2011 Ninth Annual International Conference on Privacy, Security and Trust

Towards machine-assisted formal procedures for the collection of digital evidence

Martín Barrère, Gustavo Betarte and Marcelo Rodríguez
Instituto de Computación, Facultad de Ingeniería, Universidad de la República
J. Herrera y Reissig 565, 5to.piso, CP 11300, Montevideo, Uruguay
Email: {mbarrere,gustun,marcelor}@fing.edu.uy

***Abstract*—The nature of computer crimes has systematically evolved with the progress of computer technologies. Due to the complexity of forensic investigations, the design of new techniques and tools for speeding up and automating tasks required by digital forensic processes has become a challenging task. In particular, the collection of (live) digital evidence is a delicate work that requires special care and proved investigator skills. This work presents a framework for the specification of collection procedures based on an extension of the OVAL language and describes a tool that has been implemented to automate the execution of those procedures.**

I. Introduction and background

Computer crimes occur on a daily basis and almost every technology used with some valuable purpose has been, or is a candidate to be, the object of some kind of attack. In most cases, successful computer attacks are not matter of luck or just coincidence; attackers dedicate time to their planning and deployment. However, in the presence of a security incident, like an attack against a corporative network, system administrators do not have neither time nor calm to analyze the incident, they have to act quickly, securing the area and returning systems and services to their normal state. Within these scenarios, forensic investigations must be performed in order to understand how an attack took place on a system. The more detailed and precise the results obtained, the more accurate shall be the strategy to seal security breaches.

In [8] it was proposed a classification and characterization of the general procedures embodied by digital forensic investigations. The model provides practitioners a better understanding of the procedures involved as well as the candidate methods and techniques. In this context, evidence collection is a crucial and delicate task that requires wide experience from who carries out the activity. Security tools such as *Sleuth Kit & Autopsy* have been integrated into powerful forensic distributions ([2], [1]) with the purpose of softening investigators activities and speed up data collection processes. However, the development of a forensic investigation using these tools still requires a high degree of involvement from and interaction with the expert who is conducting the investigation.

For years organizations have developed structured procedures and protocols to face security incidents. Nevertheless, most of the required tasks are typically human-dependent, they are not given a precise specification and are usually very time-consuming. Digital investigators must be aware of several technical aspects and comply to best practices when performing digital evidence collection, such as, for instance, preserving the integrity of the evidence and clearly defining the order in which the evidence should be collected. As stated in [13], proper *standard operating procedures (SOPs)* are essential for digital forensic practitioners to perform investigations that ensure the validity, legitimacy and reliability of digital evidence. Furthermore, such procedures should be flexible enough to evolve with digital forensic environments dynamics. In addition to expert management of appropriate tools, deep knowledge and fast reaction are also required skills for the individuals in charge of this task. Automation mechanisms are deeply needed. They should provide non-ambiguous means for specifying standard forensic procedures thus fostering forensic knowledge exchange, and at the same time, automatic machine-based tools capable of interpreting and executing the specified directives in a reproducible manner. In this paper we put forward a framework for the definition and execution of forensic procedures which meets these requirements. In particular this framework can be used for the collection of live digital evidence and is based on three fundamental blocks.

The first one concerns the formal and machine-readable specification of forensic procedures. This has been addressed by developing a mechanism, called XOval, to define extensions of the language OVAL (Open Vulnerability and Assessment Language) [6]. OVAL's main target are system vulnerability descriptions. In [12] it is used for gathering computer systems properties and characteristics for further analysis, whereas in [11] it is extended to describe the capabilities gained by an attacker when a certain vulnerability is exploited. XOval allows us, for instance, to specify data to be collected from a target system that could not be expressed in terms of pre-defined OVAL objects. We also exploit the ability of the language to express, in a declarative way, what has to be done to collect the desired data. The second block of the framework concerns the automation of collection of digital objects. We have designed and implemented a tool, called XOvaldi, which is an OVAL-based interpreter which can be easily extended to run forensic primitives defined using XOval. The third block is a first, and very initial step, concerning the process of identifying and precisely specifying the forensic primitives required for investigating a particular scenario. We have experimented, using XOval and XOvaldi, with the methodological approach described in [10] to conduct a computer forensic investigation of a system. In this paper we just provide some insights concerning the way the formal procedures described

