

Software Security, Privacy, and Dependability

Metrics and Measurement

George Hatzivasilis and Ioannis Papaefstathiou,
Technical University of Crete

Charalampos Manifavas, Rochester Institute of Technology Dubai

// A practical, easy-to-use methodology measures a software system's overall security, privacy, and dependability on the basis of the standards for each of these properties. It determines the system's attack surface and the effectiveness of the implemented protection methods. //



SOFTWARE MEASUREMENT IS important in risk management during software development.¹ For example, NASA applies software inspection throughout development to corroborate design and requirements fulfillment and to enhance product


dependability, especially in mission-critical systems.² Measurement techniques that comply with the relevant standards help indicate the validity of the solution and the business.³

In an industrial setting, it's imperative to develop measurement

methods that evaluate many attributes at the same time and present useful information about the overall system. For example, Ford has established a research methodology for modeling automotive software security, privacy, usability, and reliability.⁴ This methodology analyzes the software functionality on the basis of these properties and assigns qualitative values (low, medium, or high) to each one. Ford has applied this methodology to real automotive software, such as for an antilock braking system and a valet key.

One particularly useful approach is to measure software's *attack surface*, which indicates its susceptibility to attack. Examples of metrics for this include the attack surface metric (ASM) and the Relative Attack Surface Quotient (RASQ). (For more on ASM and RASQ, see the sidebar.) However, up to this point, attack surface metrics such as these have considered only security, not privacy or dependability. Moreover, these metrics reveal only a system's attackability, not its protection level.

To alleviate this situation, we propose a multimetric methodology to evaluate and quantify software system security, privacy, and dependability (SPD). Our SPD evaluation is based on the ASM, but we've enhanced the security specification on the basis of the Common Criteria Evaluation Methodology (CEM)⁵ and *Open Source Security Testing Methodology Manual*.⁶ For privacy, we employ ISO/IEC standards 29100⁷ and 27018⁸ and the European Data Protection Directive 95/46/EC. We derive dependability from IEC standard 60300.⁹ We've also enhanced the basic surface calculation with a risk analysis of the



applied protection mechanisms' effectiveness. The metric calculates the real protection level of the potentially attackable points.

The SPD Multimetric

The SPD multimetric employs two methodologies. First, we determine the SPD surface, similarly as in the ASM and RASQ. At this point, we perform relevant security analysis employing those two metrics, extended by the privacy and dependability perspectives. The results reveal the system's attackability and the potential damage.

Second, we employ systematic risk assessment to determine the effectiveness of the system's protection mechanisms. The overall analysis estimates the real protection level by aggregating the risk analysis of the attackable points and the protection mechanisms' effectiveness.

In effect, the SPD multimetric measures the separation between the potential threats and the protected assets. Figure 1 illustrates the SPD methodology's main features.

The SPD Surface

When analyzing the SPD state, we determine whether possible threats exist. In the case of security, a threat is the potential for abuse of protected assets resulting from malicious human activity.^{5,6} In the case of privacy, a threat is a malicious or nonmalicious event that affects protected assets related to personal identifiable information (PII).^{7,8} In the case of dependability, a threat is a nonmalicious event in the fault chain (fault-error-failure).⁹

Not all threats contribute equally because not all interactions are equally likely to cause harm or cause the same damage. So, we identify the



TWO ATTACK SURFACE METRICS

To attack a system, an attacker exploits the system's resources (such as channels, methods, and data items). The attack is successful if the attacker directly or indirectly interacts with a resource. The *attack surface metric* (ASM) assembles the related parameters to estimate the attack surface, assuming that the smaller the surface (the exposure of the system's resources), the higher the security.¹ Not all resources contribute equally to the attack surface; attackers benefit more from specific resources (for example, attacking applications with root privilege to gain higher access rights to the exploited assets). To account for this, the ASM estimates the *damage potential–effort* (DP-E) ratio for each entry or exit point. (For more on the DP-E ratio, see the section “The SPD Surface” in the main article.)

