Securing Automation Systems Against Malware Intrusion

R. Fitz and W. A. Halang

***Abstract*— Conventional measures do not sufficiently protect computing systems anymore against intruders and malware of any kind. The main reason for this is that the system architectures are based on highly insecure and error-prone foundations. Whereas some time ago this shortcoming could still be partially coped with by swift counteraction, today this "race" must be considered lost right from the start due to the fast data networks. There are no reactive measures anymore that could compensate for the aggressors' temporal advantage. Since computers employed for automation and control purposes are more and more connected to networks and are, thus, endangered by malware, new architectures for their hardware and software as presented in this paper are necessary, which solve the security problems by their intrinsic properties.**

***Keywords*—** *Computer control, automation, safety-related control, security, malware, intrusion prevention, hardware-based security measures.*

I. INTRODUCTION

IT has become fashionable to employ even for safety-related tasks in automation technology computers whose hardware and software are neither secure against intruders nor able to provide acceptable real-time performance. Thus, to avoid conversions and to minimise times necessary to become acquainted with adequate industrial systems, more and more automation applications are implemented on the basis of cheap PCs as control computers and the popular Windows operating systems. As such computers are swamped with attacks for already some time now, there is a considerable risk also for industrial computing systems to be infected by malware like, for instance, Stuxnet [6] and, thus, to become unsafe. This is exacerbated by the presence of almost any enterprise in the Internet, and since firewalls are unable to protect the intranets of enterprises against external attacks.

Primarily the fast, high-capacity global communication networks and the monoculture in hardware and software technology has led to this situation, which favours the swift spreading of malware. When an electronic intruder was detected in former years, the companies dealing with counteracting them usually had sufficient time to update their products. Owing to the fact that customary software products can provide just a certain degree of protection against already known and analysed electronic malware, most computers are defenceless in the hands of new, not yet sufficiently analysed destructive programs. As there are some tens of thousands new ones of such programs every day according to studies of Bundesamt für Sicherheit in der Informationstechnik (German Federal Agency for Security in Information Technology) [1], some experts now recommend to update the installed "antivirus software" already on an hourly basis, in order to be able to provide, at least, a certain "basic level of protection".

Due to the system homogeneity mentioned above and the increased speed of the proliferation of malware, by now it is a generally accepted fact that trying to warrant security with malware detection programs and firewalls is not an adequate solution anymore. Hence, the security problem must be solved in a fundamentally different way by appropriate architectures of hardware and software. To this end, constructive security measures are presented in this article, which render the virus problem manageable and, thus, contribute to the ultimate solution of this kind of security problems. The feasibility of building systems which can match the contemporary potential of threat will be shown constructively. Moreover, it turns out that such systems can even be maintained more easily as well as can provide higher performance and greater robustness as the automation systems presently prevailing.

An analysis of the various intruders, particularly in form of programs and executable Internet content with malicious intentions, reveals that they are based on some common principles of operation. If these operation principles are thwarted by appropriate measures, malware is prevented from spreading and from launching its destructive effects. The security measures presented in the sequel disable the operation principles of all known malevolent programs in an effective way. In developing them, great importance was attached to the presented solutions being simple and easy to duplicate, in order to be understood and applied without any problems by the users of computers, as unnecessary complexity is the enemy of any effort towards enhancing security.

Recently discussed approaches based on cryptography can be ruled out because of their lacking verifiability and unnecessary complexity, in particular for use in automation technology. Their benefit for the users in improving the security of conventional systems is very doubtful in consideration of the fact that there is no practically applicable cryptographic method known which could not be deciphered by attackers – let alone the costs incurred and the performance absorbed by encoding and decoding data. Moreover, in the past experience has shown that cryptographic solutions provoke playfulness, and even persons without malicious intentions feel urged to decipher systems protected this way: a kind of competition or popular sport has emerged.

