Analysis of Informed Attacks and Appropriate Countermeasures for Cyber-Physical Systems

Francesca Saglietti, Matthias Meitner, Lars von Wardenburg, Valentina Richthammer

Software Engineering (Informatik 11)
University of Erlangen-Nuremberg
Martensstr. 3, 91058 Erlangen, Germany
{francesca.saglietti, matthias.meitner, lars.wardenburg}@fau.de,
vs.richthammer@studium.fau.de

Abstract. Based on considerations about the knowledge required to carry out different types of network attacks, this article discusses the logical demands posed to the attacker in order to circumvent the most classical checks for message trustworthiness. In view of the limitations of existing avoidance and detection techniques, the article stresses the need for targeted testing strategies aimed at the identification of exploitable code vulnerabilities. For this purpose, it proposes a paradigm for the generation of intelligent test cases meant to maximize the chances of anticipating challenging scenarios during early verification phases.

Keywords: Cyber-attack, informed attack, level of knowledge, communication constraints, confidence constraints, control constraints, arithmetical overflow, buffer overflow, testing.

1 Introduction, Motivation and Intention

It is well-known that the threat of IT attacks is continuously increasing and provoking high losses due to non-productivity and maintenance costs [1]. In case of safety-relevant information the effect of IT threats goes beyond economic concerns as soon as they may severely affect the possibility of timely intervention on physical entities requiring urgent support – be it the case of hospital patients crucially requiring medical treatment or of critical technical processes subject to appropriate automatic control, as is the case in industrial plant control or car-to-car communication.

For several reasons, including distance, distribution and logistics, the communication between controlling and controlled entities typically makes use of increasingly complex networks joining sensors with computer nodes via corresponding terminals over which the operators may need to update relevant plant configuration data. Even in case of proprietary networks the connection complexity and the user facilities of such architectures complicate the task of analysing the risk of potential misuse and of excluding severe IT attacks [2].
The intention of the present article is to address this problem in a systematic way by considering the chances and the impact of attacks dependent on the amount of application-specific knowledge available to potential attackers and on the complexity of the underlying logical constraints. With respect to the expertise required in the application and control domain considered, the following levels of application- and control-specific knowledge resp. information may be distinguished:

- a mere network-specific insider knowledge is likely to suffice in order to identify the location of communication media, to perceive the transfer of bit streams, or to access protocol meta-data (in the following denoted as “low level”, s. section 2);
- additional insight concerning the technical process to be controlled and the envisaged control system behaviour is required in order to allow for a meaningful interpretation of messages (in the following denoted as “medium level”, s. section 3);
- full information is available to insider attackers in case they additionally know the software-based control system including potential code vulnerabilities (in the following denoted as “high level”, s. section 4).

The levels considered are summarized in Table 1.

<table>
<thead>
<tr>
<th>levels of network, application, control knowledge</th>
<th>knowledge / information domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>high</td>
<td>cable location</td>
</tr>
<tr>
<td>medium</td>
<td>bit streams</td>
</tr>
<tr>
<td>low</td>
<td>protocol and protocol meta-data</td>
</tr>
<tr>
<td></td>
<td>process behaviour and data</td>
</tr>
<tr>
<td></td>
<td>control system behaviour and specification</td>
</tr>
<tr>
<td></td>
<td>control system code and potential vulnerabilities</td>
</tr>
</tbody>
</table>

Especially in view of the classes of insider attacks involving the highest level of insight, the present article aims at evaluating strengths and weaknesses of existing avoidance and detection counter-measures. Based on the identified limits of the state-of-the-art, the article will successively focus on the need and on the chances of developing dedicated intelligent testing strategies targeted at optimizing the chances of anticipating informed attacks during a preliminary security-based verification phase (s. section 4).
2 Network Attacks based on Network Knowledge

Evidently, the term “cyber-physical systems” refers to a wide range of applications which may vary in terms of several attributes. Common to them is the inclusion of one or more physical processes communicating via sensors and actuators with one or more logical units. Depending on several attributes, among them number, distribution and distance of communicating entities, networking complexity and criticality this may give rise to a wide range of patterns. In order to ease the analysis of attacks, the following considerations will focus on a simplified, low-sized scenario consisting of the following entities (see Fig. 1):

- a physical process (e.g. the production process of an industrial plant) sending information on its current physical state via sensors and subject to changes initiated by actuators;
- a computer-based control system receiving and processing the process information sent by the sensors to identify whether an intervention is needed; in this case, a corresponding message is sent to the process actuators;
- a communication network supporting message passing between the physical process and the control system;
- one or more human(s) able to influence the communication between the senders and the receivers.