Similarly, Microsoft models the Relative Attack Surface Quotient (RASQ) to mathematically quantify the relative attackability of the Windows server OS platforms.² This approach assigns a vulnerability level to each potential exploit point. RASQ is the summation of the effective attack surface values of all root attack vectors (system features that can affect security). Attack biases are aggregated, representing the risk of exploiting each attack vector. The metric is also validated against real exploits, on the basis of relevant reports from CVE (Common Vulnerabilities and Exposures) and US-CERT (US Computer Emergency Readiness Team) databases.

References

1. P.K. Manadhata and J.M. Wing, “An Attack Surface Metric,” *IEEE Trans. Software Eng.*, vol. 37, no. 3, 2010, pp. 371–386.
2. M. Howard and Microsoft Corp., *Determining Relative Attack Surface*, US patent 7299497 B2, Patent and Trademark Office, 2007.

possible *threat flows* (TFs) each involved system resource introduces. Then, we assign a *damage potential–effort* (DP-E) ratio to each TF. The DP-E ratio (which is based on the ASM) takes into account

- the potential damage to the system in case of disclosure and
- the effort the attacker is willing to devote to attack the resource.

The higher the potential damage or the lower the effort, the higher the TF's contribution to the attack surface.

We estimate the damage potential and effort in tandem because these two factors are usually related in real attack scenarios. For example, an attacker might be willing to spend effort in gaining high privilege levels to gain more access rights and cause more damage.

As we mentioned before, our SPD surface considers privacy and dependability in addition to safety. Although the general notion that attackers would target a high-privilege method seems correct in many cases, attackers could exploit a lower-privilege

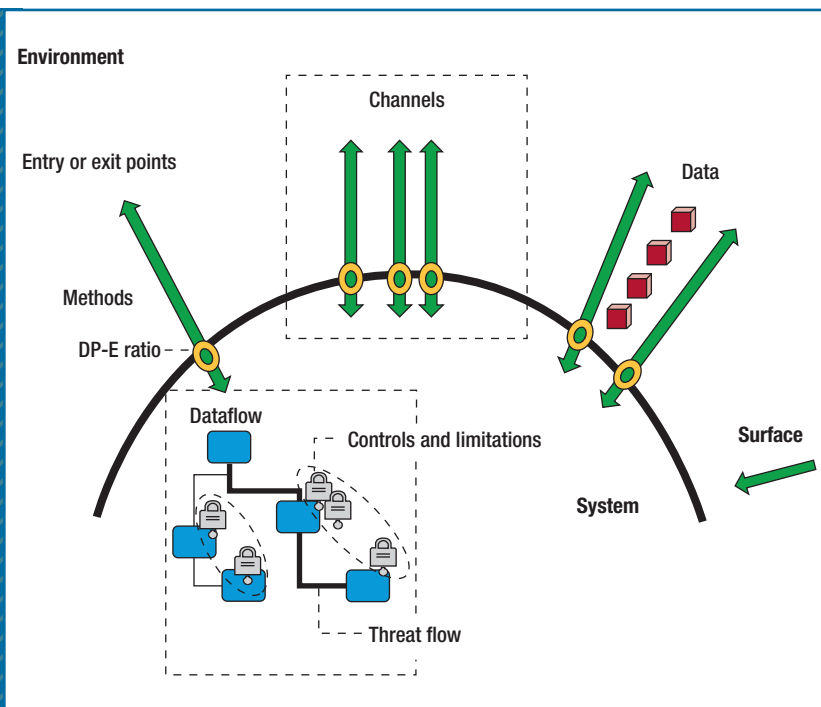


FIGURE 1. Our SPD (security, privacy, and dependability) methodology. The system's surface is the solid curve separating the assets and the environment. The interaction points are protected by controls along the dataflow. DP-E stands for the damage potential and effort of exploiting a dataflow and performing an attack.

method that has access to sensitive personal data. Nevertheless, according to surface metrics that focus just on safety, the latter method would be less likely to be attacked, with a lower contribution to the attack surface. Similarly, for dependability, a mission-critical method or a component with high dependency in the overall system architecture might be more valuable to an attacker than a method with administrator access rights.