R. Fitz, Hochschule für Angewandte Wissenschaften Hamburg, Germany, robert.fitz@haw-hamburg.de
W. A. Halang, Fernuniversität in Hagen, Germany, wolfgang.halang@fernuni-hagen.de

II. MEMORY SEGMENTATION

Software with malicious intentions often interferes with application programs or even with operating system routines in order to manipulate them for its destructive purpose or to deactivate software-implemented security functions. Here a memory segmentation measure as developed in [2] takes effect. It reliably prevents unauthorized accesses to the storage areas of operating system and application programs. To this end, a hardware-supervised segmentation of memory is introduced, which protects programs against modifications not permitted. The mass storage of a computing system must, accordingly, be partitioned into at least two segments. At least one of these segments has to be provided with a hardware-implemented write-protection to allow for the storage of safety-related programs and data such as operating system, utility programs and their databases, or fixed nominal values for operation and devices whose failure to be met could lead to the destructions of devices. As shown in Fig. 1, in further segments not protected this way data are stored which, according to experience, are subject to frequent changes. At the same time, these segments can be used to test programs. This protection needs to be ensured throughout all storage levels.

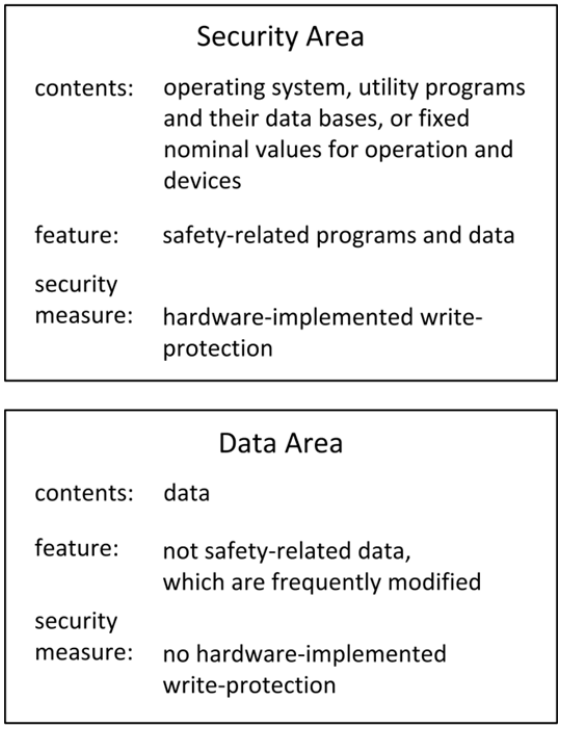

Figure 1. Hardware-supervised segmentation of memory.

The more than half a century old and still predominant *Von Neumann architecture* with its minimalistic principles is totally inadequate for systems that need to be safe and secure, as it does not separate *data* from *instructions* and, thus, does not permit to protect both kinds of information in an optimum way (see Fig. 2).

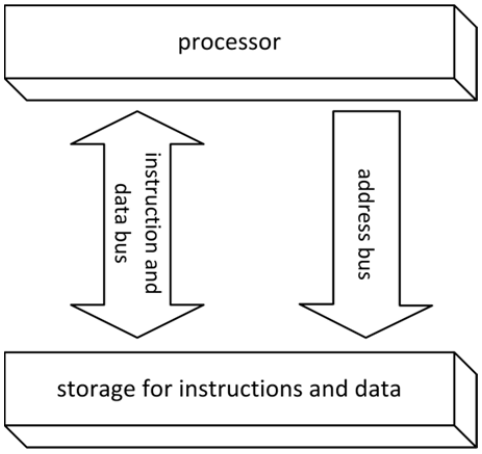

Figure 2. Von Neumann architecture.

The *Harvard architecture* (see Fig. 3), on the other hand, provides this separation throughout and, therefore, represents an adequate construction principle. It is a pleasant side-effect that systems based on this architecture are faster than the currently prevailing ones.

