Can We Ever Build Survivable Systems from COTS Components?

Howard F. Lipson, Nancy R. Mead, and Andrew P. Moore

CERT® Coordination Center, Software Engineering Institute
Pittsburgh, PA 15213 USA
{hfl,nrm,apm}@sei.cmu.edu
http://www.cert.org/research/

Abstract. Using commercial off-the-shelf (COTS) components to build large, complex systems has become the standard way that systems are designed and implemented by government and industry. Much of the literature on COTS-based systems concedes that such systems are not suitable for mission-critical applications. However, there is considerable evidence that COTS-based systems are being used in domains where significant economic damage and even loss-of-life are possible in the event of a major system failure or compromise. Can we ever build such systems so that the risks are commensurate with those typically taken in other areas of life and commerce?

This paper describes a risk-mitigation framework for deciding when and how COTS components can be used to build survivable systems. Successful application of the framework will require working with vendors to reduce the risks associated with using the vendors' products, and improving and making the best use of your own organization's risk-management skills.

1 Introduction

Lower upfront costs, and a belief that the cost savings extend throughout the system's lifecycle, are primary motivators in the shift from custom-designed to COTS-based systems. The disadvantages associated with COTS-based design include the absence of source code and lack of access to the artifacts of the software engineering process used to design the COTS components.

Whether you've built your system using COTS components from many vendors, or a single vendor has provided you with an integrated solution, many of the risks associated with system management and operation are not in your direct control [2], [3], [8], [10]. Each vendor that plays a role in the design, development, acquisition, integration, deployment, maintenance, operation, or evolution of part (or all) of your system affects the risks you face in your attempt to survive cyber-attacks, accidents, and subsystem failures. We propose

® "CERT" and "CERT Coordination Center" are registered in the U.S. Patent and Trademark Office.

A. Banks Pidduck et al. (Eds.): CAISE 2002, LNCS 2348, pp. 216–229, 2002.

continual vendor-based risk evaluations as a critical part of the system lifecycle for mission-critical systems that use COTS components.

Survivable systems are those that continue to fulfill their missions (perhaps at a reduced level of service), despite having components or subsystems that are damaged or compromised by attack, accident, or failure. SEI research into the design and analysis of survivable systems [9], [6] has shown that system survivability is dependent upon well-reasoned tradeoffs among the various quality attributes of a system's architecture and implementation. The design rationale and quality attribute tradeoffs are among the many engineering artifacts that are not available to the consumers of COTS components. Is it then impossible to build survivable systems out of COTS components?

Risk management is central to the achievement of survivability [11]. Those who acquire, design, implement, operate, maintain, and evolve systems that use COTS components can significantly enhance the survivability of such systems by working with vendors to reduce the risks inherent in the vendors' products and processes, and by improving and making the best use of their own organization's risk management skills. This paper suggests approaches that point the way towards future COTS components and vendor processes that provide sufficient visibility into a product's internals to give ample evidence that the use of these components can contribute to the assurance of overall system survivability.

2 Survivability and COTS Components

Survivability cannot be achieved without a clear understanding of the context in which modern systems typically operate – unbounded domains. Unbounded domains, such as the Internet, are characterized by a lack of central control, and a lack of complete, timely, or precise information. Moreover, a typical contemporary system constitutes an unbounded domain. In the absence of full control and full visibility into a system and its environment, achieving survivability (i.e., fulfilling the mission of a system) is an exercise in risk-management and risk tolerance. If your system is primarily composed of COTS components, then you have a rather extreme case of lack of control and lack of visibility regarding the ultimate behavior of your system under a variety of circumstances that could threaten its survival [12].

System vulnerabilities that are extremely unlikely to cause mission failure due to the actions of a normal user may very likely be exploited by an intelligent adversary, e.g., by taking advantage of buffer overflow vulnerabilities [5], particularly when scripts are developed that encode the often intricate and detailed steps needed for successful exploitation. Survivability, therefore, demands high assurance that (1) such vulnerabilities do not exist or cannot be exploited, or (2) that their exploitation does not compromise the mission or can be recognized and recovered from to continue the mission [6]. This need for high assurance is what makes the use of COTS components in mission-critical systems so difficult.

