After a period of approximately 3 months after implementing the security measures, a new security audit should be taken. The new security score Ss is calculated and compared to the stated aimed Security score using the security measures. If there are security factors that score too low, these should be investigated and adjusted.

Conclusion

The awareness that security is a management problem is everywhere present. It's critical to know what are the critical resources and processes of the company and their weaknesses. Our security audit is a handy solution. We have developed BEVA, a method to critically analyse the company and to uncover the weak spots in the security system. BEVA results in security scores for each security factor and also in a general security score. The goal is to increase the security score Ss to a postulated level by focusing on the critical security factors, those with a low security score. The results of the audit are an ideal start to do risk analysis.

Bibliography

[Shannon, 1949] C.E.Shannon. The Mathematical Theory of Communication. In: The Mathematical Theory of Communication. Ed. C.E.Shannon and W.Weaver. University of Illinois Press, Urbana, 1949.

[Jean-Marc Lamère] la sécurité informatique; Dunod : La méthode MARION (Méthodologie d'Analyse de Risques Informatiques Orientée par Niveaux) www.eisti.fr/~bg/COURSITACT/TXT/m_marion.txt

[Val Thiagarajan B.E, 2005] Information Security Management; BS ISO/ IEC 17799:2005; SANS Audit Check List: author:., M.Comp, CCSE, MCSE, SFS, ITS 2319, IT Security Specialist.

Security Management: A New Model to Align Security with Business Needs; Sumner Blount, CA Security Solutions; August 2006

[Schreurs J, Moreau R.] ICT security management- ECEC 2007 (www.riskworld.net/7799-2.htm)

[Lawrence A. Gordon, Martin P. Loeb, William Lucyshyn and Robert Richardson] 2006 CSI/FBI-study about cybercrime: COMPUTER CRIME AND SECURITY SURVEY

https://event.on24.com/eventRegistration/EventLobbyServlet?target=registration.jsp&eventid=27372&sessionid=1&key=42F39B89EE0B30BA951711A5E7A98EDD&sourcepage=register

http://mediaproducts.gartner.com/gc/webletter/computerassociates/vol3issue3_risk/index.html

Authors' Information

Jeanne Schreurs – prof. Business informatics, Universiteit Hasselt; gebouw D, Agoralaan, 3590 Diepenbeek, Belgium; e-mail: jeanne.schreurs@uhasselt.be

Rachel Moreau - Universiteit Hasselt; gebouw D, Agoralaan, 3590 Diepenbeek, Belgium; e-mail: Rachel.moreau@uhasselt.be

COMPLEX PROTECTION SYSTEM OF METADATA-BASED DISTRIBUTED INFORMATION SYSTEMS

Denis Kourilov, Lyudmila Lyadova

Abstract: A description of architecture and approaches to the implementation of a protection system of metadata-based adaptable information systems is suggested. Various protection means are examined. The system described is a multilevel complex based on a multiagent system combining IDS functional abilities with structure and logics protection means.

Keywords: adaptable information systems, protection mechanisms, metadata, multiagent systems.

ACM Classification Keywords: D.2 Software Engineering: D.2.0 General – Protection mechanisms; K.6 Management of Computing and Information Systems: K.6.5 Security and Protection – Authentication, Insurance, Invasive software (e.g., viruses, worms, Trojan horses), Unauthorized access (e.g., hacking, phreaking); I.2 Artificial Intelligence: I.2.11 Distributed Artificial Intelligence – Multiagent systems.

Introduction

Modern information systems (IS) are developed for various application domains and environments that influence their protection and reliability level. The typical peculiarities of modern IS are:

- *Complexity*. As the complexity of information systems grows, they reveal more and more vulnerabilities that are difficult to disclose and repair.
- *Openness and integrability*. The openness of information systems and their integrability, and their being interconnected with internal IS are potentially responsible for IS intrusion vulnerability.
- *Adaptability and expandability*. IS are flexible enough to be configured for certain working conditions and users' needs, internal developers can also expand the systems' functions. This also creates the risk of malware intrusion.
- *IS are distributed*. IS' subsystems can interconnect via network, posing extra security threats, such as IS and client back-ends attack.

