Haruspex—Simulation-driven Risk Analysis for Complex Systems

Fabrizio Baiardi is a professor in the Department of Computer Science at the University of Pisa, Italy.

Claudio Telmon, CISA, CISSP, is a freelance consultant in ICT security and risk management. He also cooperates with the University of Pisa's Department of Computer Science on the same topics. He is a member of the ISACA Milan Chapter.

Daniele Sgandurra, Ph.D., is a postdoctorate researcher at the Institute of Informatics and Telematics, National Research Council (CNR), Pisa, Italy.

Haruspex[1] is a risk evaluation methodology defined and implemented by the research group on risk management in the Department of Computer Science at the University of Pisa, Italy. It may be adopted in various risk assessment and management frameworks to evaluate the probability that an intelligent threat agent could successfully implement a multistep attack. The framework should be paired with others to discover these threats, the vulnerabilities they can exploit and the assets to be protected.

THE PROBLEM

A well-known problem with risk evaluation in IT security is that the estimation of the probabilistic component of risk is currently very difficult and highly subjective.[2]

When dealing with IT security risk, intelligent threats (or intelligent threat agents) trying to violate the security policies of an organization are usually considered. Each agent has some goals to achieve (e.g., some system components to control) and aims to minimize the effort to achieve these goals. The risk posed by each threat agent is a monotone, increasing function of both the impact of his/her attacks and the probability that these attacks are successfully implemented. In other words, in general one can assume that the risk posed by a threat increases with the impact of an attack implemented by the threat and/or the probability that the threat can successfully implement the attack. The Haruspex methodology is not focused on a detailed definition of risk; instead, it is a methodological framework with supporting tools intended to evaluate the probability that an intelligent threat can select and implement an attack against a system that results in an impact, e.g., a loss, for the owner of the systems. Haruspex computes this probability as a function of elementary factors that can be more easily evaluated in an assessment. The analyst can use the probability returned by Haruspex to compute the resulting risk, according to the adopted definition, and to evaluate and select effective and cost-effective countermeasures to reduce this risk.

The evaluation of the impact of an attack may be difficult (consider, for example, the reputational damage), but it remains a common practice for organizations, and proper methodologies are available.[3] A rather more complex problem is the evaluation of the success probability of an attack by an intelligent threat agent. The agent implements complex attacks,[4] each requiring several steps. Each step corresponds to a simple, or elementary, attack against a single system component; by composing all the simple attacks, the agent reaches its final goal. Usually, an organization relies on the expert performing the assessment to evaluate the probability that the threat agent successfully implements a complex attack. However capable this expert may be, this evaluation will be subjective and disputable, because the external and internal factors that can affect the success probability are too many and too difficult to evaluate in the case of IT systems.

The problem of interest cannot be solved in terms of experimental data. First of all, given how fast things change in IT, even large sets of experimental data on successful or attempted attacks are of little use and often refer to a too small and heterogeneous set of systems. Furthermore, experimental data for complex attacks are usually not available because organizations are not willing to share them, as this may result in leaking information on internal processes.

HARUSPEX METHODOLOGY

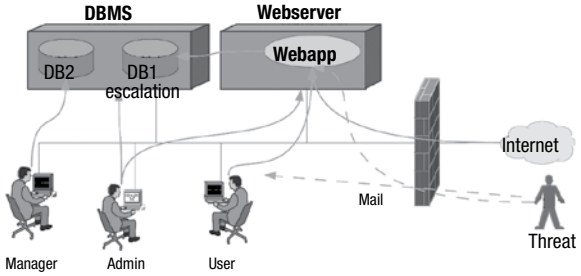
Haruspex deals with the aforementioned problem by applying a divide-and-conquer approach that decomposes the probability of interest in its components and deals with each of them separately. While this decomposition simplifies the evaluation of the factors that influence the success probability of attacks, it introduces a problem: how to define and implement the

complex procedure to evaluate the actual risk in terms of the large number of factors resulting from the decomposition. Haruspex deals with this problem by implementing a simulation of threat agents and the attacks they implement and by collecting the relevant statistical data from the simulations. In this perspective, the system is modeled as a set of components interacting through channels.[5]

These channels are also those the threat agents exploit when attacking one component from another, provided that they have gathered the required privileges. The level of detail in the representation of the system components can be easily adapted to the analysis requirements, reducing the time and effort to collect useful data. The proposed model can also represent users as further components that can be attacked through such methods as spear phishing.[6] After successfully attacking a user, the user rights can be exploited to implement further attacks.