Of course, COTS components can always be used to implement non-critical system functions, that is, functions whose properties do not impact system sur-

vivability. Some architectures have demonstrated how to structure a system so that the critical function is isolated to small, high assurance components, thus allowing COTS components to be used anywhere else [7]. Unfortunately, such approaches are currently limited to fairly narrow properties, such as military confidentiality.

Where such approaches are not available, the question is whether COTS components can be used to implement critical system functions. Survivability techniques often rely on redundancy to tolerate compromises of individual components. Layered defenses (e.g., using intrusion detection and recovery to complement resistance measures) also help to tolerate failures in critical function implementations. As a result of a design team's judicious use of replication, redundancy, and diversity, the property of system survivability can emerge from the interactions among the individual components of a system even when the components themselves are not survivable. But what assurances are required of COTS components that implement critical system functions?

The criticality of a system influences the assurance requirements for COTS components that implement essential services. A system has high criticality if the consequences of system failure are severe. A system has low criticality if the consequences of system failure are negligible. Fig. 1 maps the COTS component assurance required as a function of system criticality. There are many factors that influence COTS component assurance, which will be elaborated later in the paper. However, for the purposes of discussing the figure, we treat COTS component assurance abstractly, assuming only that assurance can vary greatly.

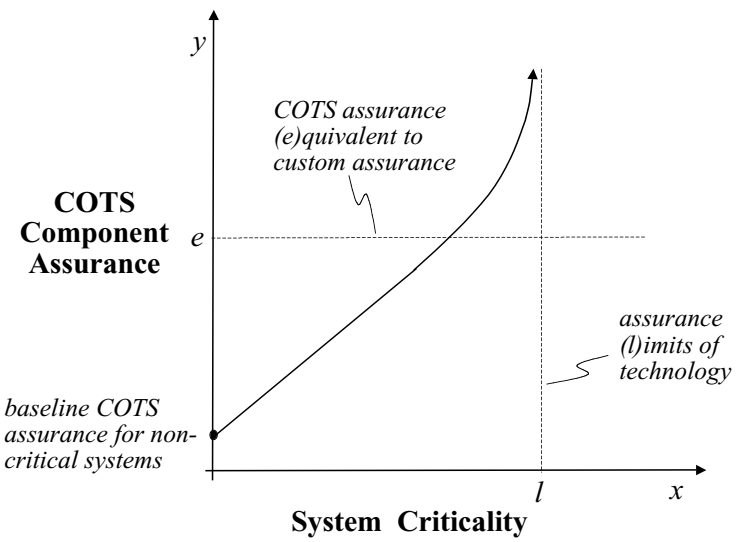

Fig. 1. COTS Component Assurance Required as Function of System Criticality

We assume in Fig. 1 that there is some baseline COTS component assurance required even for non-critical systems. Suppose (c, a) is a particular point along the curve, where $c < l$ is the criticality of the system being built. Then a represents the minimal assurance permitted of any COTS component used to implement the system's essential services. The area above the curve represents acceptable use of COTS components, while the area below the curve represents unacceptable use. We assume that the proposed system relies on the use of COTS for the survival of the system's mission. We also assume that there is a point at which the potential impact is so severe that computing/network technology should not be used regardless of the assurances that the technology affords. This limit is shown as the dashed line l in Fig. 1. The asymptotic nature of the curve to the limit l reflects the need for "infinite" COTS component assurance to implement such high consequence systems.

