layer as a separate, reusable and maintainable piece. A separate communications channel is used as a peer-to-peer (P2P) overlay. While this overlay is not strictly speaking a layer between any of the other layers, but rather stretches over all the other ones, it is shown here separately due to its central role in achieving many of the set requirements. These components will be discussed in more detail next.

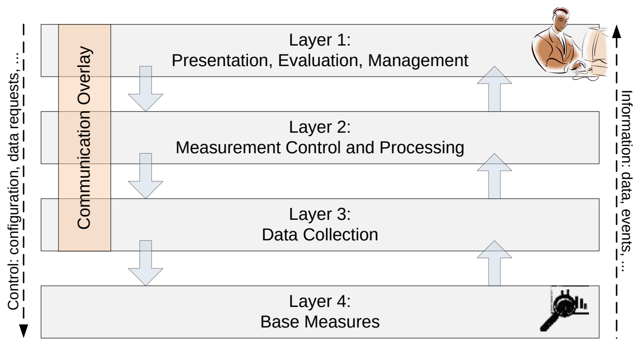
Layer 1:
Presentation, Evaluation, Management
Layer 2:
Measurement Control and Processing
Layer 3:
Data Collection
Layer 4:
Base Measures
Communication Overlay
Control: configuration, data requests, ...
Information: data, events, ...

Figure 3. A conceptual layered architecture.

Layer 1 corresponds to the client using the MFW. Typically the communication with a specific client is based on the custom interfaces of the client and needs to be implemented separately. Generally, the functionality of these messages can be described as including the communication of basic measurement values, MFW infrastructure events and passing of measurement requests and configuration messages between the client and the MFW.

Layer 2 handles data processing and configuration control over the MFW infrastructure. For the MFW infrastructure, it handles making notifications of events such as disconnected probes to the client, as well as taking any defined adaptation actions based on observed events. This layer also handles mobility of nodes and keeping the infrastructure model for the client in synch (and abstracting away the dynamic aspects) with the dynamically evolving infrastructure in order to provide correctly over time the required measures to the client.

Layer 3 collects the data from the different probes in the system. It takes the data provided by the probes and communicates this to the MFW control and data processing layer. This layer views the passed data simply as information to be communicated and thus sets no restrictions on it, e.g. whether it has been processed before or is “raw” probe data.

Layer 4 includes the actual probes. It is responsible for handling the base measures for the MFW. The components in this layer are typically not a part of the MFW itself but rather separate components such as commercial off-the-shelf or open-source software components. These are used from the Layer 3 MFW components to perform the actual required measurements and to acquire the requirement measurements.

Considering the links between these four layers, the first layer only communicates with the second layer through a centralized server providing access to the MFW services and measurements. The second layer provides this interface to the first layer and communicates with the distributed nodes of the measurement infrastructure. It issues measurement requests, configuration commands and similar messages to the distributed measurement and router nodes according to the requests from the first layer. It does not see the actual probes performing the measures. The fourth layer only communicates with the third layer by receiving measurement instructions (requests) and configuration data, and providing measurement data in response. This layer is unaware of any other layers and only sees the third layer that serves as the adaptation layer between the probes and the MFW agents. The fourth layer is also the only layer that needs to be directly in touch with the actual observed system in order to provide the measurements. The other layers can be separated to address the isolation requirements. This is where the communication overlay comes in.

The communication overlay is both separate from and interlinked with the different layers. As an overlay it can be separated from the different MFW components by use of standardized network interfaces. At the same time, by binding the MFW components to this interface, it forms a separate, dedicated communication channel for these components. This also allows for handling secure communications through mechanisms in this overlay. The fourth layer is basically separate from this overlay as it communicates not only with the MFW components only but also with probes in the actual observed system. Thus the use of an overlay here allows for minimizing intrusiveness as well.

These layers, the overlay and some generic components are discussed in the following subsections. We start with generic components shared by the different layers, and follow with layer- and overlay-specific components.

A. *Component Platform and Automated Updates*

Using a component platform as a basis for the MFW provides several advantages. In our implementation, we have used the Open Services Gateway initiative (OSGi) [21] component framework, but other component frameworks with similar functionality can also be used. However, in the rest of this paper we discuss this from the perspective of the OSGi platform. This type of a component platform as a basis provides for loosely coupled services that can be composed in different ways, allowing for easy extension and customization of chosen parts of the MFW. In the following subsections the different layers of the MFW are described as a set of plugins that can be combined in different ways in the different MFW elements to customize and distribute the functionality of the MFW in different ways.