Security methods existing today do not allow us to fully protect dynamically adaptable IS that function in distributed environment. They can protect either the program code, or IS data. IS can be adapted with: database dynamic restructuring tools; automatic generation and tuning user interface; query and reporting facilities; business process management tools; connection oriented services for software components created by outside developers. In view of this, there is a growing importance of such problems as IS security, protecting IS resources and software from unauthorized access and distribution. Dishonest users being qualified and having the provided tools at their disposal, can abuse software technologies for adaptable systems.

According to the approach presented in the article, IS software functioning in distributed environment is considered as an integrated software product. Complex protection of IS software and IS defining data and metadata is necessary. Complex protection implies IS data and program code protection and choosing the best licensing scheme.

IS are traditionally considered as composite software complexes, the components of which are set up on network nodes and interconnect via data transfer through communications links. This view of IS generated a congruent approach to IS security management. The approach implies exploiting various security mechanisms to protect network nodes from unauthorized access and resource usage (particularly, protection from malware including different viruses and Trojan software). Besides, this approach includes network interaction channels protection with a range of hardware and software tools (for example, shielding, network traffic analysis, etc). Such an approach to security organization is quite applicable and reasonable for protection of software systems found on separate workstations or within a small network. However, in case of distributed IS that go beyond the bounds of a separate PC or a small local area network, a number of significant disadvantages of the traditional approach can be pointed out:

- *It is difficult to maintain IS security at a proper level*. Since most services for information security support (such as Symantec Intruder Alert, family of ISS RealSecure systems and others) are focused on signature methods of intrusion detection, they require regular updating at all IS nods. This investigation is devoted to the approach to security system management which does not deny services of this kind, but allows dependence on them to be reduced.
- *Information systems are considerably vulnerable to new types of intrusions*, for instance to those based on detection and usage of the IS vulnerabilities that have not been abused yet, including operation system and network software vulnerabilities. The reason for this is signature methods of analysis predominating in today's market of information security systems.
- *There is practically no protection from intrusions that were specifically worked out for hacking a certain IS*. These intrusions are based, particularly, on vulnerabilities and errors in program realization of IS modules. Consequently, there is a need in additional protection agaist attacks performed "within" the IS, with the use of previously hacked modules.
- *There is a need in additional tools to control input and output dataflow*. Input dataflow control implies protection against spam, phishing, malicious ad-ware and other similar external threats. Output dataflow control involves scanning all outcoming information transferred to external systems, thus detecting the protected corporative information.

- *It is difficult to provide a sufficient user authentication level*. In large IS containing protected data and services, it is inefficient to use a standard user identity check method which is based on checking the knowledge of a secret password. In this case there is a risk of information leakage.

The traditional approach to distributed IS security has a lot of drawbacks, because most security tools *do not utilize information about the structure and semantics of the IS under protection*.

Security System Architecture

In this investigation a conceptually new attitude to security system design is suggested. Information system is not considered as a complex of computation nodes that interact while functioning. It is rather regarded as a complex of services provided by the IS components which are implimented in a number of interrelated network nodes. It should be pointed out here that IS as a *complex of services* requires total security instead of protecting separate structure units and data channels.

The approach under consideration is justified due to large IS' enhancement tendency which aims to provide users with daily necessary functions within a single integrated system (for example, such common operations as entering and editing data within business processes automated by IS components, exporting and importing e-mail, data processing and report generating, etc).

The authors suggest an approach to designing an integrated security system for protecting dynamically adapted distributed information systems, based on using metadata defining all aspects of IS functioning. The architecture of an integrated security system includes several levels. It combines various security tools integrated with the IS being protected, using this IS defining metadata.

The suggested security system is a multi-level complex, designed on a *multi-agent system* (MAS) basis. The complex combines the functionality of modern *intrusion detection systems* (IDS) and the tools of *IS structure and program logic security*.