Individually analyzing the three SPD properties could also produce inaccurate conclusions regarding attackability because an attacker could formulate an attack involving all three properties. So, the SPD surface evaluates the aggregated effect of the three properties, resulting in

more accurate system analysis and better design.

The DP-E ratio for each TF integrates three underlying ratios for security, privacy, and dependability. For security-related interaction, we determine the damage potential by evaluating the method's privilege and define the attacker's effort by the method's access rights. For privacy, we base the damage potential on the PII type and base the effort on the actuator type that processes that data. For dependability, we derive the potential damage from the component's criticality and derive the effort from the component's dependency on other components. Table 1 summarizes the DP-E ratio parameters and the values we used in our study.

The DP-E ratio for a TF is the summation of the relevant DP values divided by E:

$$TF_{DP-E} = \frac{(S_{DP} + P_{DP} + D_{DP})}{(S_E + P_E + D_E)},$$

The system surface is the summation of the underlying DP-E ratios:

$$Surface_{sys} = \sum TF_{DP-E}.$$

The SPD Value

Determining the SPD value involves analyzing the software's *porosity*, *controls*, and *limitations*.

Porosity. Porosity refers to the set of TFs for which interaction between assets and threats is possible. We break down each TF into *access*, *trust*, and *complexity pores* representing the effect on security, privacy, or dependability, respectively.

Controls. Controls are the means to influence a threat's impact when interaction occurs. They constitute the protection mechanisms that aim to prevent interactions of assets and threats and enhance their separation.

There are two types of controls. *Interactive controls* directly affect operations when interaction occurs. *Process controls* indirectly affect operations by protecting assets after interaction and creating defensive strategies.

We consider 12 interactive-control types (five for security, four for privacy, and three for dependability) and 14 process control types (four for security, six for privacy, and four for dependability), as described in the relevant standards. Tables 2 and 3 define these controls.

Limitations. Limitations restrict a control from working properly or prevent the separation of an asset and a

TABLE 1

DP-E (damage potential–effort) ratio parameters and values.

SPD property	Damage potential	Effort
Security	Privilege ⁵ <ul style="list-style-type: none"> • Root: 5 • Debugger: 4 • Authenticated user: 3 	Access rights ⁵ <ul style="list-style-type: none"> • Administrator: 4 • Authenticated user: 3 • Anonymous user: 1 • Unauthenticated user: 1
Privacy	PII (personal identifiable information) type ⁷ <ul style="list-style-type: none"> • Sensitive personal data: 5 • Personal data: 4 • Statistical data: 1 	PII actuator ⁷ <ul style="list-style-type: none"> • PII principal: 4 • Contracted PII processor: 3 • Third party: 2
Dependability	Criticality ⁹ <ul style="list-style-type: none"> • Mission critical: 4 • Business critical: 3 • Business operational: 2 • Business supporting: 1 	Dependency level ¹⁰ <ul style="list-style-type: none"> • Coupling: 4 • Outgoing dependency: 3 • Incoming dependency: 2 • Independent operation: 1

threat. There are six types of limitations. The first three directly affect porosity. *Vulnerabilities* increase access pores. They're flaws or errors that prevent access of authorized entities, allow privileged access to unauthorized entities, or allow unauthorized entities to hide assets or themselves. Disclosure increases trust pores and represents intentional or unintentional revelation of PII. Exposure increases complexity pores. It refers to unjustifiable actions, flaws, or errors that enable direct or indirect visibility of assets.

Two limitation types affect the two control types and their overall effect, influencing all three SPD properties. *Weaknesses* constitute flaws or errors that disrupt, reduce, or nullify interactive controls' positive contribution. Similarly, *con-*

cerns affect the process controls' flow or execution.