Additionally, OSGi is used as a basis to provide a uniform and automated update mechanism for all components. The goal is to perform updates without interruption of the functionality of the updated components. On a general level, an update can be considered to comprise installing new components, removing existing components or updating existing component implementations. OSGi provides the basic framework for this in allowing for runtime installation and removal of components, although it needs to be extended to make this happen in a distributed fashion.

To support automated updates, each component of the MFW needs to define an interface for reading its internal state so that the new version can replicate the state of the component being replaced. For states that cannot be easily transferred (e.g. data elements whose processing has been started), we use a different replacement strategy. A new version of the component (service) is installed beside the existing one in the same OSGi instance and new messages are routed to the new instance. The old version is removed after it has finished processing all its queued input. This mechanism allows updates of single components. In cases where the ordering of the data is not important this approach is effective; otherwise queues would need to be used to buffer new messages between the old and new versions [22].

Large-scale updates (e.g. interface changes) are less common but can have a potentially large impact and require a more centralized approach. To support this, a central update manager is needed within the MFW. This keeps track of component versions (which can be queried from the components) and tracks dependencies between components. This enables the management of large-scale updates if there is, for example, an interface change that requires all components to be updated. From the operation correctness viewpoint, these are also more central for the operation of the MFW, as a failure in the update manager will likely cause the complete system to enter a failure state. Thus it also requires more thorough testing and verification. These features of an update system are common to any system embedding such features and we do not go into details in depth here but rather refer to other existing works such as [22]. However, we do note that while much of the existing work on live updates focuses on addressing complex cases, we in practice have experienced that such complex needs in a MFW are rare. The data passed in usually is not sequentially dependent but rather timestamped, thereby simplifying these requirements and the required technical solutions.

In addition to updating elements of the MFW itself, we also expect most probes to require updates, for example, to add new features and address found issues. One option for this would be to use the MFW to perform these updates using a common update interface shared by MFW probe agents. Each probe agent would translate the update requests to a suitable format for the specific probe in question. However, due to the complexity (in organizational policies and technical challenges in heterogeneous systems) of allowing and providing for automated install of arbitrary SW on various systems, we rely on existing tools being used for remote SW deployment to update the (custom and COTS) probes. Thus we do not provide technical solutions in the architecture of the MFW itself for this, but instead rely on existing tools intended for this purpose. However, we identify this as an important aspect of a usable MFW.

However, we rely on updating probe configurations through the probe agents, each of which is expected to have the ability to read and set the configuration of the managed probes. This information is then communicated to the client in order to allow the provision of features for probe configuration management. This is based on the expectation that each configuration can be described with a set of basic data elements (e.g. numerical values and text strings) and each probe agent is capable of describing the probe configuration parameters. Client users are also expected to know enough about their system in order to understand the information required to configure these probes. In special cases where the probes are built to be managed by the MFW itself, they can also be reconfigured by the data processing and control layer.

B. *The Communication Overlay and Protocols*

In this subsection we discuss the communication aspects of the MFW. This includes both the overlay that is used to provide the separate communication channel for the different MFW components and the protocols used to communicate between the different MFW components over the overlay.

The P2P Overlay

The need for a separate communication overlay has been discussed before. Here, we discuss the use of a P2P-based private, dedicated overlay as a means to solve many of the requirements for runtime adaptation and intrusiveness. The overlay we have used in practice is the Armature P2P overlay, the foundations for which were laid out in [23].

This overlay provides a separate communication channel for the measurements that is dedicated for the MFW. As it is based on Virtual Private Network (VPN) technologies and features discovery of nodes and overlay-layer routing, it provides a separate, secure, adaptive and relatively non-intrusive virtual communication channel. This overlay thus provides us with a virtual (logical) communication channel that can be dedicated for the use of the MFW and deployed alongside the actual nodes of the observed system, which is the type of solution usually needed for practical systems. We call our implementation of this overlay Armature. For better separation, Armature nodes are contained within small virtual machines running embedded-system language interpreters. This addresses many of the intrusiveness and security requirements of the MFW. Besides separation from the system, it also simplifies deployment, as the overlay configuration changes needed for security controls, such as firewalls, in the observed system are the same across the system.

Due to the peer-to-peer nature of the overlay, it also addresses many of the runtime adaptation requirements of the MFW. When deployed, it forms a virtual network among the deployed peers and automatically calculates optimal routes to uphold a robust communication mechanism between the different nodes. When parts of the overlay are separated, they form their own subsystem. When they are re-joined, they will automatically connect to each other and reform new routes as needed. For the reset of the MFW infrastructure (agents) the use of the overlay is simple, as it is simply visible as another network interface on the hosts where it is deployed.