The dashed line labeled e in Fig. 1 represents COTS component assurance that is as convincing as the custom development assurance. While this varies with the custom development process used, postulating such a point reflects the real possibility that COTS assurance can exceed that of custom development. Some may argue that COTS components with such high assurance do not currently exist and, even if they did, they could not be considered to actually be COTS at that point. We do believe, however, that within our characterization of COTS, COTS components may have certain assurance benefits over custom development – for example, vendor expertise, history of use, and ensured evolution. Given this, the part of the curve above the dashed line e represents systems that are so critical that they actually require the use of very high-assurance COTS components instead of custom development.

3 COTS vs. Custom Design — A Binary Choice, or a Spectrum of Choices?

The term COTS, as it is generally understood, refers to widely-available commercially produced software supplied as object code, for which the only information you might have about the product is the purchase price, a list of features, a user's manual, some vendor claims about the product, a license agreement, an application programming interface (API) specification, and your own experience with the product or trial version (or some third-party experience or test results of which you are aware). Hence, your visibility into the product and the process used to construct it is limited in the extreme. The only control you have over the product is whether to buy it and how many copies you'll purchase.

At the other extreme of the software development world is custom-designed software and systems. However, software design and development always involves risk management, since the processes for even well-engineered custom-designed software involve a lack of full control and a lack of full visibility. The iterative analysis and design methodologies that help to determine the ultimate functional and non-functional attributes of a system, and the tradeoffs among the software quality attributes, functionality, and cost, are suspended when you feel comfort-

able that the risks associated with the design, implementation, deployment, and use of the system are below your risk-tolerance threshold [9]. Moreover, your compiler and other programming tools are likely to be COTS products, as are your development and target platforms. You can never be completely aware of the backgrounds and skills of the personnel building your system, though the efforts you make in terms of background and reference checks help to reduce your risk, in exchange for some additional time and expense.

Therefore, we wish to dispel the notion that organizations seeking to build survivable systems have only a binary choice, COTS or custom, where COTS-based systems lack almost all control and visibility, but are relatively inexpensive, and custom-built systems give you full control and visibility, albeit at much greater up-front cost. We believe there is a middle spectrum of design choices, ranging from 100% black-box COTS component integration to 100% custom-design, that allows a much more flexible, cost-effective, and risk-mitigating approach to the design of survivable systems. The *V-RATE* method, described in the next section, outlines a set of enabling strategies for mitigating the risks associated with using COTS products. Some of the strategies enable risk reduction by providing more control and more visibility into the internals of a product and the processes used to construct it. However, V-RATE includes other alternatives for mitigating risk that don't involve increasing control and visibility.

4 The V-RATE (Vendor Risk Assessment & Threat Evaluation) Method

Building survivable systems using COTS components is a daunting task because the developer has little or no access to the artifacts of the software engineering process used to create the components. These artifacts are the primary sources from which assurance evidence for a composite system is derived. One way to partially compensate is to use vendor risk assessments as a tool to help you build, maintain, and evolve survivable systems. Such an assessment can be used as a new source of assurance evidence of a system's survivability.

Our proposed vendor risk assessments are based on a *V-RATE (vendor risk assessment and threat evaluation) taxonomy* described below. Two broad categories are at the highest level of our taxonomy: (1) vendor-inherent risk elements and (2) vendor risk elements associated with your own risk management skills. The output of an assessment based on the V-RATE taxonomy is a *vendor-risk profile* for the system being evaluated. We envision a large and growing collection of vendor-risk profiles tied to real-world performance histories, providing empirical data against which a newly generated risk profile can be compared. A vendor-risk profile can be used to assess the risk associated with the use of a product in a particular threat environment, and to identify areas for additional risk-mitigation activities. Because a single numerical rating would not provide sufficient guidance for these risk-mitigation activities, the vendor-risk profile helps you to identify your risks in each of the V-RATE taxonomy areas,

and allows you to consider your risk tolerance with respect to each element of the taxonomy.