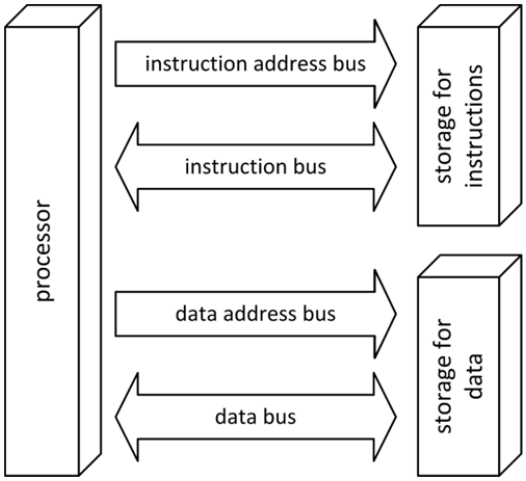

Figure 3. Harvard architecture.

III. CONTEXT-SENSITIVE MEMORY ALLOCATION

In contrast to programs, data are subject to frequent modifications. Therefore, a hardware-implemented write-protection as in [2] is not feasible for reasons of handling. Data can be protected against programs for spying out and modification, however, by a context-sensitive memory allocation according to [3], as shown in Fig. 4. Applying this measure, any unauthorised access to data is precluded. To this end, a system's mass storage, in particular the data area, is further subdivided by a partitioning into context-dependent segments. In an installation mode it is precisely specified which accesses to these segments are permitted to the programs. This is oriented at the data to be protected and not the programs, i.e. in general to each program there exist several data segments separated from one another.

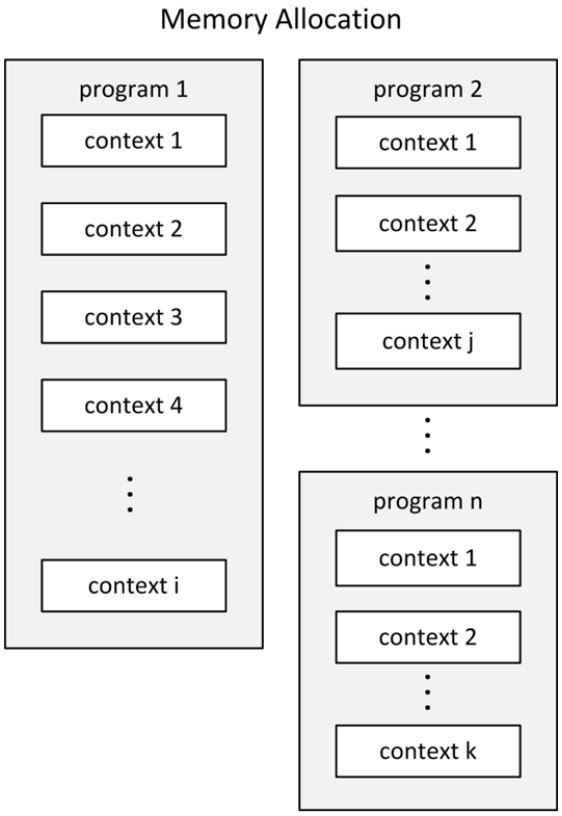

Figure 4. Context-sensitive memory allocation.

In other words, this method is characterised by permitting memory references to any application program and operating system service only by means of using access functions write-protected by hardware, which release the storage areas required for the respective application case for writing and reading or just for reading accesses. Accordingly, in a hardware-protected installation mode the users must establish for any program at least one access function, if they want to use this program in the application mode. As the protection mechanism shall not hinder the users in their daily work and, in particular, shall not hamper the systems' real-time behaviour, the bounds of the memory segments assigned to the different access functions are, for instance, stored in write-protected electrically erasable programmable read only memories (EEPROM), and loaded from there to control accesses to mass storage. Not all admissible memory areas are masked. It suffices to merely supervise the address lines and to control the write or read signals, respectively, of the mass storage media. If an access not permitted is requested, the processor is halted and a signal is generated, which allows the user to uniquely identify the incorrectly working program. For especially endangered programs a variety of access functions should be provided in order to keep the effects of infection by malware as low as possible. Electronic requests arriving from the outside, for instance, always ought to be placed first with their attachments, if any, in a separate and enclosed data area, and processed there.