The security system is based on *distributed MAS*. System agents' community is *closed* and *protected* against malicious influence from the outside with the help of its own security mechanisms as well as the way the agents' work is organized. *Each agent is an independent entity that covertly functions within the system under protection*. The information about the agents can be found nowhere in the system beyond the agents community which operate in the system and in the threads where they are run. Hiding is implemented by means of two main methods: agent execution in the threads hidden in the OS core with the help of a security driver, and agent execution in the thread of the protected system with the help of the thread context switching mechanism, actively used by the operating system [2].

All the MAS agents fall into two classes: *analyzer agents* and *sensor agents*. Analyzer agents are intellectual agents built on the basis of the InteRRaP architecture that belongs to the class of multi-layer architectures with vertical layer division [4]. Each layer implements a certain type of an agent's interaction with the environment (system area where this agent operates). The current system-status information is transferred from the lower layers to the higher layers, control is transferred from the higher to the lower layers.

The agent structure is represented by the 3 layers:

- *The layer of behavior* is responsible for reactivity, real-time behavior. The responsibility area of this layer is decision-making under typical circumstances, the examples of which can be user registration, remote network node connection or an intrusion attempt of a known type, the signature (i.e. script) of which is already in the system.
- *The layer of planning (local planning)* – is realization of the *cognitive paradigm* of MAS building. As the information is transferred from the layer of behavior, there is the agent's knowledge base inference on the layer of planning. The aim of this process is to evaluate the class of the current situation and to choose an adequate behavior template (set of responses to the changes in the environment state) in order to further apply it on the level of behavior.
- *The layer of communication (collective planning)* is responsible for realizing the *mechanisms of agents' communication*. This layer represents the possibility of decision-making on the basis of the data arranged by the other agents of the system. It also in charge of controlling team work.

The knowledge used by the agents to evaluate situations is introduced in the *frame paradigm* [3], the rules for decision-making are presented on the condition-action basis.

Sensor agents are used to collect the data about the current state of the security system, of the information system and its modules, as well as the network in which the IS is functioning. These agents are implemented on the basis of Reactive Architecture [4], they serve to collect statistics, post events and detect anomalies on the basic level.

Agents interact on the Contract Net model that implies solving different tasks directing them to the most suitable agents. This model was chosen since it has a number of advantages that make it comply with the requirements as fully as possible:

- Each agent has a functionality system that allows performing some tasks involving no other agents of the system (high level self-efficiency of agents).
- There is a small time interval from when a problem appears to when the process of solving starts.
- There is little possibility of incorrect problem-solving since problems are directed to more competent agents that contain all the necessary functionality.
- There is less overhead expences, as there is no need in every agent regularly analyzing the current system status.
- The high efficiency of system control results from the agents being subject to arrangement into hierarchic structures.

Levels And Mechanisms of Security

Security of Structure and IS program logics is an indispensable part of protection, they are unreasonably ignored by modern information security systems due to the fact mentioned above – they contain no information about semantics of the IS they protect. *IS structure security* serves to prevent hacking program modules from replacing server-side and client-side components of IS, as well as to raise the efficiency of protection measures against unauthorized connections to the IS services. *Program logics protection* is aimed at restraining unauthorized attempts of IS program code modification, which can be intended to error injection for building back doors [5] to arrange subsequent intrusions.

Information about the protected IS semantics is introduced by means of a *hierarchic 3-layer model* that fully describes all the security-critical aspects of the IS.