Figure 1 further describes the system model of Haruspex. In this diagram, solid arrows represent legitimate interactions between components. Dotted arrows represent the initial (legitimate) access rights for the threat agent. In the example, the threat agent can access the public web application and send email messages to the system users. Let us assume that the goal for the threat agent is read access to DB2, the name of an internal database. In **figure 2**, attack paths through the system are represented with solid arrows. (For the sake of simplicity, just a few paths are shown.) The combination of many solid arrows represents the complex attacks that may lead the threat agent to DB2, which is the agent's goal.

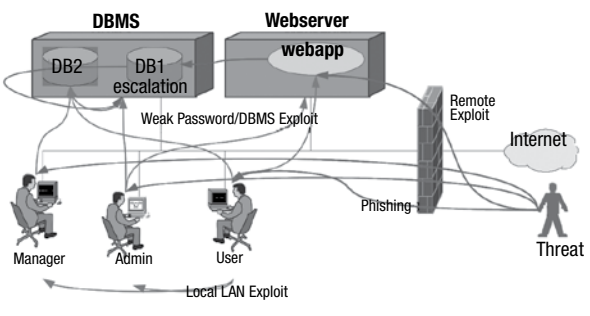
Figure 1—Legitimate Interactions in the Haruspex Model



The threat agent can acquire new access rights to components in two ways:

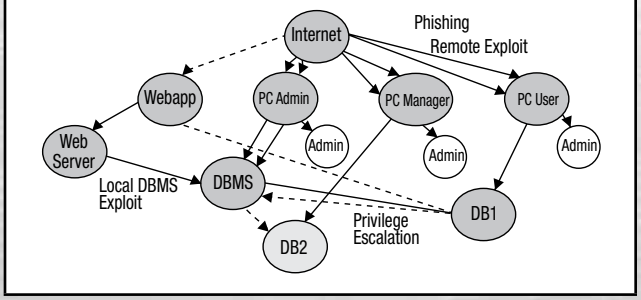
- **By attacking a component**—This requires the discovery of a vulnerability in the component and enough resources, knowledge and time for the threat agent to take advantage of the vulnerability.[7]
- **By deploying the access rights of an already compromised component (e.g., a user or an application)**—For example, after successfully compromising an administrative component through a spear-phishing attack, a threat can then deploy administrative access rights to the database management system (DBMS).

Figure 2—Attack Paths Against a System



In a complex attack, a threat exploits a vulnerability in the web application and, from there, "legitimately" accesses DB1. From DB1, the threat can exploit a vulnerability in the DBMS for a privilege escalation, gaining "legitimate" access to DB2. A second path, presumably easier to deploy, involves attacking a manager component with spear phishing, and then "legitimately" accessing DB2.

Figure 3—The Attack Graph With the Attack Paths Against the System From Figure 1



All these attack paths can be represented in an attack graph,[8] shown in **figure 3**, where the dotted arrows represent the first of the two complex attacks previously outlined. For the sake of clarity, nodes are represented as components, whereas in the Haruspex methodology they represent sets of privileges. For a formal discussion of both the model and the methodology, please refer to the publication "A Simulation-driven Approach for Assessing Risks of Complex Systems."[9]

While an attack graph can represent all possible complex attacks a threat can implement, it does not allow one to determine the complexity of each attack, the probability that all the required vulnerabilities are actually present in the system or how easily the threat can exploit them. In short, an attack graph does not support the evaluation of whether the agent can successfully implement the attack described by each path. Therefore, the Haruspex methodology introduces a probabilistic component in the overall description: Each vulnerability has a probability of existing and being discovered (by the threat) in a given time frame, and each attack has a probability of being successfully implemented, depending on many factors. For example, even if a weak password vulnerability exists in an authentication scheme, and the threat has the resources and knowledge required to deploy it, it still may be unable to recover a valid password in the defined time frame. It may seem that nothing was gained, since probabilities still need to be evaluated, and now the analysis of the attack graph is more complex due to the probabilities that have been introduced. However, these probabilities can be determined more easily than those of a complex attack. In other words, it is easier to evaluate the success probability of one step in a path (e.g., a single attack), than the success probability of the entire path (e.g., a complex attack). This is also confirmed by other approaches that define the various factors that influence the occurrence of an event.[10]

For example, one can consider the path corresponding to a complex attack that includes a spear-phishing attack against a user component, an attack on the local area network (LAN) to the manager's personal computer (PC) and a "legitimate" access to DB2. The probability of a user being vulnerable to spear-phishing can be locally evaluated, e.g., through a penetration test simulating this specific attack. The probability of a PC being vulnerable to attacks from another PC on the same LAN can be evaluated based on the frequency of security bulletins for the adopted operating system and related to this kind of vulnerability.