978-1-4577-0584-7/11/$26.00©2011 IEEE

in that work can be given an automated treatment using our framework. Space constraints prevent us from including a more detailed discussion of the results obtained from the experiments that have been conducted. By the time the work presented here was under development, a similar approach for automating live forensic procedures, called the XLIVE framework, was reported in [9]. That framework, however, has been designed to work only on Windows-based desktop PCs and relies on the use of a non standard XML-based language to specify data gathering.

The rest of this paper is organized as follows. Section II describes a methodological proposal for the specification and automated collection of (live) digital evidence. In Section III is presented XOvaldi, a live forensic multiplatform and extensible OVAL-based tool for digital evidence collection. Section IV describes further work and concludes.

II. A PROPOSAL FOR AUTOMATED LIVE DIGITAL EVIDENCE COLLECTION

In what follows we motivate the use of the OVAL language as the formal setting for specifying formal forensic procedures, and put forward an OVAL-based framework for the automation of digital data gathering.

A. OVAL overview

The OVAL language, developed by MITRE Corporation [5], is an information security community effort to standardize how to assess and report upon the machine state of computer systems. OVAL is an XML-based language that provides means for uniformly describing vulnerabilities, analyzing and detecting them, and exchanging security related information. It covers a wide range of computer platforms and provides support for assessing several types of system components such as, for instance, running processes, open network connections and files with a specific content. While OVAL is a specification language allowing to express specific machine states like vulnerabilities or specific configuration settings, real analysis is performed by OVAL interpreters.

The OVAL language provides a framework to specify and conduct the three main steps of a system assessment process: (1) precise description of the configuration information of the system to be inspected; (2) analysis of the target for detection of a particular state of the system, like a vulnerability, specific configuration or patch state; (3) reporting of the assessment results. From a digital collection perspective, one might apply the first step of the assessment to specify the data that must be collected, the second step to actually perform the data gathering, and the third step to present the collected evidence.

When an OVAL process is performed, a report is generated that includes a detailed list of the gathered system components and their characteristics required to perform the analysis stage. If one think of such report as a document containing each collected object required to be analyzed within a forensic investigation, the initial OVAL document may be grasped as a precise specification of the procedure the investigator must follow during the evidence collection stage. Under this perspective, we propose to use OVAL definitions for expressing digital evidence to be collected within a forensic investigation and to use the OVAL report mechanism for presenting the collected digital evidence. In this way, OVAL documents constitute a standard means for expressing forensic knowledge. Because OVAL is an XML-based language, it inherits all XML features such as platform independence, interoperability, transportability and readability. Another interesting feature of OVAL is that the language provides a declarative way to express what have to be done during a collection process. Functional issues such as how the process steps are actually interpreted are clearly separated and they are considered as an implementation problem as described in Section III.

B. XOval, an extension of the OVAL language

As technology and forensic techniques evolve, new types of evidence arise as important pieces of digital investigations. Despite the wide spectrum of digital components which are pre-defined in the OVAL language, it does naturally not cover every type of data that might be of interest in a forensic scenario. While experimenting with the system we rapidly got into cases where data to be collected to perform a particular analysis, like logged users on a system, could not be expressed using the primitive constructions of the language. Therefore we developed a mechanism to define extensions of OVAL, which we call XOval (eXtended Oval). With the definition of this mechanism we aimed at providing a uniform and simple way to incrementally leverage the expressive power of the language, thus allowing to describe new digital components. The OVAL language presents a hierarchical structure based on XSD files (XML Schemas) where each schema inherits common attributes from the core schema. The OVAL tests are located on different schemas depending on the software they describe, typically associated to the platform they belong. In order to extend the language to support further data gathering in a platform, we propose to create new schemas which specify the characteristics of the data to be collected that is not currently supported by the OVAL language. The separation between the original schemas and the new ones, allows us to perform an upgrade of the whole OVAL specification without requiring a rewriting of existing extensions. If a class of digital evidence is required by a digital investigation but it is not supported by OVAL, the language can be monotonically extended so as to cover one such type of system data source.