This way, a spying or modification program that has infiltrated into a data segment without permission can be denied to spread to other segments leaving possible damage narrowly bounded. Based on the segmentation measures presented, a protection against unnoticed modification of data within such a segment can reliably be implemented by already established redundancy measures. Moreover, a finely structured segmentation also protects well against the negative effects of common programming errors, and provides a basis for lucid system maintenance.

IV. HARDWARE-IMPLEMENTED COUPLING OF WRITE-PROTECTION TO AUTHENTIFICATION AND AUTHENTIFICATION-DEPENDENT VIRTUAL ADDRESS SPACE

In order not to endanger the advantages of memory areas write-protected by hardware measures during the installation phases of programs, and to ensure separation on all storage levels throughout, it is necessary to accommodate service programs and their databases also in areas write-protected by hardware and separated from the program area. In doing so, it must be prevented that program and service areas are enabled for writing at the same time, and the memory management must be extended in such a way that the virtual addresses can be used to supervise the computer, since such addresses are linear and, thus, much easier to observe. This is achieved by utilising a hardware device according to [4] generating a write enable signal, which inherently prevents that more than one such signal is generated at a given instant. For this it is necessary to ensure a unique and safe authentification of the user, which is dependent on this person's momentary function, and by means of which the access rights required for the computer's protection are selected. This implies that these systems do not designate omnipotent administrators with rights, which cannot be controlled or are extremely difficult to protect, as they always proved to be a considerable weakness in a vast number of systems under different operating systems. Therefore, almost all attackers seek to gain administrator rights, in order to exercise complete control over a system. This possibility is constructively excluded in the here presented solution, since there is a kind of self-supervision of the correspondingly structured systems at any point in time. Expressed more precisely, hardware-protected and, thus, by software not attackable components of these systems control the respective other parts, even in installation phases. Suitable for user authentification are those methods which cannot be influenced by programs and which are, for instance, based on personal property or biometrical features. To supervise the address space of a computer, safe virtual addresses dependent on authentification and start addresses of page directories are used. The memory management unit is placed between storage and processor to protect the former against direct access by the processor. The unit is equipped with a hardware-implemented protection mechanism, which transmits the required programming signals of the processor in case of correspondingly privileged authentification, only.

V. DISCLOSURE OF REQUIRED RESOURCES

Destructive programs and software-based aggression from the Internet often use components of digital systems, which they would not need for their feigned nominal functions. Here the hardware-supported security measure detailed in [5] takes effect. For instance, for program modules responsible to receive electronic mail the access to communication components must be permitted, but not for those modules which display or even interpret the messages received. This example makes clear that to fulfill their nominal function programs do not need many of the available resources at all or, at least, not all the time, and that permanent release of all resources represents an irresponsible security risk, particularly as common programming errors without malicious intentions cannot be precluded for complex software. On the other hand, users cannot be expected to disable resources on a case by case basis, especially not for automation systems for which this is not possible at all. Therefore, measures need to be devised which protect the users, but do not unduely trouble or restrict them.

All these problems can be solved if any program, any interpretable file and any executable Internet content first discloses which resources it requires for execution. The disclosure of a program's nominal functions enables to install boundary values for systems and to supervise their operation in an effective way. By this supervision and, at any point in time, by locking all resources not needed at that time by means of hardware as shown in Fig. 5, it can be safely warranted that the desired nominal functionality is observed.