All the information on IS functioning, and its application environment is distributed among the three layers of the system security model $S = (Str, Ev, Msg)$ where

- *The Layer of structures Str* contains description of the distributed IS structure, including information about network nodes and application domains (IS subsystems), communication channels through which subsystems interconnect. In the model, the level of structures is presented by *P-graph (graph with poles [7]) Str = (N, A)*, where N is a set of vertices with poles representing application domains, network nodes; A is a set of arcs connecting them and representing communication channels.
- *The Layer of events Ev* = $\{T, E, Q, Init(Q), Init(E), Ch, Sch\}$, where T is a set of time moments, E is a finite set of events; $Init(Q): T \to Q$ is a mapping of the initial state; $Init(E): T \to E \times T$ is a mapping of the initial event planning; $Ch: E \times Q \times T \to Q$ is a mapping determining the new state to which the system changes as a result of an event; $Sch: E \times Q \times T \to E \times T$ is planning ratio that represents cause-and-effect relations of the events. The layer of events displays IS operation description in time. This layer comprises data on different states the system may be in, and events causing change of states. Representation of this layer in IS model is a *directed graph*, the vertices of which correspond to IS states at different instants of time. The arcs show events (including those related to receiving messages), causing change of states. This set can be tied with each vertex of structure Str.
- *The layer of messages* Msg comprises description of data which can be shared by the subsystems of IS, and rules for this data conversion. The layer is specified as determination of the layer of events.

The security system is also a multilevel one, it includes the following levels: basic security logics level, privilege control level, inherent security level, system security level.

The multilevel approach to security engineering, above all, makes it possible to *independently design various protection mechanisms*. Particularly, it has become possible to put the “high-level” security logics into practice (for example, activation entry checking), on the ground that it is supposedly impossible for an intruder to modify the program code performing these functions, because the code is protected at another level.

At the level of *basic security logic* essential functionality is realized, irequired from intrusion detection systems in accordance with ISO 15408 Standard: network traffic control, information system services control, anomaly detection.

The main mechanism of this level is an *active audit* subsystem which realizes the statistical and signature approach to activity detection and analysis. These approaches are described in FAU_SAA Security Audit Analysis requirements. The leading function of the the active audit subsystem is detecting anomalies in IS operation. Generally, any intrusion attempt is an anomaly pinpointed by means of a statistical analysis of the IS operation for a long time period. Top priority is given to analyzing how different services operate at the IS server modules.

There is a need in compensating the disadvantages of the statistical approach to analyzing such activities as complex decision-making in the context of the lack of an established empirical facts basis, and difficulty of attack detection in case activity parameters are gradually modified towards those typical of an attack. Aiming at the compensation, within an active audit subsystem, one applies the signature method of malicious activity detection that agrees with FAU_SAA Complex Attack Heuristics requirements.

In this context, signature is understood as a certain sequence of events, characteristic of a system cracking attempt. Efficiency of active audit mechanism is achieved by using the possibilities of a distributed multi-agent system being the basis of the security system. Information sent by sensor agents from various nodes of the network, is received by analyzer agents which are responsible for this data processing and forming the summary of the current system status and its potential security threats. The analysis is produced on the basis of the hierarchic three-layer IS model which also contains information about the security system itself.

The level of privilege control has a function that supports control of the system users' rights based on the stored profiles of activity, according to FAU_SAA.2 Profile Based Anomaly Detection requirements. As a rule, the aim of attacking large corporative IS is obtaining access to confidential data or protected services. It eventually means obtaining high level of privileges in the attacked system [5]. The main mechanism of this level is users' activity analysis subsystem.

In IS, some *user groups* are distinguished, each of them possessing a definite *privilege set*. During the process of the IS configuring and testing, statistical data about activity types of the use groups is collected, and group models represented by activity graphs, are formed. A group model includes the information characterizing the behavior of the user group members when logging into the system, working in the system and logging out of the system. After the group models are built, an individual model is constructed for each user.