Communication Protocols between the Components

In addition to a communication channel, protocols for passing the data over this channel are needed. An overview of the communication protocols we use is shown in Figure 4, building on the ideas from [5]. A custom communication protocol is needed for each probe as these are assumed to be COTS or custom-made SW components that are not designed with the MFW in mind. It is thus necessary to have an adaptation layer for reading their values and controlling their configurations. This adaptation is handled by the probe agents, consisting of both generic and probe-specific parts. The generic part is shared by all the probe agent implementations and provides a ready implementation of the functionality needed to communicate with the server and router agents over the chosen middleware protocols.

For communications between the different MFW agents, we use a general description of a Message Oriented Middleware (MOM) technology. Using a suitable MOM helps hide the details of communication from different agent implementors and provides all the benefits of MOM use, enabling us to rely on well tested and designed components for this otherwise complex task. It also provides for several choices depending on the specific needs of the target system. For example, one may use open-source protocols such as XML-RPC [24] or Representational State Transfer (REST) [25] web services, or commercial MOM implementations, such as ICE [26], that are heavily optimized for performance and other factors.

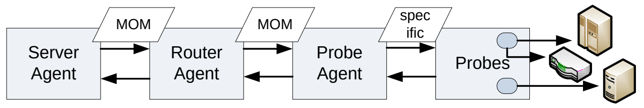

Figure 4. Communication protocol data formats.

C. Layer 4: Base Measurement Layer

The base measurement layer is based upon probes (COTS or custom-made) that are embedded or deployed in the observed system infrastructure in order to collect any kind of data that is considered relevant and can be collected with a probe. A probe agent connects each probe to the MFW and controls the probes. One probe agent can be mapped to one or more probes, and one probe should be mapped to only one probe agent to avoid synchronization issues.

In addition to providing the basic data, this layer also provides events related to the functionality of the probes and probe agents, including the availability of new probe agents and the loss of a probe (becoming non-responsive). It is basically responsible for describing the available measurements from a probe (its characteristics), controlling it as needed and providing the raw measurements from the probe on request. What is supported depends on the types of probes that are available and the features they support.

In our experience, some different considerations need to be taken into account when implementing probes at this level. In some cases, a specific functionality may be needed to perform a measurement of a specific part of the observed system: for example, to read a specific configuration parameter that can only be read programmatically. While we have used a common platform for the development of our MFW agents due to the benefits this brings, as discussed before, we cannot assume that such platforms can be installed on all nodes where measurements need to be performed. Instead we need to be able to make use of what is available and possible on the target of measurement. The probe agent is then a component used as the bridge linking these specific tools and formats to the rest of the MFW.

One generic example here is the use of SSH-based measurement probes. One may have a probe agent capable of creating an SSH connection to a remote host and executing scripts on the target to collect measurement information (such as reading system logs or configuration files, or executing custom probe commands). In this case, the generic output can be, for example, the output of the script execution. Thus in this case the probe is the SSH script executed on the SSH server in the target. The probe agent is a component of the MFW capable of performing these measurements over that SSH.

On the other hand, various constraints need to be considered. In many cases, such as mobile nodes, the address of the target of measurement may vary. Similarly, the availability of the connection to that node may vary. Further, having a separate server of the target of measurement available to respond to queries can be a problem due to the need to have open ports and other similar constraints and intrusiveness aspects. For this reason, a more commonly suitable approach to address these constraints is to make the connection from the target towards the probe agent. In this case, only the probe agent needs to host a server and provide a suitably static address for the collection of the data. The probe agents can then be deployed as needed and will forward the data to the server agent. The server functionality on the probe agent can be different, such as a generic SSH server or a generic Hypertext Transfer Protocol (HTTP) server. However, this type of architecture allows one to deploy any type of a service composition found useful.

D. Layer 3: Data Collection Layer

The data collection layer gathers the data provided by the base measures layer and communicates these to the control and data processing layer. It can also incorporate more advanced features such as processing of the data (e.g. multiplexing) for more efficient communication and smaller network bandwidth use similar to the MFWs described in [5, 6]. In addition to raw probe data, this layer also provides events to the control and data processing layer to describe any observed events in the MFW infrastructure. Looking at Figure 2, all the different agents take some part in the implementation of this layer although this is mainly focused on the router-agents.

In practice, a router agent in this case is a node in the peer-to-peer overlay. These agents handle the relevant parts of the dynamic adaptation, and security features as described before. This layer also handles passing data through (sub-) network boundaries where necessary, through filters such as firewalls. The router agents are basically the mediators of the communication between the server and probe agents, which connect to it by using the overlay interface. For more advanced support, it is possible to build additional agents as separate agents on top of the overlay to support features such as multiplexing of collected data, handling of authentication, authorization and encryption at the subnet level and dealing with network filters such as firewalls and Network Address Translation (NAT) services, similar to [6].