These few examples show that in the proposed model, the success probability of a complex attack is derived from the probabilities of local and more measurable factors. However, the overall complexity cannot disappear, as confirmed by the increase in the complexity of the analysis of an attack graph with a probabilistic component. Haruspex faces this problem by applying the Monte Carlo[11] method, a well-known and widely applied strategy to collect statistical data on complex events when no mathematical model of the event is available. The Monte Carlo method can be explained with a simple example. If one wants to know the probability of a specific sequence of cards in a game that is too complex to calculate the result, one can play several hands of the game and count how often the card sequence appears. Provided that one plays enough hands, the count will be a good approximation of the actual probability.

Before explaining the role of the Monte Carlo method in the Haruspex methodology, it is important to consider how threats, or better threat agents, are modeled. Dealing with IT security, one focuses on intelligent threat agents (i.e., agents with their own resources, a strategy and usually goals with an impact). A nonintelligent threat can be modeled as a threat with a strategy based on random or fixed behavior. Intelligent threats will have the ability to select their actions based on many factors, including possible countermeasures. A key point is that, as already mentioned, Haruspex does not support an analysis to discover which threats may attack the system, or the frequency or reason why they will attack the system; this evaluation is left to other methodologies that the analyst can freely select.[12] But, if the analyst decides that a given threat will attack the system, Haruspex will evaluate the

probability of that threat reaching some or all of its goals and impacting the system. In this way, the selection of the threats and their interest in the system can be dealt with separately in the risk-evaluation process.

To better explain how Haruspex uses the Monte Carlo method, let us consider another application of the method: the computation of the area under a convex curve that lies in a finite rectangle. To compute the area of interest, one can generate a number of independent random points (n) that are uniformly distributed within the rectangle—the two coordinates, for example, are uniformly distributed for each side of the rectangle. Then, the area delimited by the curve can be approximated by the product of the area of the rectangle multiplied by the percentage of generated points that have fallen under the curve. As an example, a unit square and the circle inscribed in this square have a ratio of areas that is $\pi/4$. Hence, the value of π can be approximated with the Monte Carlo method previously described. The error in the approximation can be reduced by increasing n, by improving the uniformity of the point distribution in the rectangle and by assuring that successive points are independent. Obviously, some proper deterministic algorithms have to be selected to generate the points.

The value that Haruspex tries to approximate is the probability that a threat succeeds in implementing a complex attack. The adoption of the Monte Carlo method implies that Haruspex simulates several times, in distinct experiments, the attacks implemented by the threat, and computes the success probability according to the number of times the threat is successful. A single experiment would be useless, since it would be biased by the specific values selected to generate the random values in that experiment. Data are collected in each experiment to compute the relevant statistic values, and confidence in these statistics increases with the number of experiments.

In each experiment, Haruspex simulates a set of agents, each trying to achieve some goals. Each threat agent starts with his/her set of privileges and has access to a set of resources in the system (e.g., public resources or the ones available for an internal role).[13] At each simulation step, corresponding to a time slice (e.g., one day):

- The simulator computes the vulnerabilities that the threat agent discovers in that time slice
- According to the resources, strategy, privileges and available vulnerabilities, each threat tries to advance by selecting and implementing an attack to acquire further privileges through some of the available vulnerabilities
- Based on the success probability of the selected attack, the simulator computes whether the attack is successful and grants to the threat agent the additional privileges it has possibly acquired. The threat agent will deploy them in the next time slice.

It is assumed that the success probability of the attack makes it possible to model the various factors, such as the time when the attack is attempted or the existence of controls that may detect the attack, that influence the success of the attack. In each simulated time step, Haruspex repeats the discovery of vulnerabilities and the implementation of attacks by the threats until the experiment ends—because either all the threat agents reached their goals or all the time slices were consumed.

Provided the number of experiments is rather high, the analyst can collect enough statistical data on successful attacks to be able to answer, with confidence, the question: *If this threat were to attack the system, what would be the probability that the threat would achieve some goals and cause an impact?* Or, given a proper definition of risk: *What is the risk associated with an attack by the threat?* What makes Haruspex unique with respect to other methodologies and other tools is its ability to join the Monte Carlo method for the probabilistic part of the model with the intelligent behavior of threat agents.