Even if it limits the following considerations to a mere quadruple of communicating agents, the proposed analysis does not restrict generality to an unacceptable degree, as any network attack over a – however complex – net topology will involve misuse along at least one communication edge; this makes it reasonable to consider such a “one-edge-attack” in more detail.

The potential of communication misuse depends on the level of knowledge of the attacker(s): evidently, very simple attack scenarios not requiring any plant knowledge nor network access, e.g. cable disruption or magnetic influence on communication cables, typically result in permanent or temporary communication disturbances.
usually resulting in service interruption. As such actions may jeopardize operation and even safety, they must be prevented by physical protection measures.

More subtle than the physical attacks just considered are actions attempting at the simulation of regular communication between process and software. Attackers enabled to access the network and familiar with protocol meta-data, for example, may gain further knowledge regarding communication frequency and topology by educated guesses. The resulting information may enable them to remove, insert or modify relevant messages without violating the protocol. For example, by copying relevant message portions and resending them at regular frequency, the attackers may falsify process information for the purpose of preventing the control system from taking appropriate decisions.

Such attacks go beyond threatening classical information security values [3], as they target inappropriate, possibly highly unsafe physical states by mere intervention on the communication medium and without necessarily jeopardizing stored data or service availability.

On the other hand, whether interfering by physical action or by ad hoc message manipulation, the uninformed attacks considered so far are usually easily recognizable by standard control software capable of identifying missing or semantically irregular message streams. Depending on the reliability and availability demands of the application considered, such control systems have to initiate upon detection proper reactions like operator alerts or plant shutdowns.

3 Network Attacks based on Application / Control Knowledge

3.1 Classification of Constraints on Attacks

As mentioned in the previous chapter, intelligent control systems must be designed to analyse data trustworthiness in order to identify message manipulation. This section is devoted to an analysis of this attack under the additional assumption that the attacker(s) are application experts fully informed

- on the technical process under control, as well as
- on the behaviour of the digital control system in use.

More precisely, in this section the attacker is not (yet) required to know about the details of the internal logic of the digital control system, but possesses knowledge about the control system specification, i.e. about the software response to the messages it receives. In addition to this expertise, the attackers are also assumed to be able to

- legally access the network over which the technical process under control communicates via sensors with its control system to inform about its current physical status and on the need of intervention via actuators;
- interpret the semantics of the message(s) sent over the network;
- read, remove, modify and insert messages.
In this case the attacker may try to influence the control system behaviour by manipulating the stream of messages such as to simulate a world different from the real one and thus to trigger a controlling intervention different from the one really required. In order to achieve this goal, the attacker is forced to make sure that neither the network nor the control system is able to identify the manipulation of the message stream as such (be it by copies, deletion or modification of single messages).

To achieve this, the sequence of messages sent over the network has to fulfil three different requirements coincidentally:

- first, it must not violate any of the constraints posed by the network to ensure regular communication (in the following denoted as *communication constraints*);
- moreover, it must not violate any of the constraints required by the control software in order to consider the input data as trustworthy measurements (in the following denoted as *confidence constraints*);
- finally, it must fulfil data constraints [2] (in the following denoted as *control constraints*) such as to impose to the control system the incorrect behaviour envisaged by the attackers, either by simulating the occurrence of relevant fake events or by masking the occurrence of relevant real events.

In the following, the three classes of constraints just introduced are further analysed and refined in more detail:

**Communication Constraints.** Usually, regular message communication between real-world and computer is subject to constraints of various nature which may be further classified as follows:

- *constraints on protocol meta-data* refer to information characterizing the transport of process data over a communication medium, like message origin (sender), message destination (receiver), action intended to be carried out on message data upon reception (e.g. read / write), communication correctness (redundancy bits);
- *constraints on message stream* refer to information characterizing legal message sequences and may concern time intervals between the arrivals of serial messages as well as the order of consecutive messages;
- *constraints on message structure* refer to information characterizing the syntax of messages as defined by the input specification of the control software.