4.1 The V-RATE Taxonomy

Elements of the V-RATE taxonomy include:

1. **Vendor's Inherent Risk Elements**
   - 1.1 ***Visibility of Product Attributes***
     - 1.1.1 Openness – Degree of visibility into design and engineering processes
     - 1.1.2 Independent testing organizations
   - 1.2 ***Technical Competence***
     - 1.2.1 Survivability capability maturity
     - 1.2.2 Existence of vendor ratings/certifications
     - 1.2.3 Evidence of adherence to applicable industry standards and government regulations
     - 1.2.4 Demonstrated diversity and redundancy in a vendor's products and services
     - 1.2.5 Existence of a vendor team that deals effectively with security/survivability issues
   - 1.3 ***Performance History***
   - 1.4 ***Compliance***
     - 1.4.1 Responsiveness to security/survivability issues (which can include related quality issues such as reliability, performance, safety, and usability)
     - 1.4.2 Responsiveness to requests for new features and improvements
     - 1.4.3 Willingness to cooperate with third-party testers and certifiers
   - 1.5 ***Trustworthiness***
     - 1.5.1 Track record / Word-of-mouth
     - 1.5.2 Evidence of skill at evaluating trustworthiness of personnel
   - 1.6 ***Business Management Competence***
     - 1.6.1 Economic viability
     - 1.6.2 Vendor's risk-management skills in dealing with subcontractors
   - 1.7 ***Controlled Evolution***
     - 1.7.1 Clearly specified (or discernible) evolutionary path
     - 1.7.2 Product integration stability
     - 1.7.3 Product evolution supports continual survivability improvement
2. **Vendor Risk Elements Associated with Your Risk Management Skills in Dealing with Vendors**
   - 2.1 ***Technical Risk-Mitigating Factors***
     - 2.1.1 Your skill at evaluating a product's quality attributes (in particular, those quality attributes that can contribute to system survivability, such as security, reliability, performance, safety, and usability)
     - 2.1.2 Your skill at evaluating vendor technical competence
     - 2.1.3 Awareness of existing vendor ratings and certifications

2.1.4 Demonstrated diversity and redundancy in the integration of vendor products and services
2.1.5 Use of architectural tools and techniques (e.g., wrappers) to limit risks associated with a vendor product
2.1.6 Your association with expert security/survivability organizations, and the existence of a dedicated security/survivability group within your own organization
2.2 ***Non-Technical Mitigation of Risk***
2.2.1 Legal
2.2.2 Economic
2.2.3 Political and social
2.3 ***Independence / Interdependence***
2.4 ***Your Exposure***
2.5 ***Mission Alignment / Vendor Compatibility***
2.6 ***Your Negotiating Skill / Bargaining Power***

4.2 Specific Vendor Risk Reduction Techniques

The V-RATE method provides a framework for assessing survivability risks associated with COTS products. Although there are many risks and much work to be done, there are specific ways that risk can be reduced. In the long term, we would like to see a full list of vendor risk reduction techniques. Each technique could be assigned a value that could be used in the V-RATE calculation to show reduction of overall survivability risk associated with specific COTS products.

For each element of the V-RATE method, specific strategies should be developed to reduce risk. In Table 1 we provide some brief examples of ways in which risk can be reduced. We align these examples with the V-RATE taxonomy.

The following are expanded examples for two of the items in the table. This expansion could be done for the entire table to form a comprehensive set of examples/strategies.

Example of V-RATE Taxonomy Section 1.4, Compliance. The vendor shows a willingness to respond to security and survivability concerns by:

- making security patches available quickly.
- allowing the client to *turn off* unneeded features and thus reduce the risks associated with those features. In this way the client can select a *core* set of needed services, rather than be forced to live with the consequences of "one size fits all."
- building recovery mechanisms into the software. Examples of such mechanisms are automated back up of data and retention of state data.
- building security (resistance) mechanisms into the software. Examples are encryption, password protection, and diversity.
- putting specific software engineering practices in place to improve security, such as inspections, testing, the use of strongly typed languages, and processes that support good programming practices. Another positive response to customer concerns would be to initiate or increase education and training in security and software engineering for the vendor's technical staff

