The Case for System Command Encryption

Marc Beunardeau, Aisling Connolly, Rémi Géraud, David Naccache
Ecole normale supérieure, Paris France
given_name.family_name@ens.fr

ABSTRACT

In several popular standards (e.g. ISO 7816, ISO 14443 or ISO 11898) and IoT applications, a node (transponder, terminal) sends commands and data to another node (transponder, card) to accomplish an applicative task (e.g. a payment or a measurement).

Most standards encrypt and authenticate the data. However, as an application of Kerckhoffs' principle, system designers usually consider that commands are part of the system specifications and must hence be transmitted in clear while the data that these commands process is encrypted and signed. While this assumption holds in systems representable by relatively simple state machines, leaking command information is undesirable when the addressed nodes offer the caller a large "toolbox" of commands that the addressing node can activate in many different orders to accomplish different applicative goals.

This work proposes protections allowing encrypting and protecting not only the data but also the commands associated to them. The practical implementation of this idea raises a number of difficulties. The first is that of defining a clear adversarial model, a question that we will not address in this paper. The difficulty comes from the application-specific nature of the harm that may possibly stem from leaking the command sequence as well as from the modeling of the observations that the attacker has on the target node's behavior (is a transaction accepted "is a door opened" is a packet routed etc). This paper proposes a collection of empirical protection techniques allowing the sender to hide the sequence of commands sent. We discuss the advantages and the shortcomings of each proposed method. Besides the evident use of nonces (or other internal system states) to render the encryption of identical commands different in time, we also discuss the introduction of random delays between commands (to avoid inferring the next command based on the time elapsed since the previous command), the splitting of a command followed by n data bytes into a collection of encrypted sub-commands conveying the n bytes in chunks of random sizes and the appending of a random number of useless bytes to each packet. Independent commands can be permuted in time or sent ahead of time and buffered. Another practically useful countermeasure consists in masking the number of commands by adding useless "null" command packets. In its best implementation, the flow of commands is sent in packets in which, at times, the sending node addresses several data and command chunks belonging to different successive commands in the sequence.

From an applicative standpoint, the protected node must be designed in a way that prevents, as much as possible, the external observation of the effects of a command. For instance, if a command must have an effect expressible within a time interval $[t_a,t_b]$ then the system should randomly trigger the command's effect between t_a and t_b. All the above recommendations will be summarized and listed as "prudent applicative design principles" that system designers could use and adapt when building new applications.

A starting point for a theoretical framework for reasoning about generic command learning attacks may be the following: Let $A(f, x)$ denote the set of states through which the machine passed while performing a protocol f run on input x. Let H denote entropy and define discretion(f) = $H(A(f,x) \mid x)$. We define the protocol f as *more discreet* than f' if discretion(f) > discretion(f'). Discretion is a very interesting metric because of its *independence of any specific attack*. Put differently, increasing discretion secures a system against attacks that…*haven't been invented yet!.* Denote by $f \sim f'$ two alternative protocols[1] computing y. Let $O(f)$ denote the time complexity of f. f is *optimally discreet* if $f' \in O(f)$ and $f \sim f' \Rightarrow$ discretion(f) ≥ discretion(f'). Hence, a prudent system engineer must always use optimally discreet protocols. Classifying existing network protocols according to this criterion is, to the best of our knowledge, new. We propose to examine with Huawei several popular protocols and evaluate their discretion. A very interesting line of research will be the design of parameterized algorithms f_w where both $O(f_w)$ and discretion(f_w) are increasing functions of w. This may concretely measure the price at which extra calculations could buy extra security (discretion) and the marginality ratio function discretion(f_w)/log($O(f_w)$) will reflect the number of discretion bits bought at the price of one work-factor bit.

CCS Concepts/ACM Classifiers

• Systems security, Distributed systems security

Author Keywords

Embedded security; encryption; commands; cyber-physical.

BIOGRAPHY

D. Naccache heads the ENS' Information Security Group. His research areas are code security, the automated and the manual detection of vulnerabilities. Before joining ENS he was a professor at UP2. He previously worked for 15 years for Gemalto, Oberthur & Technicolor. He is a forensic expert by several courts, and the incumbent of the Law and IT forensics chair at EOGN. www.ens-paris.fr.

HRI 2017, February 25–March 1, 2017, Portland, OR, USA.
ACM ISBN 978-1-4503-4335-0/17/02.
http://dx.doi.org/10.1145/2998181.3020283

[1] While f and f' output the same y, the condition $\forall x, f(x)=f'(x)$ does not capture the notion $f \sim f'$ because f and f' use *different inputs.* This can be rigorously fixed by introducing two *input extraction* algorithms that "distill" x and x' from the machine's initial state.