The ability to support accurate and specific countermeasure selection, both in design and audit, is something especially useful in Haruspex. First of all, several other methodologies assume that countermeasures remove vulnerabilities. This is seldom true; instead, countermeasures usually reduce the success probability of attacks or the existence probability of a vulnerability. On the other hand, Haruspex supports the evaluation of countermeasures in the general case.

To show how Haruspex can be used, let us take a case in which some procedure has been defined to select a proper set of countermeasures according to their cost, the number of agents they can stop or some proper combination of these factors. Haruspex is not involved in the selection process, but given a set of countermeasures that reduce the success probability of some attacks, it can be used to implement a new simulation to evaluate the actual overall risk reduction

enabled by the selected countermeasures. In this way, the effectiveness of alternative sets of countermeasures can be evaluated in distinct simulations to discover the most cost-effective one.

In the previous example, DBMS hardening, while probably more expensive, may be less effective than a targeted training for managers against phishing and the activation of segregation controls on network devices for network traffic.

USABILITY

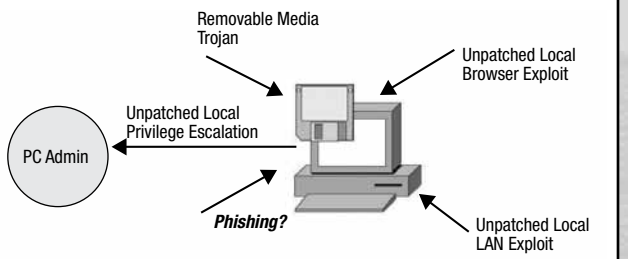
The adoption of Haruspex to evaluate real systems poses two main problems: the complexity of system modeling and the evaluation of the local probabilities for component vulnerabilities and attack success.

With respect to system modeling, Haruspex does not force the modeling of the entire system from the beginning at the same detail level. For example, the analysis that has been previously outlined represents the web server/web application component as a single component.

If a more in-depth analysis of this component is required to select the most effective countermeasures, this analysis can decompose or zoom in on this component. Provided that the same relations with other components are maintained, the exploded subsystem can replace the original component without changing the rest of the system model. In this way, a high-level analysis can be quickly performed, adding more details afterward. Vulnerabilities, attacks and their probabilities must be assigned to the entire component; however, an additional feature of Haruspex is that probability ranges can be tested with simulations to assess how much a change in a local probability affects the overall risk. In some cases, this may avoid a useless effort in defining a precise value for a not-so-relevant probability.

Haruspex also encourages the creation of libraries of predefined components, each with its vulnerabilities and probabilities that can be made available to analysts in an interface for system modeling. For example, a Windows 7 desktop with its typical applications is a component that is almost the same in distinct organizations and can be defined with its typical attack channels and the probability of the related vulnerabilities (see **figure 4**). These probabilities can be based on the frequency of bulletins and on the data shared among organizations, and could then be personalized by the analyst for a specific organization (e.g., according to local configuration policies and patch management).

Figure 4—Vulnerabilities of a Standard Component



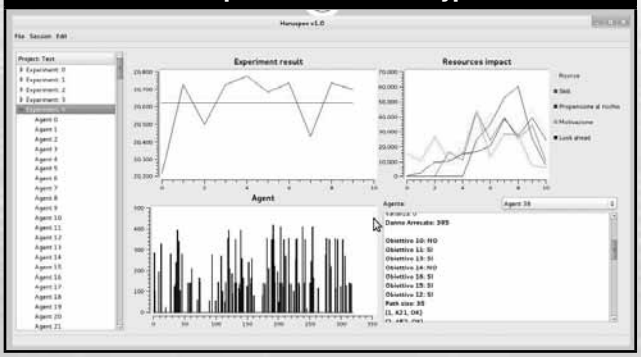
This, in turn, could encourage information sharing between organizations (e.g., between computer security incident response teams [CSIRTs] or in industry associations). It is easier to share information on single components and their vulnerability frequencies than on entire complex attacks that may expose too much information on the organization policies and on impacts. In fact, a lot of statistical data are already collected by many companies (e.g., antivirus vendors). Local information could be collected by vulnerability assessment tools.

CONCLUSION

This article has presented the basic characteristics of Haruspex, but the potentialities of Haruspex are, by and large, still to be explored. For example, Haruspex can also model defenders as additional agents that try to close the vulnerabilities optimally (e.g., according to the information returned by sensors on the progress of threat agents in the system). The authors are currently planning several simulations to validate and evaluate the current prototype (see **figure 5**).