As the overlay sees the data passed through simply as something to be transferred, different strategies to address scalability at the level of data processing at different nodes can be employed without impact on the overall MFW architecture. For example, more of the processing of the base measures can be handled locally by the probe agents, resulting in less data being passed through the MFW infrastructure. This data can then be handled by a specific functionality in the server-agent. These types of approaches will result in less network traffic and lower likelihood of congestion, distributing some of the processing to the different MFW nodes. The choice of this strategy depends on the needs of the observed system and the MFW.

E. *Layer 2: Control and Data Processing Layer*

The control and data processing layer provides services for processing the data from the data collection layer and for controlling the MFW infrastructure. This is basically equivalent to the server-agent shown in Figure 2. Specific clients can then be built to access the MFW through the server-agent. In order to support scalability, extensibility and interconnection with different clients of the MFW, we use an architecture based on the blackboard architectural style [27] as illustrated in Figure 5.

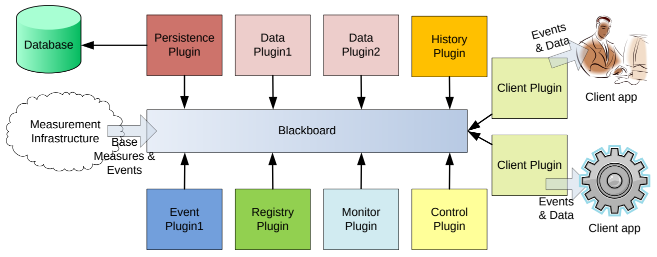

Figure 5. High-level architecture of layer 2.

This type of architecture is actually shared across all the different nodes (agents) of the MFW and not just the server-agent. However, it is described here in terms of the server-agent to provide a concrete example of its application. The goal is to achieve a highly cohesive and decoupled composition of components to support their composition in different ways in different nodes and to address different requirements such as live updates of different parts and verification of functional correctness. To achieve this, all data is passed through the blackboard, whereas messages between the plugins are passed using the OSGi middleware mechanism.

Regarding the data, all measurements, events and client commands are processed through a blackboard component, which provides this data to all registered plugins that have subscribed to this type of data. These plugins can provide additional data to the blackboard, which can further be processed by other plugins. This allows for clear separation of different aspects of data processing, event handling, client and MFW infrastructure communication and other aspects. Again, the plugins are mapped to OSGi services in our implementation. This also provides the means to do updates of specific plugins through the OSGi update mechanism. In the case of message passes where simple data values are not optimal, the bindings are handled dynamically through the OSGi service binding mechanisms. This allows us to address dynamic composition, decoupling and cohesion on different levels.

The control factor in this layer is related to mapping the measurements requested for specific properties of the observed system to the probes of the MFW infrastructure. This includes abstracting all dynamic evolution of the infrastructure(s) in an infrastructure model provided to the client(s). The control factor is also related to responding to events that require taking actions and control over the deployed MFW infrastructure.

In order to support all the requirements described in Section III, different types of functionalities need to be supported. Client-specific data processing functionality can be provided as customized plugins for the blackboard. For example, basic processing functionality can be provided in the form of a plugin that takes instructions from the MFW client that define calculations and threshold values of the monitored data. These can be used to calculate more advanced values from the base measures, and to provide events to the client when a given threshold value for the calculations is exceeded. As the communication is handled through the blackboard, additional processing of the values provided by one of these plugins can be done with another plugin that subscribes to the provided values of the previous plugin.

Customized plugin functionality can be, for example, used to provide specific data to a specific client or to perform custom control and configuration of the observed system based on the observations. Although the needs for different measurement domains may vary, we define a set of basic plugins offering generic services for different purposes. This includes a functionality to store all the data processed through the blackboard to support historical analysis actions, a control plugin to handle the adaptation of the MFW in response to the events observed in its operation (e.g. to configure probe sampling rates) and a registry for handling the abstraction of infrastructure changes to the client. Similar to [3], filters can be attached to any plugin to control the data it processes and to allow for more fine-grained configuration of plugins (e.g. what the persistence plugin stores).

F. *Measurement Abstraction*

Besides addressing the different needs for runtime adaptation in terms of adaptive communications, the MFW also needs to provide a means for the client to make requests for a specific type of measurement without the need to define explicitly which specific probe will provide it. This abstraction is the role of the control and data processing layer in the MFW (Layer 2). In our case, we have described our solution in our previous work [1] and here we shortly summarize the main points.