The last limitation type is *anomalies*, which include unidentifiable or unknown elements that can't be observed during normal operation. Because such conditions can't be controlled, a proper audit should notify developers of any anomalies. An anomaly might obstruct all SPD properties, so we evaluate its effect on each property.

Determining the SPD value. A system's SPD value is a triple vector representing the degree of asset protection in accordance with the controls or the reduction of the threats' impact. Each vector's value ranges from 0 to 100. However, an SPD value of <100, 100, 100> doesn't guarantee perfect protection. It just

denotes that all the required protection mechanisms are properly placed for all potential interaction flows and are safe, on the basis of the current set of known limitations—for example, from CVE (Common Vulnerabilities and Exposures) bulletins or US-CERT (US Computer Emergency Readiness Team) advisories. Higher values could denote wasted or redundant protection mechanisms and higher development or operational costs.

Applying many similar controls to protect the same operation type doesn't provide in-depth protection because defeating one of them usually leads to defeating them all. Increasing an asset's SPD value requires applying different and safe controls. Clear dataflows and business processes are prerequisites for high SPD values.

TABLE 2

Interactive controls.

SPD property	Aspect	Control	Description
Security	Confidentiality	Authentication	Challenges credentials on the basis of identification and authorization.
		Resilience	Retains protection in the case of corruption or failure.
	Integrity	Subjugation	Ensures that interactions occur according to a defined process, removing freedom of choice and liability in the case of disclosure.
	Availability	Continuity	Retains interactivity in the case of corruption or failure.
		Indemnification	Involves a contract between the asset owner and an interacting entity. It might include warnings as a precursor of legal action and public legislative protection.
Privacy	Collection	Consent	Involves the PII (personal identifiable information) principal's freely given, specific, and informed agreement to the PII's processing. The PII shouldn't be disclosed or shared with third parties without the PII principal's consent.
		Opt-in	Involves a process or policy in which the PII principal agrees explicitly to the PII's processing, before relevant consent.
	Access	Indentifiability	Results in identifying, directly or indirectly, the PII principal on the basis of a given set of PII. It might include complete indentifiability, pseudonymization, or anonymity.
		Notification	Notifies PII principals that their data is being collected.
Dependability	Reliability	Survivability	Provides degraded but useful operations that are acceptable to users for a specified period during a failure.
		Performability	Ensures how well the system will perform over a specified period in the presence of faults.
	Maintainability	Removal during use	Records and removes faults through the maintenance cycle, after production.

SPD Evaluation

The ASM details the methodology for identifying direct or indirect interaction between assets and threats and calculating a pore's DP-E ratio. The porosity value is based on the summation of the TFs, called $Porosity_{Base}$.

Controls

We place controls to restrict porosity. Although they're mapped to a specific SPD property, they influence all three of them. So, the final calculation considers the aggregated contribution of each property's controls. For every pore, we identify the applied controls. To achieve

perfect coverage of a pore, at least one instance of each control must be mounted. All controls contribute equally to a pore's coverage. We denote the perfect coverage of interactive and process controls as

$$PerfectCoverageInteractive = Porosity_{Base} \times 12,$$

$$PerfectCoveragePassive = Porosity_{Base} \times 14.$$

Limitations

The justification of a limitation is a risk decision. You can either control

the limitation somewhat or accept the damage it can cause. (For example, the cost to fix the problem might not be justified by the potential damage.)

Vulnerabilities, disclosure, or exposure.

To measure the contribution of vulnerabilities, disclosure, or exposure, we perform a calculation based on the *attack potential*.⁵ This function combines the attacker's expertise, motivation, and preference to exploit particular limitations and attack specific assets, and the resources he or she is willing to devote. We extend this function to include not only

TABLE 3

Process controls.