C. Automating evidence collection with XOval-based procedures

XOval has been developed as a conceptual, and formal, mechanism to leverage OVAL definitions so as to fit new forensic requirements. Our proposal for automating formally specified evidence collection involves, in addition, two further requirements: (1) a methodological guide to identify and precisely specify the forensic primitives required for investigating a particular scenario; (2) an automated tool capable of interpreting and running over the target system those forensic primitives. Concerning the second requirement, the solution has been to design and implement a tool, called XOvaldi, which is presented in Section III.

As to the first requirement, we have experimented with the methodological approach described in [10]. To conduct a computer forensic investigation of a system in response to the suspicion or actual occurrence of an attack on that system, the author proposes a formal model for analyzing and constructing forensic procedures. In that model, an attack a_i is specified as the set $C_i = \{c_h, ..., c_k\}, C_i \subseteq C$, of affected components by the attack, where $C = \{c_1, ..., c_n\}$ denotes the set of all components on all operating systems. In order to detect an attack a_i, the model defines a forensic procedure $F_i = \{f_h, ..., f_k\}, F_i \subseteq F$, where each entry denotes a forensic primitive f_j that focuses on the investigation of exactly one component; F denotes the set of all forensic primitives.

The main objective of our methodological experiment was to explore the feasibility of defining formal specifications of collection procedures exploiting the conceptual constructions defined in Leigland's model and providing an interpretation of those formal structures in terms of a standard security language. The approach we have followed is to identify and give a precise characterization of the conceptual mapping that relates the abstract constructs defined in Leigland's model with OVAL main building blocks. This is illustrated in Figure 1.

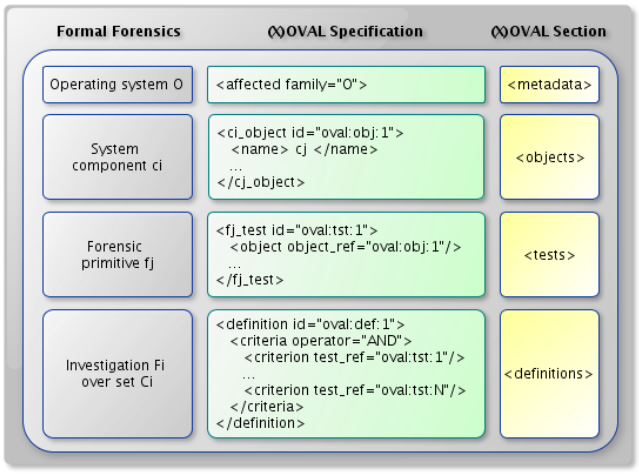

Fig. 1: Specifying forensic primitives using (X)OVAL

Thus, if a forensic investigation must be carried out on an operating system O, an (X)OVAL document should be produced that precisely describes the digital components required by the forensic process. The metadata section identifies the target (operating) system of the investigation. Each affected system component c_i required to be analyzed can be specified using (X)OVAL objects. Such (X)OVAL constructs permit to describe the characteristics of the digital components that must be collected. The second row in the figure shows the specification of an hypothetical component c_i where the attribute *name* is available to be specified. As stated in Leigland's model, each forensic primitive is connected with only one digital component. A forensic primitive then is represented by an (X)OVAL Test which makes reference to an (X)OVAL Object, the one it is intended to inspect. In the third row it is introduced the representation of a forensic primitive f_j as an (X)OVAL Test with id *"oval:tst:1"*, and shows how the test is connected to the (X)OVAL Object *"oval:obj:1"* that it must assess. A forensic procedure then is formalized as an (X)OVAL Definition which is formulated as a logical combination of a set of forensic primitives. The fourth row in the figure shows how a forensic procedure F_i is described as an (X)OVAL Definition with id *"oval:def:1"*, assuming that N (X)OVAL tests must be performed. It makes it clear what system components must be collected for further analysis.