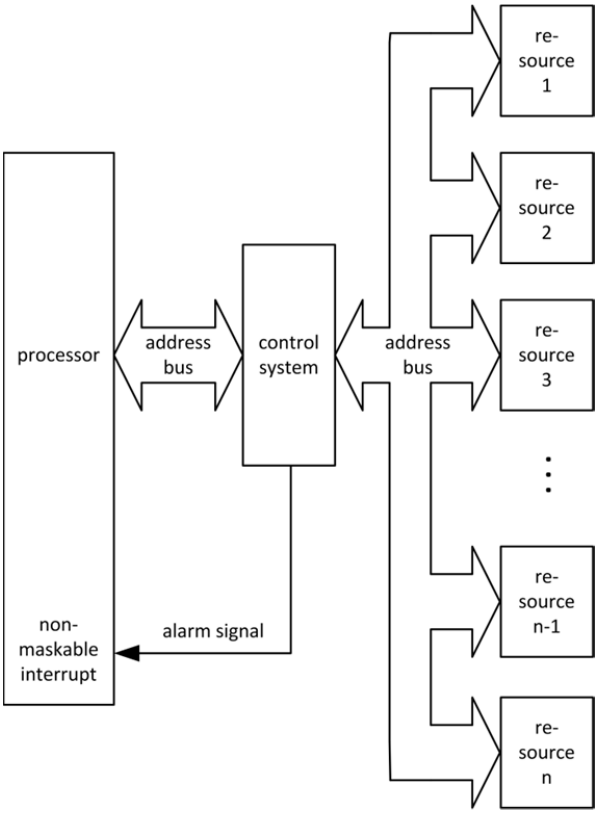

Figure 5. Hardware-controlled resources.

For, in installation modes, during which application software may not access processors, memory nor communication equipment, the users set the limits for resource accesses. Only after that it is possible to execute application programs under permanent hardware-supported supervision based on the constraints defined before. Hereby, not only the resource accesses are supervised, but also the execution times. Thus, the real-time capability is guaranteed. As a positive side-effect, this approach also prevents, up to a certain degree, damage caused by common programming errors without malevolent intentions. Upon deviation from its required nominal function the corresponding program is aborted. All resources seized before are reset and released again. This has the advantage that another program can immediately be put in execution after an illicit action, i.e. the system remains available.

The here presented methods solve the problem of executable Internet contents as well, which is currently of extreme urgency. For, executable Internet contents can be considered as programs for potential spying out and modification, whose program code resides on remote computers. To cope with them, the following procedure is to be adhered to.

- Before a program stored on a different, remotely located computer may become active, it must first provide information about its nominal functionality and the resources required for this.
- If the intended activities are considered uncritical, the execution is initiated without asking the users unnecessary questions. What hereby is regarded as uncritical was defined before by the respective users themselves, and stored in an area write-protected by hardware. Since a program's alleged activities are securely supervised in any case, the credibility of communication partners is not of such a decisive importance for a computer's security as it is the case for the currently prevailing solutions. Confidential information as exchanged, for instance, in electronic commerce is secured and encrypted for transmission here as well.
- If the data disclosed indicate critical functions, the further proceeding depends on whether there is already a certain trust in the source of the data, and which actions were permitted to it. In case the actions requested are within the framework already authorised, also here there is no feedback to the users. If the range of actions of an application or a data source is to be extended, first the users disconnect the communication links to extend the conceded framework of actions, and resume the connection to the communication partners not before the supervisor data and all resources not required have been hardware-protected against unauthorised access.

This solution by far outperforms established methods such as, for instance, the trust-based one of "ActiveX" or the "sandbox" method of "Java", as decisions can be made on the basis of a much finer granularity, without imposing on the users unreasonable restrictions or urge them to admit everything.

VI. CONCLUSION

To be secure, automation systems must fulfill the following requirements.

- Data and instructions have to be separated throughout.
- Authentifications may not be influenceable by software.
- Protection systems may not be attackable themselves. This means that their implementation must be proven correct and safely protected against modifications not permitted.
- The protection of systems may not be put out of effect during the installation phases of application programs or of operating system components as well.
- All storage levels (main memory, mass storage etc.) have to be protected throughout against unauthorised accesses by means of authentification-dependent virtual address spaces.
- Constraints and nominal functionalities of programs are defined in installation phases, and permanently supervised in the course of operation. Their observance is guaranteed even under real-time conditions.
- To protect data against effects of common program errors or malicious interpretable files and to enable context-sensitive memory allocation, a means for the instantiation of programs must be provided, which employs access functions.