As soon as one or more communication constraints are ostensibly violated, the underlying message-passing can be regarded as affected by intentional or accidental events reducing or annulling the trustworthiness of the information transported. In such cases, the control system will initiate a predefined countermeasure like an operator warning or an automatic shutdown.

**Confidence Constraints.** A further category of constraints addresses the actual process data contained in the message(s); the following constraint classes can be distinguished:

- *validity constraints* relate to information characterizing the acceptability of sensor measurements in terms of their lying within predefined physical ranges corresponding to the instrument accuracy;
• **consistency constraints** refer to information characterizing the level of agreement between redundant sensor measurements required to consider them as sufficiently accurate;

• **plausibility constraints** refer to information related to the quantitative relationship between valid measurements of different process variables connected by inherent physical dependencies.

As soon as one or more confidence constraints are violated, the process measurements received can be regarded as irregular; this indicates that some anomalous event takes place which requires a timely counteraction. On the other hand, as long as all communication constraints are fulfilled, there is no reason for the control system to mistrust data accuracy.

**Control Constraints.** As already mentioned, upon fulfilling communication and confidence constraints, the attack target may be two-fold:

• on the one hand, attacker(s) may intend to simulate anomalous process behaviour such as to initiate countermeasures which under regular circumstances may reveal wasteful at the best, but potentially hazardous by contributing to destabilize the balance of the technical application supervised;

• on the other hand, they may want to simulate regular process behaviour while manipulating or even sabotaging the technical process such that any safety-related intervention required to be immediately initiated by the automatic control system may be crucially delayed or jeopardized for an indefinite time.

In both cases the attackers need to consider so-called control constraints whose fulfilment characterizes regular process behaviour and whose violation characterizes anomalous process behaviour. Typical examples of control constraint classes are:

• **constraints on the value of a single process variable** refer to information characterizing the range of a particular process variable during regular operation;

• **constraints on a tuple of values of different process variables** refer to information characterizing the combined ranges of several process variables; combined constraints may be more stringent than individual ones as they address the coincidental approach of boundaries by different process indicators;

• **trend constraints** refer to information characterizing regular time-dependent evolution of any process variable as indicated by consecutive messages.

### 3.2 Examples

The constraint classes and types are summarized in Table 2 together with a few typical examples illustrating them.

### 3.3 Avoidance and Detection Techniques

Among the major countermeasures classically applied for the purpose to avoid network attacks is *encryption* which undoubtedly can significantly contribute to reduce the chances of cyber-criminal actions in general. Signing signals at their source can increase data integrity as far as the encryption technique can be considered
as secure. With respect to informed attacks, however, encryption is likely to find its limits in the need to allow insider network users to intervene on communication by legal updating of relevant process parameters. This may require to allow them access to encryption information.

Table 2. Constraint classes and types illustrated by examples, partly inspired by [4]

<table>
<thead>
<tr>
<th>constraint classes</th>
<th>constraint types</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>communication</td>
<td>protocol meta-data</td>
<td>Profibus meta-data referring to source and destination address, function code (e.g. request or send/request, station type)</td>
</tr>
<tr>
<td></td>
<td>message stream</td>
<td>minimal time / maximal time between consecutive messages; sequence number of previous message incremented by 1 modulo memory size of the counter</td>
</tr>
<tr>
<td></td>
<td>message structure</td>
<td>header, sequence number, sensor_data_1, ... sensor_data_n, checksum, trailer</td>
</tr>
<tr>
<td>confidence</td>
<td>validity</td>
<td>$S_{min} \leq S \leq S_{max}$ where $[S_{min}, S_{max}]$ denotes the accuracy range of the sensing instrument</td>
</tr>
</tbody>
</table>
|                    | consistency                       | IF $S_1 \geq 0.95 \cdot S_{max}$ average {$S_1, S_2, S_3$}  
IF $S_1 \geq 0.975 \cdot S_2 \wedge 1.05 \cdot S_1 \leq S_3$ average {$S_1, S_2$}  
IF $S_1 \geq 0.975 \cdot S_3 \wedge S_1 \leq 0.95 \cdot S_3$ average {$S_2, S_3$} |
|                    | plausibility                      | thermodynamic dependencies between steam level and pressure |
| control            | single boundary                   | $P \leq P_{max}$ where $[0 ; P_{max}]$ denotes the pressure range for regular operation |
|                    | combined boundary                 | $P \leq 0.9 \cdot P_{max}$ and $L \leq 0.9 \cdot L_{max}$ |
|                    | trend                             | $\Delta P \leq \Delta P_{max}$ where $\Delta P$ denotes the rate of change of $P$ by quadratic interpolation |