The behavior model is a directed graph $G = \{V, A\}$ where $V = \{v_i\}$ is a set of vertices in which the order relation is defined according to the following rule: the element included in set V last, has a higher number in it; $A = \{a_{ij}\}$ is a set of arcs of graph G. Each element $a_{ij} \in A$ is put in correspondence with some weight $w_{ij} \in W$, where W is a set of admissible weights of arcs. Vertices $v_i \in V$ represent values of the controlled parameters. Arcs $a_{ij} \in A$ represent semantic relations among the controlled parameters' values, characterizing the order of adding vertices matching parameter values, i.e. the elements $v_i \in V$, to graph G. Weights $w_{ij} \in W$, appointed to arcs $a_{ij} \in A$, fix semantic distances among the values of the controlled parameters by means of the corresponding vertices incident to these arcs $v_i, v_J \in V$. Semantic distance characterizes the difference among the values of the controlled activity parameter. The model allows controlling the correspondence of parameters to some reference values, "accumulating" the changes for the subsequent analysis.

The models are based on the analysis of different types of *users' activity parameters*:

- *Categorial parameters*. The examples of categorial parameters can be changed files, records in the database, IS services in usage, initiated commands, types of errors, etc. Categorial parameters analysis has an *event-oriented* character.
- *Numeric parameters*. This type comprises any activity parameters, which can be valued numerically - for instance, the quantity of transmitted and requested information, the number of services being in use simultaneously, as well as the number of vertices and arcs of the model.
- *Intensity parameters*. For example, the number of the user's entries into the system during a certain period of time, intensity of the database queries, and the like.
- *Event distribution* parameters. This type may include the frequency ratio of such events as view query and change query, references to certain IS services.

The models are mainly applied via realizing authentication mechanism based on correlating the current user's behavior with statistics on his usual activity parameters. This mechanism is an *addition to the standard authentication mechanisms*; it is aimed at protection from unauthorized access to privileges via the legal users' identity theft.

Individual users' models and group models in dynamically configurable IS can be also utilized for the purposes that are not related to security; for example, user interface automatic generation and configuration, based on statistics of an IS functionality being applied by the user or user group.

The security levels described above are projected, according to the statement that it is impossible for an intruder to modify the program code. On the *inherent security* level some mechanisms are applied to protect the IS program code from analysis and modification.

The key mechanisms of this level are:

- The mechanism of *explicit program code entity control*: it initiates an instant reaction of the security system. Within this mechanism, the application program code is checked for unauthorized changes, and cryptographic security of program modules is realized.
- The mechanism of *implicit control* is used to arrange the deferred system reaction to intrusion, with the purpose of preventing the cracked application from being used. As it becomes evident that an intruder modified the program code or deactivated either security mechanisms of the first levels or the explicit control mechanism, the security system is switched to imitation mode. However, there are no signs of attack detection, but the IS modules that had been abused, get actually isolated, i.e. it is impossible to use them for accessing the key data and IS services.
- The mechanism of *concealing the location of the security system functions*. This mechanism is chiefly aimed at the functions that are responsible for the user feedback. For instance, protected service lockout displays messages on access restriction in case attacks are detected (the so called nag screens). Feedback functions generating this kind of messages are in most cases a convenient starting point for the system hack [8]. Concealment is performed by exporting all the vulnerable functions to the dynamically generated program modules. Besides, the functions generating "dangerous" messages are not saved in the application files, it is difficult enough to detect and modify them.

System security level. The majority of malicious programs can not function without obtaining certain privileges which give access to protected system functions. System functions access is necessary for such tasks as opening network ports (e.g. for interaction with a trojan module installed in the attacked system), executing programs in debug mode (in order to find security breaches), access to protected external memory partitions or address spaces of the executed programs as well as to input/output controllers. Ideally, the code becomes available for execution at ring 0 privilege level. It allows a direct access to any resources of the attacked system, including functions of the operating system kernel and physical units. Kernel level security serves to prevent intruders from access to protected OS functions and OS kernel in particular.

The *main mechanisms* of this level are: the mechanism of processes detection, the mechanism of network interaction control, the mechanism of ring 0 security.

The mechanism of *network interaction control* analyzes network ports status in order to disclose unauthorized attempts to open new ports and change running modes of the active ports.

