



## Security Discrepancies

### *The suppressed realities*

Jürgen Pabel, CISSP

Security has become a hot topic in recent years. It has become such a hot topic, that it's almost omnipresent. Unfortunately, this also has negative effects, in that security has become a sales argument and the marketing effect has hollowed it of its meaning. Security is nowadays commonly perceived as a functional idiom: a product is secure, if it features security mechanisms. What's commonly ignored, are the technical aspects: whether a product's security mechanisms are actually effective. Yes, any difference between the two should merely be implementation flaws – but, one must be aware of this discrepancy, and that it will only continue to expand, due to the increasing complexity of Information Technology.

Just only a few years ago, one had to master both networking protocols and the internals of computer systems, in order to successfully attack networked computer systems. With the advent of scripting languages in the server environment, it became possible to compromise systems with only limited expertise. Surely, those languages have largely eliminated the dominant vulnerability classes of other implementation options (like buffer overflows – still present in most of today's software), but they generally allow for exploitation on a higher level. Therefore, exploiting today's environments is conceptually easier than in the past – the best examples for this are web applications: unfortunately, many still encode all state data inside the requests – which are entirely under the user's control. Attackers often manipulate those parameters, in order to gain access to otherwise inaccessible data, obtain other users' identities, or even run commands of their own liking on the server. Currently, there's no simplistic resolution to these issues, as these environments lack more resilient alternatives – well, at least there's a lack of more resilient alternatives, that have been accepted by the general software development industry. Resolving this issue isn't going to be easy: software developers in general like development environments, that provide them with all the necessary “freedom” to express their ideas – but, often it's exactly this “freedom”, that allows attackers to penetrate the protective measures. Expertise and awareness to potential security issues are the only effective countermeasures, which are increasingly hard to master, because the dominant server software environments are simplistic in style, but very complex in terms of capabilities.

Although security has been a major design aspect in emerging software environments,



it's going to take time to migrate away from ancient paradigms. Recent evolutions have centered on enhancing security in technical realms, what has largely been left unchanged is the ancient structure of software development: security mechanisms are implemented by developers, not provided by the environment<sup>1</sup>. Relieving application developers from having to provide their own technical security framework will represent a leap towards better security. Until then, we'll surely see an increasing number of security vulnerabilities of the “oh no – that can't be true” kind.<sup>2</sup>

---

1 Please refer to “Calling for a revolution” for further discussion of this topic:  
[http://labs.akkaya.de/pub/ACLabs/Publications/Revolution\\_en.pdf](http://labs.akkaya.de/pub/ACLabs/Publications/Revolution_en.pdf)

2 I won't discuss increasing liability for software as a solution here, as it's not a technical issue. Please refer to my upcoming book, “IT-insecurity”, for further discussion of this topic:  
<http://www.it-insecurity.com/>