The constraints considered and classified above contribute to filtering the communication messages in order to identify corrupt communication and sensing. The more complex and extensive the constraints and corresponding filtering techniques, the more difficult it is for the potential attacker(s) to carry out...
unidentifiable attacks. In addition, network decentralization also contributes to prevent attacks by restricting the access of potential attackers to only a part of the relevant message information. In this case the scope of attack of each attacker would be limited such that the fulfilment of consistency and plausibility constraints would require the synchronized action of several attackers.

4 Network Attacks based on Code Knowledge

In this section it is assumed that – in addition to the levels of knowledge considered so far – the attacker(s) also possess full information about the code inside the control system. This applies for example in case the network users have been previously involved in code development when they may have had opportunities to identify or insert code portions potentially increasing system vulnerability.

4.1 Overflow-based Network Attacks

Typical sources of code vulnerability are overflows occurring when data is greater than actually storable or data size is greater than supported by a given buffer; typical examples are the following cases:

Arithmetic Overflow. Arithmetic overflows occur when (intentionally or unintentionally) computer calculations result in values which are out of the permitted range. The effects are false values for relevant variables possibly resulting in dramatically incorrect behaviour. On the other hand, arithmetic overflows must not be banned in general, as they may provide suitable solutions for specific problems like the management of circular buffers.

Stack Buffer Overflow. In case of a stack buffer overflow a program stores part of its data inside the call stack, but outside of the specified memory area reserved. This may result in overwriting
   • a variable residing close to the buffer, or
   • the return address by a new pointer. This is particularly critical if the new pointer leads to the execution of malicious code previously inserted, possibly after traversing no-op instructions intended to close the gap between the jumping point and the start of the shellcode.

Further overflow variants involve similar considerations for heap buffers or for combinations of overflow types, e.g. in case of arithmetic overflows of an integer variable meant to determine the size of the free buffer space left.

4.2 Overflow Avoidance, On-line Checks and Static Analysis

Overflow effects such as those mentioned above are meanwhile well-known; for the purpose of their avoidance and detection, a number of existing techniques reveal different strengths and weaknesses.

Avoidance Techniques. During development, constructive techniques can help avoid the problem by making use of particular languages like Java or C# which prevent
buffer overflows from occurring by providing mechanisms for checking buffer boundaries. For performance and licensing reasons, however, programming language prescription may represent an unacceptable restriction. Buffer overflows can also be avoided by exclusive use of safe library functions including boundary checking instead of unsafe functions like strcpy() or gets().

**Online Detection Techniques.** Countermeasures preventing the execution of code residing on a stack are supported by some operating systems based on so-called executable space protection (ESP). A further countermeasure for online overflow detection is provided during compilation by tools which, like StackGuard [5], reorganise the stack by including a so-called canary value just before the return address. After any function call, this value is checked for any modification. Attackers aiming at overwriting the return address must also overwrite the canary value. If a modification is detected, the program is usually terminated. This and all other countermeasures based on dynamic checks are evidently limited by the execution time overhead involved.

**Static Analysis Techniques.** During early verification phases, different analysis techniques may be applied. Among the static ones, there are techniques targeted at the early identification of unsafe functions by means of appropriate tools [6]. More sophisticated techniques analyse the data flow for the purpose of identifying overflow hazards; among them, integer range analysis [7] maps the problem of buffer overflow identification onto an integer constraint problem. Although beneficial, such approaches are doomed to incompleteness, as they can only address statically identifiable overflow causes.

### 4.3 Overflow Detection by Testing Techniques

The limitations of the state-of-the-art identified above reveal the need for novel testing techniques targeted at the early detection of exploitable vulnerabilities like overflows. Depending on the control and data flow complexity of the program considered such testing techniques may rely on analytical reasoning or require heuristics-based approaches. Their potential is the focus of the authors’ ongoing investigations.