Each measurement probe can be identified with a Base Measure Identifier (BM ID) that is composed of a Measure ID coming from BM taxonomy, and a Device ID that identifies which infrastructure object it is measuring. Note that the term Device ID may be misleading at times as the target of measurement here can also be a service and as such hosted on or a subpart of the functionality hosted on a device.

It should be noted that several taxonomies or other approaches can be used as a basis for defining the Measure ID. The actual choice is domain dependent and can even vary inside the domain based on the specific application choice and target inside the domain. For example, in the security assurance domain, we have used as an example the security countermeasures taxonomy from [28], which classifies different measurement targets such as firewalls and intrusion detection systems inside the security assurance domain. Another example inside this same domain is the use of the categories from Common Criteria.

Using the two identifiers we have defined, all measurement results can have Unified Resource Identifiers (URI), or to be exact, Uniform Resource Locators (URL), for example, of type: `MFW://<device_id>/<measure-id`.

A client of the MFW can then make requests for the different types of measurements on different types of targets. The MFW can then provide best-effort measurement results according to the available probes and their characteristics. For more advanced support, the different probes can also be described in terms of different characteristics such as their ability to provide precise results. Such characteristics can only be defined by those who deploy the specific probes, as their interpretations will vary over different probes. However, the MFW can use by default this data as ordinal scale values and let the user define the scales.

Finally, each MFW Uniform Resource Identifier (URI) can then be used as a hyperlink between different measurements. Using this method, it is possible to make BMs dependent on other BMs where this is found useful.

G. Security

As described in Section III.E, security-enforcing mechanisms of the MFW infrastructure need to include user and data authentication, authorization, data confidentiality, data and system integrity, data and system availability, non-repudiation and resilience solutions. This is again supported by the composition of the MFW agents from different services (components). These can be used to compose the agent to support features such as encryption and authentication with specific modules such as described in [9, 10]. This also allows fulfilling the security demands on different scales of distribution. In more distributed architectures this requires decentralization of these features over subnets in the router-agents, while in a centralized version this support can also be centralized at the server-agent level. This can also be handled at the middleware level as provided by the MOM platform (e.g. through the security features of the P2P overlay).

Each MFW component basically has a single user type, which is the higher-level agent (probe → probe agent (→ router agent) → server agent → MFW client) that can issue control over it. Similarly, each one has a single type of a user that can provide monitoring information, which is the lower-level agent (opposite order to the previous one). This knowledge can be used to simplify the required authentication process as only one type of a user needs to be supported at each level, and this user type is always known.

Considering the communication between the different elements of the MFW, each probe agent that controls a probe and provides data to the server-agent needs to register with proper authentication mechanisms before being able to communicate measurement results. Similarly, each data transmission needs to be authenticated to ensure that no false data is provided by unauthorized attackers.

In order to limit the possibilities of attackers using MFW components to perform intrusive actions on a system or as an attack vector, the capabilities of probes and agents should be limited where possible. For example, user accounts can be created to access the required information for the probes, with said users only being authorized to access the required data. To ease management, a roles strategy can be used to enable multi-domain or subnets management where the configuration of the accounts can be replicated without local or specific particularities.

In the following list, we describe some of the technologies and solutions we have applied in addressing these related security requirements.

For achieving integrity of the exchanged data:

- Applying hash methods, which will verify that the received data is the same as the sent data.
- Digital signature methods to provide non-repudiation features.

For ensuring confidentiality of the exchanged data:

- Encrypted communications, communications mechanisms based on public and private key strategies.
- Similarly, applying encryption to all configuration data stored by any MFW node (probe/agent) such as usernames and passwords.

Supporting the availability of the exchanged data:

- Providing features that monitor the availability of the different systems of the infrastructure (agents and cockpit) and that will alert when a device has availability problems. This is related to the self-monitoring features of the MFW.
- The use of data filtering techniques to, for example, reduce the possibility of (DoS) attacks. Specific considerations are needed; for example, when the MFW addresses are mapped for mobility and dynamic aspects (e.g. added or removed element).
- Consequently, the communication protocol used between the probes and aggregation server can propose an alternative management solution, and if historical data is considered to be important, probes should implement the management of local history.

H. Separation of the MFW and the Observed System

As elements of the MFW (probes and probe agents) are installed to measure properties of the observed system, their deployment is bound to have some impact on the observed system. To minimize this intrusiveness, different aspects need to be considered. The first aspect is related to separating the communication of the measurement data and the effects it can have on the observed system. Here we have described the use of the P2P overlay to address this aspect.