Under this perspective, the problem of defining and automating the execution of a collection procedure reduces (1) to provide an XOval Test of each forensic primitive, as just explained, and (2) to associate each primitive to a piece of software capable of executing the actions needed to investigate the specified component. Concerning the implementation of the forensic primitives, the architecture of the XOvaldi tool, which is discussed in the next section, has been designed so as each one of these software pieces can be implemented as a single *plugin* that can be fetched and invoked by the tool out of plugin repositories.

III. An XOval interpreter

XOvaldi is a multiplatform and extensible OVAL-based tool for (live) digital evidence collection[1]. XOvaldi has been designed and implemented as a conservative extension of Ovaldi [7], which is a free OVAL interpreter maintained by MITRE [5] as a reference implementation for evaluating OVAL definitions. The current release of Ovaldi covers a wide, but not complete, part of OVAL's specification. The design of XOvaldi, which has been conceived as an extension of Ovaldi, is motivated by the need to count with an OVAL-based interpreter which could be easily extended so as to be able to provide an implementation of the forensic primitives defined using XOval.

A. Forensic primitives in XOvaldi

The methodological approach for the definition and execution of forensic primitives proposed in section II identifies a set of requirements, mainly extensions to the core components of the framework, that can be achieved following a sequence of restructuring steps. In what follows we present and explain solutions to those requirements which are based in the use of the functionalities provided by XOvaldi.

- **XOval extension.** The XOval language specification is based on XML Schema files; new digital objects to collect have to be specified within an appropriate existing XML schema –from a platform viewpoint– or in a new XML schema representing a new family of components.
- **XOvaldi model regeneration.** In order to support language extensions, XOvaldi's logical data model has to be regenerated to fit the language constructions. Our implementation relies on the JAXB technology [4], a Java architecture for XML binding, that provides automatic mechanisms for generating Java classes from XML

[1]Readers interested in experimenting with the tool please contact the authors

Schema specifications. Therefore, XOval model regeneration is automatically done. Moreover, JAXB provides automatic means for parsing and writing XML documents enabling seamless code upgrade over underlying language extensions with almost no cost.

- **Plugins codification.** Once the interpreter is able to manage the new types of digital components, functional extensions are required to perform the evidence collection. XOvaldi's architecture easily allows to extend functionality of the interpreter by simply adding new plugins on the plugins repository. XOvaldi loads on demand the required plugins in order to perform the newly defined procedures.

As we stated in II-C, each plugin is connected to one specific forensic primitive. Moreover, XOvaldi implementation follows the OVAL philosophy concerning the way the language is organized. XOvaldi plugins repository is composed by families of plugins according to the platform they belong.

B. Characteristics of XOvaldi

We highlight some of its relevant characteristics:

- **Extensibility.** Forensic scenarios are dynamic and they are always changing, so there is a need for adaptable tools capable of evolving with environment changes. XOvaldi meets this requirement by design, enabling functional extensibility through a plugin-based architecture.
- **Portability.** XOvaldi's core has been purely developed using Java-based technologies [3]. An XOvaldi release contains every needed dependency, including its own Java Runtime Environment. Then, it does not relies on any software component within the target device. This allows XOvaldi to be run from any removable media, including read-only media, making it appropriate to be used as a live-response oriented tool.
- **Distributed.** XOvaldi can be used in two different modes: standalone mode or client-server mode. The former is the standard or local mode, where requested activities are performed on the local context and their results are also stored within an accessible local path. When client-server or distributed mode is used, two XOvaldi running instances can communicate each other in order to perform remote evidence collection or remote evidence storage. Based on a client-server model, the approach works on a request-response basis.