Whether systematic or random-based, such approaches are aimed at the simulation of intelligent attackers operating under an application- and network-specific profile. Intelligent test data must be generated such as to maximize the chances of activating code vulnerabilities under the given profile conditions.

The underlying mathematical problem to be addressed involves multi-objective optimization pursuing the following targets:

- **target 1:** in order to prevent an early attack detection, test data must fulfil all network-specific communication constraints as well as all application- and control-specific confidence and consistency constraints (see section 3.1); this target may be systematically achieved by means of constraint-based test data generation techniques;
• target 2: in order to provoke an overflow, test data must maximize the chances of enabling writing operation(s) outside of the allocated memory space (see section 4.1);

• target 3: in order to support both search width w.r.t. target 2 and test confidence in case of unsuccessful search, test data must maximize a predefined control or data flow coverage measure.

On the whole, a solution to this multi-criteria optimization problem would provide either a meaningful message stream actually capable of exploiting a code vulnerability or at least significant quantitative evidence against the potential of such a class of attacks by means of objective measures reflecting the amount and the coverage of secure behaviour observed.

Concerning testing targeted at reliability-rather than security-based properties, similar test data generation problems were successfully approached by genetic algorithms. Such heuristics revealed as useful in many testing contexts, including unit testing based on control and data flow coverage [8], integration testing based on component interaction coverage [9], statistical testing based on operationally representative coverage of component interactions [10] as well as testing of cooperating robots based on agent interaction [11] and operational coverage [12].

It is well-known that genetic algorithms consist of stepwise generating successive populations of individuals based on a given initial population. Successive generation are derived by

• evaluating each individual of the current population by means of a fitness function reflecting the degree of fulfilment of the target-based criteria;

• applying genetic operators (like selection, recombination and mutation) on the current individuals based on the fitness values obtained as well as on predefined probabilistic parameters.

This iterative process is concluded as soon as the criteria are considered as acceptably met, at the latest after a maximum number of iterations or even after a given number of iterations lacking any meaningful improvement. Genetic algorithms involve a wide range of variants, e.g. concerning elitism strategies applied to transfer the fittest individuals to the next generation in order to support monotonic improvement, while ensuring at the same time the enrichment of genetic material by mutation.

For the particular case of security-targeted testing considered in this section, the fitness of an individual, i.e. of a message stream, is captured by a measure reflecting the degree of its meeting the target criteria. Target 1 addresses a must criterion which may be constructively enforced such as not to require any particular fitness evaluation procedure. Target 3, on the other hand, addresses maximal code coverage; depending on the coverage metric considered, this may allow for an absolute fitness measurement, or at least for an evaluation of the relative improvement of a population when compared with the previous one. Finally, target 2 addresses the exploitation of a potential vulnerability, i.e. the fulfilment of a condition of unknown satisfiability. Therefore, the definition of an appropriate fitness measure is not straightforward. An option currently under investigation consists of measuring the distance of the current buffer fill level to a (lower or upper) buffer boundary and in rewarding individuals approaching such a boundary.
5 Conclusion

This article analyzed the impact of network attacks classifying them in terms of the level of network, application, control or code knowledge required; hereby, it focused on the highest levels of knowledge which assume extensive expertise in the application-specific domain, in process control and, in particularly challenging situations, even in the details of the logic residing in the control system code.

On the basis of a variety of informed attacks, the article elaborates on the logical demands posed to the attacker in order to circumvent the most classical checks for message trustworthiness.

Finally, the article considers the state-of-the-art in prevention and detection techniques by comparing strengths and weaknesses of a selection of existing constructive and analytical approaches. As a result, it stresses the need for more targeted testing techniques aimed at an early detection of exploitable vulnerabilities; for this purpose, it proposes a paradigm for the generation of intelligent test cases meant to maximize the chances of anticipating challenging scenarios during early verification.

6 Acknowledgment

The authors gratefully acknowledge that a major part of the work presented was supported by the German Federal Ministry for Economic Affairs and Energy (BMWi), project SMARTTEST. The project is carried out in cooperation with the partner institutions University of Magdeburg, University of Applied Sciences of Magdeburg-Stendal and AREVA GmbH. In particular, the authors thank Robert Fischer und Robert Clausing for inspiring discussions.

7 References