Table 1. V-RATE Risk Reduction Examples

V-RATE Element	Example
1.1 Visibility of Product Attributes	The vendor is willing to allow client to see source code corresponding to installed binaries.
1.2 Technical Competence	The vendor has demonstrated (rated) competence in key survivability activity/practice areas (using a survivability capability maturity model).
1.3 Performance History	The vendor has a track record – experience, statistics, testimonials, and word-of-mouth.
1.4 Compliance	The vendor makes security patches available quickly.
1.5 Trustworthiness	The vendor consistently checks character references of new hires and periodically re-checks all personnel.
1.6 Business Management Competence	The vendor's prospects for long-term economic health are good.
1.7 Controlled Evolution	The vendor shares plans and procedures that indicate controlled product evolution.
2.1 Technical Risk-Mitigating Factors	You have the skills needed for direct technical risk evaluation (including, but not limited to Survivable Systems Analysis).
2.2 Non-Technical Mitigation of Risk	You have access to legal or economic protection, such as insurance, warranty and license agreements, performance clauses and penalties, regulatory protection, and performance bonds.
2.3 Independence / Interdependence	You examine the vendor products and services associated with your system and look for interdependencies that could threaten survivability.
2.4 Your Exposure	You determine what elements of the system are dependent upon the competence, trustworthiness, and thoroughness of the vendor.
2.5 Mission Alignment / Vendor Compatibility	You evaluate alignment of your mission and required software quality attributes (SQA's) with vendor mission and SQA's.
2.6 Your Negotiating Skill / Bargaining Power	You partner with vendor to obtain early notification of potential security/survivability problems.

Example of V-RATE Taxonomy Section 1.7, Controlled Evolution. The vendor's plans and procedures indicate controlled product evolution as follows:

- Vendor upgrades do not require massive re-integration (such as major rewrites of API glue code).
- Applying security patches should not be delayed by the ripple effects of changes in the vendor product.
- There is a low degree of feature coupling.
- Changes in a few features do not cause massive maintenance headaches.

- The vendor is willing to provide insight into business plans for the product, so that the client has some idea of the stability of the product.
- The vendor agrees to support the product, particularly from a security and survivability perspective, over the long term.

5 A V-RATE Example

Although the V-RATE framework is at a very early stage of development, it is crucial to understand from the outset that it is not the goal of this research to assign a single or small number of composite numerical ratings to vendor product and processes for purposes of direct comparison. We instead envision that the output of the application of the V-RATE method will be a vendor-risk profile that is personalized to the specific organization that is interacting with a vendor or group of vendors for the purpose of acquiring, developing, operating, or maintaining a mission-critical COTS-based system. Each element of the V-RATE taxonomy represents an area of added risk that is not present in custom-designed systems. A heightened awareness of these risks (which enables you to take steps to reduce them) is the main benefit to be achieved by mapping the V-RATE taxonomy onto an existing or proposed design that includes COTS components.

Let's consider an e-commerce system that is used for Internet purchases as exemplified in Fig. 2, incorporating a number of COTS products, such as a Web server, a firewall, and a database application. The V-RATE taxonomy can serve