Utilising the presented measures industrial computer control systems are effectively protected against inadmissible accesses. This holds in particular for still unknown attack patterns or malware, too, because there is no more need for databases of malicious code or attack prototypes, which become obsolete within hours anyway due to the swift spreading of current malware via the Internet. It has been shown that it is possible to build systems which are immune against intruders and espionage. In addition, it was shown that separation and structuring considerably facilitates the maintainability of computer control systems, too, and even increases their performance. Furthermore, it became clear that systems protected by the above mentioned measures exhibit, on the basis of disclosing their nominal functions, of the permanent supervision against set bounds, of the context-sensitive allocation of data and of the impossibility to attack operating systems and application programs, a degree of *robustness* which allows them to maintain their functionality despite some failing application programs – a property being of fundamental importance for automation systems and highly safety-critical applications.

The measures presented here guarantee, with reference to [7], the observance of the *protection objectives*

1. **Privacy:** unauthorised gain of information is made impossible, i.e. spying out of data is obviated,
2. **Integrity:** unauthorised modification of information is precluded,
3. **Availability:** unauthorised influence on the functionality is precluded and
4. **Attributability:** at any point in time the responsible persons can be identified with certainty.

REFERENCES

[1] "Der Schädlings-Flut Herr werden", *Bundeswehr aktuell*, 48(4)5, 30 January 2012, on-line: s337251796.online.de/2012/KW4/html/10005.html

[2] W.A. Halang and R. Fitz, "Speichersegmentierung in Datenverarbeitungsanlagen zum Schutz vor unbefugtem Eindringen", German patent registration DE10031212A1, 2000

[3] W.A. Halang and R. Fitz, "Kontextsensitive Speicherzuordnung in Datenverarbeitungsanlagen zum Schutz vor unbefugtem Ausspähen und Manipulieren von Daten", German patent registration DE10031209A1, 2000

[4] W.A. Halang and R. Fitz, "Gerätetechnische Schreibschutzkopplung zum Schutz digitaler Datenverarbeitungsanlagen vor Eindringlingen während der Installationsphase von Programmen", German patent 10051941 since 20 October 2000

[5] W.A. Halang and R. Fitz: "Offenbarendes Verfahren zur Überwachung ausführbarer oder interpretierbarer Daten in digitalen Datenverarbeitungsanlagen mittels gerätetechnischer Einrichtungen. German patent registration DE10055118A1, 2000

[6] R. Langner, "Stuxnet: Dissecting a Cyberwarfare Weapon", *IEEE Security & Privacy*, 9(3)49–51, 2011

[7] K. Rannenberg, A. Pfitzmann and G. Müller, "Sicherheit, insbesondere mehrseitige IT-Sicherheit", in: *Mehrseitige Sicherheit in der Kommunikationstechnik*, pp. 21–29, Bonn: Addison-Wesley 1997.

Robert Fitz, born 1965 in Villingen-Schwenningen, Germany, received BSc and MSc degrees in electronic engineering in 1988 and 1999, respectively, and was awarded Dr.-Ing. in electrical and computer engineering in 2001. After working in industry and academia, he was appointed professor of electronic engineering and computer science in 2002. His current research interests are systems on chip as well as availability and security of data processing systems.

Wolfgang A. Halang, born 1951 in Essen, Germany, received doctorates in mathematics (1976) and computer science (1980). After working in industry and academia, he was appointed chair of computer engineering and department head at the University of Groningen in the Netherlands, and 1992 at Fernuniversität in Hagen. He was co-director of the 1992 NATO Advanced Study Institute on Real-Time Computing and visiting professor in Maribor, Slovenia, and Rome, founded the journal Real-Time Systems, is member of four further journals' editorial boards, (co-)authored 40 books and 400 refereed publications, holds 20 patents, gave 80 guest lectures worldwide, and is active in various technical committees and 240 programme committees.