THE IMPLICATIONS OF COTS VULNERABILITIES FOR THE DOD AND CRITICAL U.S. INFRASTRUCTURES: WHAT CAN/SHOULD THE DOD DO?

Robert H. Anderson, Richard O. Hundley
The Implications of COTS Vulnerabilities for the DoD and Critical U.S. Infrastructures: What Can/Should the DoD Do?

Robert H. Anderson, Richard O. Hundley
RAND Corporation, Santa Monica CA

1. THE PROBLEM

Critical systems on which the security and safety of the United States depend are increasingly based on commercial off-the-shelf (COTS) software systems. The trend toward use of COTS software is driven by several factors, including significantly lower procurement cost. COTS products are often highly complex, some of them involving tens of millions of lines of code, so that no one knows their contents and behavior in detail. These products are developed by multinational corporations, often involving programmers and others who are not U.S. citizens and may have other allegiances. The security situation for COTS software is not improving; market forces tend to favor functionality over security and reliability. The U.S. Department of Defense has little power to control or manage the development of COTS software, because its “market share” is small compared with the total U.S. or international mass market.

For the purposes of our discussion, we focus on vulnerabilities exploitable by deliberate perpetrators who may harm the: (1) confidentiality; (2) integrity; or (3) availability of essential information and functionality for U.S. defense or critical infrastructure operations. However, we note that it is relatively easy for an adversary to disguise denial-of-service attacks, and perhaps data corruption, as accidents -- easier than in physical environments, where traces of plastic explosives or other forensic evidence can often be found. In addition, many of the kinds of remediation that would be appropriate for accidental events would also be appropriate for intentional ones. For both of these reasons, we suggest at least ensuring that any effort to address deliberate attacks be coordinated with efforts to improve the robustness and quality of COTS systems in general: treating accidental and malicious software flaws too separately would be counterproductive.

We elaborate on aspects of the “COTS problem” below, then discuss potential solution strategies that may aid in ameliorating this vulnerability in critical U.S. information systems. We then give our view of the “non-U.S. citizen” or “offshore worker” software problem. Finally, we conclude with the outline of a possible research agenda addressing the problem of COTS vulnerabilities.

a. There is growing reliance on COTS products

Plans for next-generation command and control systems within DoD depend heavily on a “common operating environment” (COE) that is based on commercial operating systems, such as Sun Solaris, HP-UX, and Windows NT Server. Other services within the COE are provided by commercial web browsers, such as provided by Netscape, office automation software such as Word, Powerpoint and Excel, and database management systems such as Oracle. Defense communications relies heavily on NIPRNet and SIPRNet, using Internet protocols depending on routers and switches whose software is provided by Cisco, Bay Networks, and other commercial providers. Most telephone switches comprising the public switched network (on which most of DoD communications is carried) are run by software that is primarily UNIX-based, although there may be a trend toward use of Windows NT Server.
The trend toward dependence on COTS software in critical systems is compelled by several forces: (1) off-the-shelf systems are much less expensive to procure than developing special-purpose ones; (2) special-purpose systems tend to become "frozen" and maintained at a particular state, whereas market forces and competition tend to stimulate COTS products to become upgraded and near a commercial state-of-the-art. By using COTS, one can "surf the wave" of continuing product improvements; (3) COTS systems tend to come with extensive documentation, training, and education courses -- either provided by the developer, or by a congeries of third-party suppliers filling a perceived need; (4) many persons coming into DoD or becoming employed by critical infrastructure providers are already familiar with common operating systems (UNIX, NT, ...) and application programs (Netscape, Word, Excel, ...), reducing the training time required. It is not clear whether all of the above factors truly favor COTS (some of the savings may be specious when all costs are considered). However, we believe all of the above forces will continue unabated, so that COTS solutions will remain attractive -- at least without more thorough cost/benefit analysis, and possibly even if such analysis is carried out.

This trend to more and more reliance on COTS products is not unique to the DoD. Many critical U.S. infrastructures (e.g., telecommunications networks, the power grid, the financial community, etc.) are going in the same direction -- largely for the same compelling reasons.

b. Typical COTS products are very large and complex

Consider one example of complexity: The total Windows NT 4.0 memory footprint is roughly 500 megabytes; NT 5.0 will be roughly one gigabyte. NT 4.0 comprises about 18 million lines of code; NT 5.0 will have over 50 million lines. (This figure includes everything that ships within the shrink-wrapped box.) At least one million lines of code run within the NT kernel's address space. The Windows NT development team has about 1,500 members. Device drivers for NT, many of them supplied by small third-party companies, run in the same address space as the NT kernel. Although Microsoft reviews device drivers for "quality" before "signing" them, no one checks for evil code per se.