This mechanism is applied by tracking calls of the corresponding OS kernel functions (it is Native API [8] for Windows operating system), it is performed by means of installing shells, realizing callback interfaces, on these functions. Kernel calls tracking is a sufficient condition for detecting unauthorized access to the OS functions which are potentially dangerous in the context of secure operation management. The reason is that calling any function of application interfaces leads to calling one of the OS kernel functions. Besides, in most cases one OS kernel function comes with several different application interface functions, which are in fact its shells making kernel function calls with a certain set of parameters [8]. The kernel level control can only guarantee security that doesn't depend on possible appearance of new program interfaces and new ways of access to potentially dangerous OS functions.

Security mechanism of the ring 0 privilege level protects the functions performed on the ring 0 privilege level of the system. The greatest security threat is presented by the so-called hacking tool kits [5] – rootkits – that work on the ring 0 privilege level. It is a sort of malware that enables the intruder to obtain almost full control over the infected system and isn't practically subject to detection and liquidation.

A rootkit can be realized in the form of a separate driver or a shell of some OS kernel function. Rootkit intrusion is prevented by *controlling OS kernel function calls* that are responsible for drivers and images uploading to the system. Detecting rootkits installing shells in the OS kernel functions is not technically difficult, as possessing information about the initial structure of the kernel functions is enough to detect unauthorized modification. Within this mechanism, there is *regular verification of the hash functions values* (*that is computed from the program code of the OS kernel functions*) to correspond to the reference values. These values are derived after security system setting-up and authorized changes in the functions. An additional sanction can be tracking attempts to memory access based on addresses matching the OS kernel functions. However, it will inevitably lead to a noticeable decrease in the secured system's productivity, that's why this sanction can be applied only in cases when the maximum level of security is required.

Hidden processes detection mechanism (of the processes invisible on the application level) is aimed at detecting malware that is able to operate on the application level. Hidden processes are disclosed with the help of a *security driver* operating on the system level.

Conclusion

The main efforts of the suggested security system are:

- *Adaptability*. The suggested security system can be adapted to new threats via modification of its knowledge base.
- *Universality*. The suggested security system is based on a multilayered model of the protected IS and therefore can be integrated to nearly any information system.
- *Extensibility*. The suggested security system is knowledge-based therefore its functionality can be extended even without providing changes in its structure or source code.
- *High performance*. Metaknowledge and knowledge on protected IS are used to maximize security system performance.

References

[1] Лядова Л.Н. Архитектура информационной системы «Образование Пермской области» // Математика программных систем: Межвузовский сборник научных трудов / Перм. ун-т. Пермь, 2002. С. 25-35.

[2] Кастер Х. Основы Windows NT и NTFS / Пер. с англ.— М.: Издательский отдел «Русская редакция» ТОО «Channel Trading Ltd.», 1996.

[3] Минский М. Фреймы для представления знаний. М.: Энергия, 1979.

[4] Huhns M., Stephens L. Multiagent Systems and Societies of Agents // Weiss G. Multiagent systems: a modern approach to a distributed artificial intelligence / Massachusetts Institute of Technology.

[5] Хогланд Г., Мак-Гроу Г. Взлом программного обеспечения: анализ и использование кода. М.: Вильямс, 2005.

[6] Лядова Л.Н., Мороз А.А. Модель защиты программного обеспечения от несанкционированного распространения // В кн.: Сборник трудов Второй международной научно-технической конференции «Инфокоммуникационные технологии в науке, производстве и образовании» (Инфоком 2) / Кисловодск, 2006. С. 120-124.

[7] Миков А.И. Автоматизация синтеза микропроцессорных управляющих систем. Иркутск: Изд-во Иркут. ун-та, 1987.

[8] Касперски К. Техника и философия хакерских атак. М.: СОЛОН-Пресс, 2004.

Authors' Information

Denis Kurilov – Perm State University, Graduate student of the Computer Science Department; Bukirev St., 15, Perm-614990, Russia; e-mail: Denis.Kurilov@gmail.com.

Lyudmila Lyadova– Institute of Computing, Deputy Director; Podlesnaya St., 19/2-38, Perm-614097, Russia; e-mail: LNLyadova@mail.ru.