Another aspect of isolation is related to deploying elements of the MFW on the same physical host machines as the one observed in the system. In many cases the functionality of the MFW (the agents) needs to be hosted on the observed infrastructure elements. The separation of these two can be addressed by using virtualized infrastructure to host the components of the measurement framework (e.g. Java Virtual Machine (JVM) for OSGi or a complete separate VM).

Finally, intrusiveness can be mitigated with self-monitoring and adaptation. In this case, specific self-monitoring plugins would be deployed at the different nodes (agents). These would monitor for specific problems such as depletion of resources (e.g. CPU, memory, network bandwidth) and adjust their operation accordingly. The specific strategies would require domain-specific tuning according to the criticality of different factors such as the availability of the measurement data and the operational intrusiveness of specific nodes.

I. *Patterns and the Reference Architecture*

This subsection provides a brief mapping of the MFW patterns listed in Section II. The requirements of the architecture are discussed in more detail in the following section. However, these requirements are also referred to in this section when relevant for the explanation of the patterns. In the following we recall the patterns from Section II and briefly note how they are visible in the proposed architecture.

- Specific adaptation layer for measurement targets [5]. This is the role of layer 3 (probe agents) together with layer 4 (probes).
- Repository of available probes for specific measurement targets [2]. As discussed we identify a set of common mechanisms such as SSH and HTTP probes. In different domains it is further possible to provide specific repositories such as XCCDF in the security assurance domain.
- Simple interface to integrate custom probes into the overall measurement system [2, 3]. This is provided by the split of the probe agent to generic (readily provided) and probe-specific (custom) parts.
- Usage of widely supported protocols [7]. Relying on available and widely developed MOM solutions addresses this.
- Allow configuration of operational probes [7]. This is supported by the interfaces of a probe agent, which can adapt to the probes as best possible.
- Optimize communications related to expected measurement data communication patterns [5, 7]. In our case the assumption is that bandwidth requirements are relatively reasonable, and thus no specific data channels are used. In other cases, this could be supported with the addition of another path.
- Usage of separated dedicated communication channels [7]. This is basically the definition of the P2P overlay we use.
- Allow customizing different aspects of the MFW based on module composition [3]. This is supported by the OSGi platform together with the blackboard-based architecture.
- Provide a registry that is dynamically updated to reflect available measurements and probes [6]. This is supported by layer 2 (server-agent).
- Use specific components to handle connection and processing over network sub-domains [5, 6]. This is transparent to the MFW agents thanks to the P2P overlay.
- Describe the measurements with a common set of properties [4, 6]. This is supported by our measurement abstraction layer.
- Optimize the availability and reliability of measurement infrastructure [13]. Again this is supported by the P2P overlay, which is specifically designed to support these properties, cleanly modularized from the MFW perspective.
- Secure the communication data [13]. The Armature P2P overlay we used handles a large part of the features related to security. Additionally, we provided the guidelines for the security mechanisms we have applied to address this.

V. DISCUSSION

While the mapping of the MFW design patterns was discussed before, this section discusses in more detail how the reference architecture (RA) addresses the different requirements described in Section III. This discussion is structured along the categories of requirements presented in Section III.

The RA supports different scales of distribution by supporting different communication protocols and strategies in the communication layer. The described P2P approach especially allows for a distributed approach. In general, new MFW agent deployment is supported by runtime registration mechanisms. Further, the common plugin architecture for all MFW agents, based on the OSGi component platform, allows for distributing functionality to agents as needed. For example, the use of a common component platform and plugin architecture allows for the deployment of some of the server-agent functionality on probe agent nodes if needed for local processing. This and the ability of server-agents to use any number of different components for data processing support scaling to varying amounts of data. Some specific issues to consider with regards to scalability include the ability to use advanced features such as discovery and registration mechanisms as a means to launch attacks on the MFW itself, and the intrusiveness of monitoring the observed system resource use to launch any measures for dynamic adaptation, such as rerouting.

In the case of many of the runtime adaptation requirements we rely on the work done to address these requirements in the middleware community. The P2P overlay discussed as the communication channel for the RA supports most of these requirements.

The RA uses several approaches for isolating the MFW from the observed system. This includes both the discussed features for the isolation of the network communication and of the SW components of the MFW. The effectiveness of this approach depends on the use of available techniques. For example, using a JVM to host the agents allows some control over the resources it can use (e.g. memory) but can be limited in other regards (e.g. CPU, files). This is a trade-off to consider for different scenarios and may require use of additional advanced techniques (e.g. sandboxing, such as [29]).