SPD property	Aspect	Control	Description
Security	Confidentiality	Confidentiality	Ensures that a processed asset isn't known outside the interacting entities.
	Integrity	Integrity	Ensures that the interacting entities know when an asset has been altered.
		Nonrepudiation	Prevents the interacting entities for denying their role in an interaction.
	Availability	Alarm	Notifies that an interaction is occurring or has occurred.
Privacy	Collection	Fairness	Ensures that the PII is collected, used, or disclosed for only the appropriate purposes.
	Access	Challenge compliance (accountability)	Ensures that PII principals can hold PII processors accountable for adhering to all privacy controls.
	Usage	Retention	Ensures that PII that's no longer required isn't retained, to minimize unauthorized collection, use, and disclosure.
		Disposal	Provides mechanisms for disposing of or destroying PII.
		Report	Reports that an interaction regarding PII is occurring or has occurred.
		Break or incident response	Manages a PII breach.
Dependability	Reliability	Tolerance	Ensures the required functionality's delivery in the presence of faults.
	Maintainability	Forecasting	Predicts likely faults so that they can be removed or their effects can be circumvented.
	Safety	Prevention	Prevents faults from being incorporated into a system. This is accomplished through good development methodologies and implementation techniques.
		Removal during development	Verifies a system so that faults are detected and removed before the system goes into production.

malicious attacks but also the general concept of a threat, as we described earlier.

To analyze a potential threat, we consider five factors:

- *Elapsed time.* The time required to identify and exploit the limitation, measured in days, weeks, or months.
- *Specialist expertise.* The technical expertise required (for

example, layperson, proficient, or expert).

- *Knowledge of the target.* Knowledge of the target's design and operation (for example, public, restricted, sensitive, or critical information about the target).
- *Window of opportunity.* A factor related to the elapsed time. An attacker might require considerable access to the target to

successfully exploit the threat without detection.

- *Resources.* The IT hardware or software or other equipment required (for example, standard, specialized, or bespoke equipment).

On the basis of these factors, we characterize a potential threat as basic, enhanced basic, moderate, high, or beyond high. This

approach doesn't consider every possible circumstance but gives a good indication regarding the protection level in accordance with standard ratings.

The factors' individual ratings could indicate high protection. However, one limitation could lead to another that's easier to exploit, resulting in a lower overall rating. So, for our SPD calculation, we compute the value assigned to each of these three limitations as a weighted summation of the relevant individual ratings: V_V , V_D , and V_E , respectively. We divide these values by the pore type's attackability:

$$L_V = V_V / \sum Access \times TF_{DP-E},$$

$$L_D = V_D / \sum Trust \times TF_{DP-E},$$

$$L_E = V_E / \sum Complexity \times TF_{DP-E}.$$

Weaknesses and concerns. The summation of the missing interactive and process controls divided by *PerfectCoverageInteractive* and *PerfectCoverageProcess* determines the impacts of the weaknesses and concerns, called L_w and L_c , respectively. As with controls, these two limitation types influence all three SPD properties.

Anomalies. Anomalies alone don't impact SPD. We evaluate their influence only in the presence of other limitations, as their summation divided by the summation of *PerfectCoverageInteractive* and *PerfectCoverageProcess*, called L_A .

Vector Calculation

The final SPD vector calculation is perfect separation (100) minus the weighted summation of the limitations:

$$S = 100 - [W_V \times L_V + (W_W \times L_W + W_C \times L_C + W_A \times L_A)],$$

$$P = 100 - [W_D \times L_D + (W_W \times L_W + W_C \times L_C + W_A \times L_A)],$$

$$D = 100 - [W_E \times L_E + (W_W \times L_W + W_C \times L_C + W_A \times L_A)],$$

where W^* is each limitation type's threat bias (similar to RASQ).

Application

To assist SPD evaluation, we implemented a user-driven scanning tool. It uses two Eclipse plug-ins: CallGraph (<https://marketplace.eclipse.org/content/callgraph-viewer>) for parsing C/C++ programs and TACLE (Type Analysis and Call Graph Construction for Eclipse)¹¹ for parsing Java programs. The plug-ins create call graphs of programs presenting the function or method calls, timing, or usage.