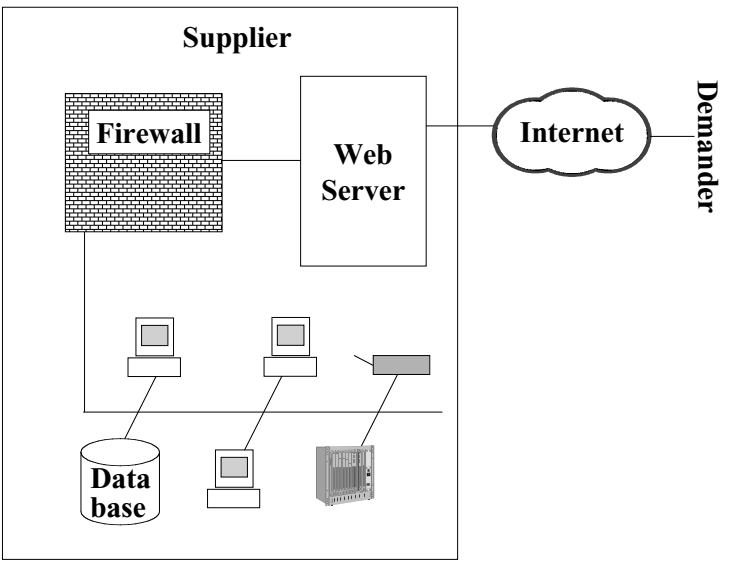

Fig. 2. Example E-Commerce System Architecture

as a road map to examine each of the COTS products used to implement the architecture and can be used in conjunction with architectural modifications, based on architectural-level survivability strategies, to enhance the survivability of the system. This process is inherently iterative and risk-driven.

Once mission requirements are defined, the design team should examine the existing (or proposed) e-commerce system architecture and modify it to support high-level survivability strategies, in the context of scenarios that threaten the business mission. Based on this scenario-driven examination of the architecture, the designers may decide, for example, to

- Add a second firewall (i.e., a DMZ) for defense-in-depth against cyber-attack, and a backup Web server in the event of accident or attack.
- Deploy redundant (and diverse) databases to recover from data loss or corruption.
- Contract with redundant (and diverse) service providers for more survivable Internet connectivity.

Once the architectural-level survivability strategies are in place, the design team must ensure that the components used to implement the architecture are technically sound. The V-RATE taxonomy can be used to gather evidence of assurance. The first step is to annotate a representation of the architecture with vendor names, and other vendor attributes, to identify areas of exposure. For example, it is well known that apparently independent telecommunications service providers offer connectivity solutions that share the same physical fiber. Designers should ask the right questions of their service providers (or use other means) to ensure diversity.

Next, proceed down the taxonomy to gather evidence of assurance. First, consider the openness of a vendor's component. Can you negotiate full access to the source, and other engineering and design artifacts (perhaps under a confidentiality agreement)? If you can, does your staff have the expertise to do the analysis and testing necessary to provide the required assurance? If not, perhaps third-party testing and analysis will provide the expertise and assurance you need, and might actually be superior to giving your own staff access to the source code. This might be an example of a situation where COTS can be superior to custom-built, for instance, if a vendor with a high degree of expertise and experience in a given industry problem domain is willing to make the results of their testing and analyses available. COTS-related risks would be further reduced by verifying the vendor's detailed technical claims through competent third-party testing and analysis.

Continue, in this manner, step-by-step down the V-RATE taxonomy. In particular, carefully consider the value of non-technical approaches to mitigate risk, such as performance bonds, legal disclaimers, and insurance. The process of survivable architecture refinement and COTS-product risk assessment proceeds iteratively as required by mission demands and business constraints.

Use the V-RATE taxonomy to gather evidence of assurance for the set of COTS products under consideration. This collection of evidence will then allow you to compare the products. Numerical ratings within each category (indicating

both strength of assurance and importance of this evidence to the system owner) and composite ratings that represent accumulated evidence across categories, allow for relative rankings of concerns, with the caveat that composite rankings derived from such numbers may only be used with extreme caution. Composite ratings suffer from the danger that ratings for taxonomy elements important to your organization will be cancelled out by ratings associated with elements that are not of concern to you.

On the other hand, a detailed vendor-risk profile would provide ratings across a number of risk-relevant elements that are keyed to emphasize what a specific organization deems important. We envision tools that provide

- multiple views or perspectives of a vendor-risk profile
- the ability to group or ungroup related taxonomy elements to yield a wide range of composite representations
- views that are suitable for "what-if" analyses and comparison of design or acquisition alternatives.