Although we stress the breathtakingly large size and complexity of NT above, contemporary browsers, with their e-mail, Java, and many other appendages, are rapidly becoming more complex. Similarly with office automation programs that can execute hidden "macros" when a document is opened, closed, or many other operations take place.

No one really understands everything that's contained within these complex COTS systems. "Easter eggs" are often discovered after shipment: distinctive displays (often giving credit to the system developers) that can be accessed by special, arcane key or mouse click combinations and are usually (thus far) benign. The latest Excel spreadsheet program contains a flight simulator program hidden within its code. It is not difficult to hide malicious code within typical COTS products.

In general, highly complex COTS software is unreliable; its behavior is not well specified, well documented, or well understood (even by the developers who wrote it). It is difficult to determine whether evil lurks within it; and complexity introduces new failure modes, such as macro viruses. The above statements are not true of all COTS software. However, they are true for some of the products underlying the DII COE and related DoD critical software systems.
c. Non-U.S. citizens and offshore workers are heavily involved in COTS software development

Sun Microsystems maintains a system development office in Russia. The staffs of many companies in Silicon Valley, Redmond (i.e., Microsoft), and elsewhere rely on recent immigrants who are not (yet) U.S. citizens. Some key encryption software products are being developed overseas, then imported to the U.S. and elsewhere, to avoid U.S. encryption export policy constraints. Much software development work, including Microsoft’s (and several large U.S. financial institutions), is being performed at sites in India. Due to the multinational, global nature of contemporary companies, it is extremely unlikely that DoD or other critical infrastructure providers could demand, or rely exclusively on, software that is developed exclusively by U.S. citizens, or within the geographic boundaries of the U.S., or other such restrictions.

This issue is dealt with further in section 3, below.

d. Current market forces and the product liability situation do not promote greater security in COTS products

The market for COTS software clearly favors functionality over security and reliability. Security is enhanced by using stable products that can gradually be “scrubbed” and tuned for security and reliability. Software developers compete in the market by developing and rushing to market new versions of their products that have additional functionality. If one were to thoroughly analyze a complex COTS products for security flaws – e.g., in order to certify that it has a high degree of security – just about the time that process was completed, a new version of that product would be announced, and the process would have to start over, since many subtle changes to the code would have been made between product releases, any one of which could affect its security.

At the current state of the law, DoD does not have much leverage in using legal mechanisms to increase levels of security and reliability in software products. The Uniform Commercial Code (UCC) allows software manufacturers to limit their liability with shrink-wrap licenses; so far, the courts have upheld the validity of these contracts and the disclaimers they often include.¹ In fact, liability laws cause developers not to promise any level of security or safety in their product, since any such promises might then become legally enforceable.

Tort law is also generally inapplicable for DoD use of COTS for critical systems, since there is no liability in tort for injury caused or exacerbated by a product when the product is used in an unexpected manner. In order to prove strict liability, the plaintiff must only establish that the product does not meet the ordinary consumer’s reasonable expectations of safety; but for its critical systems and its novel security requirements, DoD is clearly not the ordinary consumer. The other tort possibility stems from the producer’s negligence. A manufacturer is negligent if it fails to practice ordinary care in designing a product. But the standards of “ordinary care” are not hard for a producer to meet. Consider the wording in a popular text on tort law:

“The seller is entitled to expect a normal use of his product, and is not liable when it is put to an abnormal one... On either basis, the seller is not liable when the product is materially altered before use, or is combined with another product which makes it dangerous, or is mishandled, or used in some unusual or

¹ See, for example, ProCD v. Zeidenberg. 86F.3d 1447 (3rd Cir. 1996).
unforeseeable way... In the ordinary case the seller may also expect a normal user of the product; and he is not liable where the injury is due to some susceptibility or idiosyncrasy which is peculiar to the plaintiff." 2

The Federal Acquisition Regulations (FAR) related to purchase or license of COTS software do not give the government or DoD special rights or privileges.3

e. The DoD lacks sufficient market strength to compel greater security in COTS products

DoD expenditures in 1998 for COTS operating system software are approximately $248 million, compared with a U.S. market of about $31 billion and international market of $62 billion. That is, DoD accounts for less than 1% of the U.S., and about 0.4% of international markets for operating system software.