As discussed before, addressing probe effects with deployment of various probes in different systems is difficult in general. Testing everything fully in a separate environment would be ideal, but it can be very expensive and difficult to simulate all the possible combinations of different probes and their environments during a system lifetime. This also applies to the combination of (sub-) systems with other systems. No generic guidelines can be provided for possible probe effects other than checking the probes and agents in operation as far as possible, and considering their properties such as files accessed and resources needed. This also highlights the importance of the isolation aspect.

When using specifically created probes that have been designed into the system from the beginning, intrusiveness is not an issue. Unfortunately, this is not always the case even when designed for, as the future probe requirements are impossible to fully predict. However, the RA aims to deploy only minimal components to the observed system infrastructure, which helps in minimizing its intrusiveness by focusing complex processing in the server-agent. It also focuses on minimizing the intrusiveness in terms of the router-agents by requiring only the deployment of agents for the P2P overlay that share similar deployment requirements.

The basic verification of the correctness of the MFW and its components requires use of testing and verification techniques before deployment. In this part we have to rely on the availability and use of advanced techniques for SW testing and verification. However, we do support this with the interfaces designed for the agents in the form of state transfer (for updates) and configuration access (for reconfiguration). These interfaces form a basis for testability features that can be used to test agent behaviour in various contexts.

Even if we cannot ensure full pre-runtime verification with architectural solutions, the RA addresses runtime verification of different aspects with its self-monitoring features similar to the intrusiveness aspects. This cannot address all possible scenarios but allows for building features to check the correctness of new agents and their functionality in new contexts. This is also supported by the provided automated update mechanisms, which allow for addressing found issues and updating the agents with new features as needed.

The different aspects of security are addressed at the different agents. Specific components are provided to be deployed as services with the MFW agents as needed. These allow for addressing the different requirements related to security such as confidentiality and integrity also at different degrees of distribution. The dependability aspects (including availability and resilience) of security are addressed by the isolation of the MFW infrastructure (similar to the intrusiveness aspects) from the observed system infrastructure as far as possible. This also includes the use of all the security- and scalability-related solutions for handling large amounts of data and unauthorized usage attempts in case of failures in the observed system or malicious usage attempts of the MFW or the observed system.

As for overall security, the MFW is not different from other distributed systems handling sensitive information. We have provided some guidelines regarding the securing of different aspects of the MFW. However, these are generic and applied only to the specific domain of the MFW. More generally, most other approaches for system security in general also apply here and are thus not discussed in detail.

Overall it can be said that different domains set different requirements and in this case some of the issues are more important to address than others. For example, in systems with high performance and large monitoring data streams, it can be useful to optimize monitoring with separate channels as in [6] and optimized data formats as in [2]. The extensibility and customization options provided by the modular reference architecture should provide a good basis for this.

VI. Conclusions and Future Works

This paper presented a core set of requirements for building a secure, dependable and adaptive distributed monitoring framework and reference architecture for addressing these requirements. This is based on both surveying existing approaches to building monitoring frameworks for different domains, and on our current work and experiences in building monitoring frameworks for different domains. The work provides a basis for building other monitoring frameworks that need to address these types of requirements. The presented requirements provide a basis for understanding the different needs of monitoring and how they are related to the domains in which the reader is interested. The reference architecture shows how these can be addressed given the common constraints we present, showing both high-level architectural solutions and practical examples of their implementation. These are topics that are increasingly relevant in many aspects of modern systems, where different runtime adaptation aspects, information collection for decision support, and other aspects need to be supported.

Future work entails describing further experiences of the different implementations and their practical applications in different domains. Future works should also include more systematic consideration of the impacts of monitoring data and analysis on the initial risk analysis that provides input for monitoring as well as their iterative refinements. Events at different levels may also require more attention. Specific future aspects to consider include the emerging new types of infrastructures such as the future internet and cloud-based services where monitoring needs to consider specific challenges posed by the shared infrastructure between different stakeholders, including both infrastructure providers and consumers.

Acknowledgment

The work presented in this paper has been carried out in the CELTIC BUGYO Beyond research project. The authors acknowledge the contributions to the topics discussed in this paper by various project partners.

References

[1] T. Kanstrén and R. Savola, "Definition of Core Requirements and a Reference Architecture for a Dependable, Secure and Adaptive Distributed Monitoring Framework," in *3rd Int'l. Conference on Dependability (DEPEND 2010)*, 2010.