We automated most evaluation steps, leaving a limited set of decisions for the user. The tool identifies the high-level entry and exit points and resources (that is, it exploits the Java aspect modifiers—public, protected, default, or private—of an object's methods and parameters), constructing the surface. The user clarifies which of these elements contribute to porosity and indicates the relevant controls. The tool precomputes and then maintains the potential-threat analysis of the limitations and the calculation of the relevant values. Finally, the user maps the limitations to pores and controls, and the tool calculates the SPD value. Designers can easily adjust the system configurations and estimate the best SPD for different settings.

The EU project nSHIELD (New Embedded Systems Architecture for

Multi-layer Dependable Solutions; www.newshield.eu) is using this tool to evaluate the protection level of prototype configurable embedded software in a social-mobility scenario. The scenario includes a wireless-sensor-network-equipped smart-city infrastructure that gathers ambient and cybersecurity information, two smart vehicles, and a back-end command and control (C&C) center that aggregates collected information and communicates with all entities.

nSHIELD is constructing a real-time critical-incident-response system in which the C&C center manages the system automatically to deal with emergencies (for example, cyberattacks or car crashes). The center employs AI with contextual information regarding the configuration states. The AI calculates the current setting's SPD level and manages the system entities on the basis of scenario events.

Ideal SPD settings might not always be possible. For example, in applications requiring low latency (such as vehicle-to-vehicle communication), smart vehicles might use simpler cryptographic mechanisms, inducing lower security. Also, smart vehicles might denote a larger (inexact) relevant location to obstruct movement tracking and enhance driver privacy. In the case of a crash, a vehicle's dependability will decrease according to the damage (if components critical for dependability break).

The prototype software frameworks are configured at runtime to comply with the specified SPD goals. The functionality includes policy-based access control; trust-based routing; secure communication and storage; GSM (Global System for Mobile Communications)

The evaluated software frameworks.


Prototype	Description	Configuration	Attack surface size	SPD value
Secure storage	Storage that locally (at the embedded-device level) maintains protected information regarding user data and log files	No encryption	2,745.67	<20, 17, 8>
		Lightweight cryptography	3,177.87	<40, 35, 15>
		Mainstream cryptography	3,194.88	<80, 70, 30>
Trust-based ad hoc routing	A secure routing protocol and an intrusion detection system for wireless sensor networks that monitor ambient information of the smart city	Simple routing	3,808.64	<30, 20, 20>
		Direct trust	4,546.56	<45, 30, 25>
		Direct and indirect trust	4,867.23	<90, 60, 50>
Policy-based access control	A management framework for access control in heterogeneous embedded-system networks, based on policies	Unencrypted communication	3,201.54	<10, 30, 10>
		Authentication	4,137.83	<50, 50, 50>
		Authenticated encryption	4,467.22	<80, 70, 80>
		Web Services Security	4,803.36	<90, 80, 90>
Location-based services	A framework for providing location-based services through GSM (Global System for Mobile Communications) with relevant security and privacy protections	Anonymous services	3,367.55	<20, 80, 20>
		Pseudonymity	3,541.13	<30, 70, 30>
		k-anonymity	3,972.46	<40, 85, 45>
		Authenticated, accurate, and secure communication	4,493.31	<80, 60, 80>

communication and location-based services; and monitoring, alerting, and management services. There are four software frameworks and 14 configurations.

Table 4 details the prototypes, surface sizes, and SPD values. Surface analysis alone was inadequate to determine the protection level. All the framework configurations that provided low protection produced a smaller surface than the safer settings. For example, the secure-storage framework presented the smallest surface when no encryption was deployed. Although the configurations that applied cryptography produced a larger surface owing to the additional functionality, they weren't the most unsafe

settings. Our SPD methodology effectively derived the protection level and the state of the safety, privacy, and dependability.

Our proposed solution will be offered as a service for secure system management and embedded-system development. Besides smart vehicles, the application of the overall methodology in other domains, such as e-health and the smart grid, can leverage the protection level of existing settings. Moreover, cloud-computing and Internet-of-Things architectures enable new ways of interaction and impose challenges for software measurement. The SPD methodology could be

extended to measure interconnected embedded devices that communicate information to the cloud. 