However, the specific numerical ratings and vendor-risk profiles that result from the V-RATE process are not as important as what can be learned by going through the process itself. We believe that risk will be significantly reduced if the system owner goes through the exercise of assigning importance to each V-RATE category and assessing the risks associated with the vendor's products and processes, as well as the system owner's management processes. Further gains can be achieved by obtaining the support of the vendor in this exercise.

6 How V-RATE Relates to the Common Criteria

The Common Criteria (CC), ISO International Standard 15408, represents an attempt to provide a structured, yet flexible approach to help consumers and vendors of security-relevant COTS products agree on and evaluate required product function and product/process assurance [4]. The CC promotes creating two documents called the Protection Profile and the Security Target. Consumers develop a Protection Profile for a class of security-relevant products of interest, such as firewalls, operating systems, and smart cards. The Protection Profile specifies the function and assurances required by a broad consumer base for the class of product independent of any particular implementation. A vendor develops a Security Target to describe their implementation of a product intended to conform to a particular Protection Profile. The Security Target specifies the security functions supported, the development process used, and an argument for why the functions and processes conform to the Protection Profile targeted. The CC sets forth guidelines for the production and independent evaluation of a vendor's Security Target in response to a consumer Protection Profile.

While originally intended as a vehicle for internationally-accepted IT security evaluation, the CC may provide a model for using V-RATE to promote increased trustworthiness of COTS products. V-RATE provides criteria for the vendor's product and process that aid the evaluation of COTS technology for use

in achieving a particular mission. V-RATE's criteria are technology-independent and thus, more abstract than the CC. V-RATE also includes criteria for assessing the consumer's own ability to deal with COTS vendors and the inherent risks associated with COTS technology. One of the criticisms of the CC is the large amount of overhead needed to produce and evaluate products within its framework. V-RATE may provide a middle ground between the black-box acceptance of COTS and a CC-evaluated product.

A significant difference between the Common Criteria and V-RATE is that V-RATE is conducted by, or on behalf of, the party whose security and survivability is at risk (and from that party's perspective), whereas a Common Criteria evaluation is typically paid for by the vendor, and is conducted on the vendor's behalf [1].

7 Summary and Future Work

Too many organizations take an all-or-nothing view with regard to the use of COTS in mission-critical systems (e.g., either COTS components are never safe to use, or COTS use should be maximized). This paper describes V-RATE criteria to help decide when and how COTS products can be used to build survivable systems. Factors that influence this decision include not only attributes of the COTS products themselves, but also attributes of the system's mission, the vendor, the vendor's development lifecycle process, and your own organization's risk management skills.

Increased vendor cooperation will improve the V-RATE method's effectiveness. Organizations often expect too little of their vendors, in terms of visibility into the internals of vendor products and processes, or meaningful guarantees of quality. Expectations need to be raised so that vendors more directly support the risk assessment and risk reduction efforts of their customers. Moreover, appropriate economic incentives to encourage vendor cooperation need to be explored.

Future work will investigate how to put the V-RATE method on a more scientific basis. Ongoing development of V-RATE may provide input into a model similar to the Capability Maturity Model® that would help acquirers to more systematically assess a developer's maturity for producing COTS components for survivable systems. More rigorous foundations will require quantitative measures of a system's capability to survive malicious attacks, and ways to measure the contribution of a given COTS product (or set of COTS products) to promoting or obstructing that capability. This must include the ability to measure the impact on system survivability of interactions among multiple COTS components.

We also plan to incorporate the V-RATE criteria into suitable system development lifecycle models, such as the Spiral Model. The process of refining a survivable architecture using COTS products is inherently iterative. Unacceptable product or vendor risks found during one iteration may require backtracking

® Capability Maturity Model is a registered trademark of Carnegie Mellon University.

to a previous iteration to incorporate different vendor or custom products within the architecture. The iterative, risk-driven nature of the Spiral Model makes it particularly appropriate for incorporating the V-RATE method.