[2] M. Pollari and T. Kanstrén, "A Probe Framework for Monitoring Embedded Real-Time Systems," in *Proc. 4th Int'l. Conf. on Internet Monitoring and Protection (ICIMP 2009)*, Venice/Mestre, Italy, 2009, pp. 109-115.

[3] F. Fusco, F. Huici, L. Deri, and S. Niccolini, "Enabling High-Speed and Extensible Real-Time Communications Monitoring," in *Int'l.*

Symposium on Integrated Network Management, 2009, pp. 343-350.

[4] Q. Wang, J. Shao, F. Deng, Y. Liu, M. Li, J. Han, and H. Mei, "An Online Monitoring Approach for Web Service Requirements," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 4, pp. 338-351, October-December 2009.

[5] E. Bulut, D. Khadraoui, and B. Marquet, "Multi-Agent Based Security Assurance Monitoring System for Telecommunication Infrastructures," in *Proc. Communication, Network and Information Security*, 2007.

[6] I. Legrand et al., "Monitoring and Control of Large Systems with MonALISA," *Communications of the ACM*, vol. 52, no. 9, pp. 49-55, September 2009.

[7] W. Vandelli et al., "Strategies and Tools for ATLAS Online Monitoring," *IEEE Transactions on Nuclear Science*, vol. 54, no. 3, pp. 609-615, June 2007.

[8] T. Kanstrén, "A Study on Design for Testability in Component-Based Embedded Software," in *6th Int'l. Conf. on Software Engineering Research, Management and Applications (SERA 2008)*, Prague, Czech Republic, 2008, pp. 31-38.

[9] Common Object Request Broker Architecture (CORBA) Specification, Version 3.1, 2008

[10] Sun Microsystems, "Jini Connection Technology", 1999.

[11] SIP: Session Initiation Protocol, Internet Engineering Task Force, RFC 3261, 2002.

[12] RTP: A Transport Protocol for Real-Time Applications, Internet Engineering Task Force, RFC 3550, 2003.

[13] R. M. Savola and H. Abie, "Development of Measurable Security for a Distributed Messaging System," *International Journal on Advances in Security*, vol. 2, no. 4, pp. 358-380, 2009.

[14] R. M. Savola and P. Heinonen, "Security-Measurability Enhancing Mechanisms for a Distributed Adaptive Security Monitoring System," in *Proc. 4th Int'l. Conf. on Emerging Security Information, Systems and Technologies (SECURWARE2010)*, Venice/Mestre, Italy, 2010.

[15] sFlow.org – Making the Network Visible. sFlow.org [Accessed: May 21, 2011].

[16] Cisco Systems NetFlow Services Export Version, Internet Engineering Task Force RFC 3954, 2004.

[17] N. Ziring, S.D. Quinn: Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.1.2, U.S. National Institute of Standards and Technology, *NIST Interagency Report 7275* (Draft), 2006.

[18] R. V. Binder, "Design for Testability in Object-Oriented Systems," *Communications of the ACM*, vol. 37, no. 9, pp. 87-101, September 1994.

[19] N. Feng, T. White, and B. Pagurek, "Dynamic evolution of network management software by software hot-swapping," in *Int'l. Symposium on Integrated Network Management (IM2001)*, 2001, pp. 63-76.

[20] D. B. Parker, *Computer Security Management*. Reston, VA, USA: Reston Publishing Company, 1981.

[21] OSGi Alliance: OSGi – The Dynamic Module System for Java. www.osgi.org/Main/HomePage [Accessed May 21, 2011]

[22] Q. Wang, J. Shen, X. Wang, and H. Mei, "A component-based approach to online software evolution," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 18, pp. 181-205, 2006.

[23] A. Hecker and M. Riguidel, "Survivability as a Complementary Operational Security Model for IT Services (position paper)," in *PERADA Workshop*, 2008.

[24] XML-RPC Specification. www.xmlrpc.com [Accessed May 21, 2011].

[25] R.T. Fielding, R.N. Taylor, "Principled Design of the Modern Web Architecture", *ACM Transactions on Internet Technology,* 2(2): 115-150, 2002.

[26] M. Henning, "A New Approach to Object-Oriented Middleware", *IEEE Internet Computing, vol. 8, no. 1, 2004.*

[27] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerland, and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns*: John Wiley & Sons, Inc., 1996.

[28] A. Herzog, N. Shahmehri, and C. Duma, "An Ontology of Information Security," *International Journal of Information Security and Privacy*, vol. 1, no. 4, pp. 1-23, October-December 2007.

[29] Y. Bennet et al., "Native Client: A Sandbox for Portable, Untrusted x86 Native Code," *Communications of the ACM*, vol. 53, no. 1, pp. 91-99, 2010.