By returning important parameters that describe the behavior of threat agents, the attacks they can successfully implement and so on, Haruspex can also support the definition and evaluation of algorithms to select proper countermeasures. In fact, current strategies select countermeasures in terms of the attacks a threat may implement, rather than based on statistics of the attacks that the threat will actually implement and reach its goal. More accurate information on the attacks of a threat can result in a more realistic allocation of the limited resources of countermeasures to maximize the return on the investment. Further applications of Haruspex are the simulation of modern worms that patch components once they have been conquered, or the modeling of threat strategies to select attacks. The simulation produces a huge amount of data on the system, the attacks and the threats, which can be used for data mining, looking, for example, for correlations between attacks.

Figure 5—Screenshot of the Haruspex Simulator Prototype



AUTHORS' NOTE

The authors are interested in interacting with the ISACA community to apply Haruspex to real-world complex systems, so that the most useful and promising research paths can be explored. Another interesting open-source project that we would like to undertake is the development of libraries to describe standard system components. The authors can be contacted at *haruspex@di.unipi.it*.

ENDNOTES

[1] The name "Haruspex" was originally the name of ancient forecasters from Tuscany who predicted the future by interpreting the entrails of sacrificed animals—mainly the livers of sheep and poultry.

[2] Hubbard, Douglas W.; *The Failure of Risk Management: Why It's Broken and How to Fix It*, Wiley, USA, 2009

[3] Risk Management Insight LLC, "An Introduction to Factor Analysis of Information Risk (FAIR)," November 2006

[4] Camtepe, Seyit; Bulent Yener; "Modeling and Detection of Complex Attacks," *Security and Privacy in Communications Networks*, 2007

[5] Baiardi, Fabrizio; Claudio Telmon; Daniele Sgandurra; "Hierarchical, Model-based Risk Management of Critical Infrastructures," *Reliability Engineering & System Safety*, September 2009

[6] A spear-phishing attack is a phishing attempt in which a user is invited to access some dangerous site that injects malware into the user's machine. This attack is generally targeted against a small group of select users who are more likely to be attracted and behave as expected by the agent implementing the attack. For further references, refer to: Egelman, S.; L. Faith Cranor; J. Hong:. "You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings," 26th annual SIGCHI Conference on Human Factors in Computing Systems, ACM, USA.

[7] Baiardi, Fabrizio; Fabio Martinelli; Laura Ricci; Claudio Telmon; "Constrained Automata: A Formal Tool for ICT Risk Assessment," NATO Advanced Research Workshop on Information, Security and Assurance, June 2005

[8] Noel, Steven; Sushil Jajodia; Lingyu Wang; Anoop Singhal; "Measuring Security Risk of Networks Using Attack Graphs," *International Journal of Next-Generation Computing*, July 2010

[9] Baiardi, Fabrizio; Claudio Telmon; Daniele Sgandurra; "A Simulation-driven Approach for Assessing Risks of Complex Systems," 13th European Workshop on Dependable Computing, May 2011

[10] Kim, Jae-n; Charles W. Mueller; *Introduction to Factor Analysis: Why It Is and How to Do It*, Sage Publication, 1978

[11] The Monte Carlo method takes its name from the well-known casino. It was first invented by Enrico Fermi and later adopted at Los Alamos, New Mexico, USA during the Manhattan Project, which resulted in the building of the first atomic weapons. For further details on the genesis of the method, refer to: Metropolis, Nicolas; "The Beginning of the Monte Carlo Method," *Los Alamos Science*, no. 15, p. 125. For a more complete reference, see: Kroese, D. P.; Taimre, T.; Botev, Z.I.; *Handbook of Monte Carlo Methods*, John Wiley & Sons, USA, 2011.

[12] Several methods and tools have been defined to discover and model the threat agents that may be interested in attacking a system. For a detailed analysis of these tools and these methods, refer to: European Network and Information Security Agency (ENISA), Inventory of Risk Management/ Risk Assessment Methods, *www.enisa.europa.eu/act/rm/cr/risk-management-inventory/rm-ra-methods*.

[13] Each agent to be simulated is modeled in terms of goals and resources it can access, which, in turn, determine the attack the agent can implement. Due to the existence of several threat agents, the resulting simulation can be described by an agent-based one—even if, in the current implementation, each threat agent is represented through a distinct data structure rather than by a distinct agent.