Our plan to incorporate the V-RATE criteria into software development lifecycle models will require us to take a close look at the concept of *open-source software*, which has been gaining increasing acceptance over the past few years. Open-source software provides access to the source code of a product for little or no cost, thereby encouraging the programming community to read, modify, and redistribute the source code, potentially leading to rapid evolution and improvement. We will investigate the suitability of open-source components for the design, development, maintenance, and evolution of survivable systems, in the context of the V-RATE criteria and integration with more traditional COTS components.

Finally, we plan to apply V-RATE to real-world, mission-critical systems. Such case studies will help us to fine-tune and validate the method, and demonstrate its use within a realistic lifecycle process. These studies will also help us to understand the risks associated with using COTS components for specific system missions. The details of the application of V-RATE (such as the specific evidence that needs to be gathered) may differ for different domains (e.g., military mission-critical systems, e-commerce systems, and financial systems). Since survivability is heavily dependent upon the context of the mission, understanding these differences is critical to V-RATE's successful application.

We intend to focus our research activities in these areas and encourage others to do the same. Only then will we be able to determine with assurance when and how we can use COTS components to build survivable systems.

Acknowledgment

We would like to express our gratitude to Dr. Carol A. Sledge, a colleague at the Software Engineering Institute, for her valuable comments and suggestions based on her review of an earlier draft of this paper.

References

1. R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*, pages 527–529. John Wiley & Sons, 2001. 227
2. V. R. Basili and B. Boehm. COTS-based systems top 10 list. *IEEE Software*, 34(5):91–93, May 2001. 216
3. L. Brownsword, P. Oberndorf, and C. Sledge. An activity framework for COTS-based systems. *Crosstalk: The Journal of Defense Software Engineering*, 13(9), September 2000. 216
4. Common Criteria Implementation Board. *Common Criteria for Information Technology Security Evaluation, Version 2.1.* Number CCIMB-99-031. August 1999. See: http://csrc.ncsl.nist.gov/cc/. 226

5. C. Cowan, P. Wagle, C. Pu, S. Beattie, and J. Walpole. Buffer overflows: Attacks and defenses for the vulnerability of the decade. In *DARPA Information Survivability Conference and Expo (DISCEX)*, Hilton Head, SC, January 2000. IEEE Computer Society. 217
6. R. J. Ellison, D. A. Fisher, R. C. Linger, H. F. Lipson, T. A. Longstaff, and N. R. Mead. Survivable systems: An emerging discipline. In *Proceedings of the 11th Canadian Information Technology Security Symposium (CITSS'99)*, Ottawa, Ontario, May 1999. Communications Security Establishment, Government of Canada. See: http://www.cert.org/research/ for additional papers on this topic. 217
7. J. Froscher and M. Kang. A client-server architecture supporting MLS interoperability with COTS components. In *Proc. MILCOM 97*, Monterey, CA, November 1997. 218
8. S. A. Hissam, D. Carney, and D. Plakosh. *DoD Security Needs and COTS-Based Systems*. SEI Monographs on the Use of Commercial Software in Government Systems. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, September 1998. See: http://www.sei.cmu.edu/cbs/papers/monographs/dod-security-needs.htm. 216
9. R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. F. Lipson, and S. J. Carriere. The architecture tradeoff analysis method. In *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems*, Monterey, CA, August 1998. IEEE Computer Society. See: http://www.sei.cmu.edu/ata/ for additional papers on this topic. 217, 220
10. U. Lindqvist and E. Johnson. A map of security risks associated with using COTS. *IEEE Computer*, pages 60–66, June 1998. 216
11. H. Lipson and D. Fisher. Survivability – A new technical and business perspective on security. In *Proceedings of the New Security Paradigms Workshop*. ACM, September 1999. 217
12. N. R. Mead, H. F. Lipson, and C. A. Sledge. Towards survivable COTS-based systems. *Cutter IT Journal*, 14(2):4–11, February 2001. 217