IV. Conclusions and future work

In this work we have presented a framework for collecting digital forensic evidence, a key stage in a forensic investigation, based on the use of OVAL, a standard security content XML-based language. It has also been explained the mechanism that makes it possible to extend OVAL definitions to define new forensic primitives and it has been presented a multi-platform and extensible tool (XOvaldi) that provides support for performing automated evidence collection based on extended (X)OVAL specifications. The framework incorporates mechanisms that (1) promote tasks modularization within forensic investigations, and that (2) make it possible to elaborate a specification of a forensic procedure independently of the technology used to its implementation. This enables the integration of knowledge provided by experts from different areas within the information technology field. Such knowledge, represented as a combination of forensic procedures, might be exploited when performing digital evidence collection. Furthermore, the one in charge of executing these procedures does not strictly need to know every detail about how the collection is carried out. The framework also provides support for defining atomic forensic primitives that can be later combined for building forensic procedures. Each forensic primitive in turn can be reused within multiple procedures. XOvaldi's architecture has been designed to support that approach by providing a mechanism for implementing a forensic primitive as a plugin of the tool. The reutilization of an implemented forensic procedure consists basically in invoking that plugin out of an XOvaldi repository.

Forensic investigations comprise several stages that in turn involve multiple tasks. As further work, we plan to analyze to which extent our proposal for the specification and automation of collection procedures might also apply to other tasks of a forensic investigation process. This analysis shall be carried out from a technical perspective, considering, for instance, forensic stages linkage and from a legal perspective, considering issues related to evidence integrity, reliability and traceability. It is also of our interest to explore alternative solutions concerning the design of a secure centralized infrastructure for performing forensic activities, with a special focus on digital evidence collection.

References

[1] BackTrack, a Linux-based penetration testing toolkit. http://www.backtrack-linux.org/. Last visited on May 28, 2011.

[2] Helix 3, an incident response tool. http://www.e-fense.com/. Last visited on May 28, 2011.

[3] Java technology. http://www.oracle.com/us/technologies/java/. Last visited on May 28, 2011.

[4] JAXB, Java Architecture for XML Binding. http://www.oracle.com/technetwork/articles/javase/index-140168.html. Last visited on May 28, 2011.

[5] MITRE Corporation. http://www.mitre.org/. Last visited on May 28, 2011.

[6] OVAL Language. http://oval.mitre.org/. Last visited on May 28, 2011.

[7] Ovaldi, the OVAL Interpreter reference implementation. http://oval.mitre.org/language/interpreter.html. Last visited on May 28, 2011.

[8] A road map for digital forensic research. http://www.dfrws.org/2001/dfrws-rm-final.pdf, August 2001. Last visited on May 28, 2011.

[9] Seokhee Lee, Antonio Savoldi, Kyoung Soo Lim, Jong Hyuk Park, and Sangjin Lee. A proposal for automating investigations in live forensics. *Computer Standards & Interfaces*, 32(5-6):246 – 255, 2010. Information and communications security, privacy and trust: Standards and Regulations.

[10] Ryan Leigland and Axel Krings. A Formalization of Digital Forensics. *International Journal of Digital Evidence*, 3(2), 2004.

[11] Paolo Maggi, Davide Pozza, and Riccardo Sisto. Vulnerability Modelling for the Analysis of Network Attacks. *2008 Third International Conference on Dependability of Computer Systems DepCoS-RELCOMEX*, pages 15–22, June 2008.

[12] Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. Mulval: a logic-based network security analyzer. In *Proceedings of the 14th conference on USENIX Security Symposium - Volume 14*, 2005.

[13] Jill Slay, Yi-chi Lin, Benjamin Turnbull, Jason Beckett, and Paul Lin. Chapter 3 Towards a Formalization of Digital Forensics. *Ifip International Federation For Information Processing*, pages 37–47, 2009.