References

1. N. Fenton, P. Krause, and M. Neil, "Software Measurement: Uncertainty and Causal Modeling," *IEEE Software*, vol. 19, no. 4, 2002, pp. 116–122.
2. F. Shull et al., "Fully Employing Software Inspections Data," *Innovations in Systems and Software Eng.*, vol. 8, no. 4, 2012, pp. 243–254.
3. L.S. Azevedo et al., "Assisted Assignment of Automotive Safety Requirements," *IEEE Software*, vol. 31, no. 1, 2014, pp. 62–68.
4. K.V. Prasad, T.J. Giuli, and D. Watson, "The Case for Modeling

SUBMIT
TODAY

IEEE TRANSACTIONS ON
BIG DATA

► **SUBSCRIBE
AND SUBMIT**

For more information
on paper submission,
featured articles, call-for-
papers, and subscription
links visit:

www.computer.org/tbd

TBD is financially cosponsored
by IEEE Computer Society, IEEE
Communications Society, IEEE
Computational Intelligence Society,
IEEE Sensors Council, IEEE Consumer
Electronics Society, IEEE Signal
Processing Society, IEEE Systems,
Man & Cybernetics Society, IEEE
Systems Council, IEEE Vehicular
Technology Society

TBD is technically cosponsored by
IEEE Control Systems Society, IEEE
Photonics Society, IEEE Engineering
in Medicine & Biology Society, IEEE
Power & Energy Society, and IEEE
Biometrics Council



IEEE
computer
society

ABOUT THE AUTHORS



GEORGE HATZIVASILIS is a PhD candidate in the Technical University of Crete's School of Electronic & Computer Engineering. His research interests include embedded-system security, privacy, and dependability. Hatzivasilis received an MSc in computer science from the University of Crete. He's a member of IEEE. Contact him at gchatzivasilis@isc.tuc.gr.



IOANNIS PAPAEFSTATHIOU is a professor in the Technical University of Crete's School of Electronic & Computer Engineering. He's interested in the architecture and design of novel computer systems. Papaefstathiou received a PhD in computer science from the University of Cambridge. Contact him at ypg@mhl.tuc.gr.



CHARALAMPOS MANIFAVAS is a professor in the Department of Electrical Engineering and Computing Sciences, Rochester Institute of Technology Dubai. His research interests include network and information security. Manifavas received a PhD in computer science from the University of Cambridge. Contact him at cxmcd@rit.edu.

- Security, Privacy, Usability and Reliability (SPUR) in Automotive Software," *Model-Driven Development of Reliable Automotive Services*, LNCS 4922, Springer, 2008, pp. 1–14.
5. *Common Criteria for Information Technology Security Evaluation*, ISO/IEC 15408, 1996–2015; www.commoncriteriportal.org.
 6. *Open Source Security Testing Methodology Manual*, ISECOM, 1988–2015; www.isecom.org/research/losstmm.html.
 7. *Privacy Framework*, ISO/IEC 29100, 2011; www.iso.org/obp/ui/#iso:std:iso-iec:29100:ed-1:v1:en.
 8. *Code of Practice for Protection of Personally Identifiable Information (PII) in Public Clouds Acting as PII Processors*, ISO/IEC 27018, 2014; www.iso.org/iso/catalogue_detail?csnumber=61498.
 9. *International Standards on Dependability*, IEC 60300, 2006; www.iec.ch/about/brochures/pdf/technology/dependability.pdf.
 10. T. Zimmermann et al., "An Empirical Study on the Relation between Dependency Neighborhoods and Failures," *Proc. IEEE 4th Int'l Conf. Software Testing, Validation and Verification (ICST 11)*, 2011, pp. 347–356.
 11. J. Sawin and A. Rountev, "Estimating the Run-Time Progress of a Call Graph Construction Algorithm," *Proc. 6th IEEE Int'l Workshop Source Code Analysis and Manipulation (SCAM 06)*, 2006, pp. 53–62.