Similarly, 1998 figures indicate that DoD purchases about $500 million in COTS application software, compared with a U.S. market of $50.4 billion and an international market of $89 billion. DoD's "market share" is therefore again less than 1% of the U.S. market, and less than 0.6% of the international market.4

In general, DoD expenditures for COTS software tend to comprise about one third of U.S. federal government expenditures.

With "market clout" of less than 1% of even U.S. national COTS software markets, DoD -- or even the U.S. government as a whole -- is not in a position to dictate or strongly influence the features and facilities of COTS operating systems and application programs, especially when most of those are supplied by multinational firms aiming at the larger international market.

---


3 FAR, paragraph 29.961.60, section 12.212 reads as follows: "Computer software. (a) Commercial computer software or commercial computer software documentation shall be acquired under licenses customarily provided to the public to the extent such licenses are consistent with Federal law and otherwise satisfy the Government's needs. Generally, offerors and contractors shall not be required to -- (1) Furnish technical information related to commercial computer software or commercial computer software documentation that is not customarily provided to the public; or (2) Relinquish to, or otherwise provide, the Government rights to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation except as mutually agreed to by the parties. (b) With regard to commercial computer software and commercial computer software documentation, the Government shall have only those rights specified in the licenses contained in any addendum to the contract." The DoD supplement to the FAR does not elaborate on, restrict, or comment on section 12.212.

4 These figures are supplied by INPUT of Mountain View CA, based on OMB Federal Agency IT Spending Budget Estimates, and their internal Federal IT market research program.
f. Widespread COTS use incurs potential vulnerabilities from homogeneity and transparency

Part of the “COTS problem” in general is that use of mass-produced COTS widely throughout DoD and infrastructure systems promotes homogeneity (essentially everyone is using the same software) and transparency (everyone — i.e., all of the “bad guys” -- knows what is in the code) within major system architectures (such as based on the DII COE). A forthcoming RAND report on the concept of a minimum essential information infrastructure discusses these and 18 other categories of potential vulnerabilities in system architectures.\(^5\)

Given the above aspects of the COTS vulnerability problem, how do we reach a deeper understanding of the significance and potential impacts of the problem? To do so, we need a better understanding of: (1) the nature and magnitude of the damage that malicious code in COTS products could cause DoD; and (2) the relative risks involved across the spectrum of DoD uses of COTS (i.e., what are the more risky DoD uses of COTS, the less risky DoD uses of COTS).\(^6\)

Two overall approaches to developing this better understanding are:

1. “Bottom up” from incidents: Systematically collect information on incidents and events related to security and safety flaws in COTS software products. Classify these events in various ways: How many flaws were accidental, vs. deliberately implanted? What were the actual consequences, and how bad could they have been if that flaw were systematically and deliberately exploited? From the statistics gathered, determine the magnitude of the risks, and study the potential to locate and eliminate these types of software flaws.

This approach suffers from being dependent on flaws and incidents so far encountered; it may miss opportunities for malicious embedding of code that have not yet been exploited, or at least discovered. Its advantage is that it is based on real data, not hypotheses.

2. A taxonomy from first principles: Create a taxonomy of means by which malicious software can become embedded in COTS delivered products. (For example: by gimmicking the compiler to embed flaws in binary code emitted; by implanting malicious “Easter eggs” in the same manner that benign ones are now implanted; by

---


\(^6\) Regarding the latter question, the relative risks involved in various DoD uses of COTS, we can make the following initial observations:

- Use of COTS operating systems entails more risk, in general, than the use of COTS application programs, since operating systems usually establish the overall security environment in which application programs function -- although “bad” software in application programs can certainly cause harm to the DoD.

- Some areas in which DoD uses COTS (e.g., GCCS, GCSS, NIPRNet and SIPRNet routers) entail more risk, in general, than other areas (e.g., office automation), since they support more critical or widespread DoD functionalities -- although “bad” software can harm the DoD wherever it occurs.
creating areas of data within the program that can become executed, resulting in adverse program behavior.) For each COTS security risk, assess the damage potential, and determine what could be done to counter this attack. Assemble the results into a “COTS user’s guide” that explains the risks so users can decide the level of risk they are facing.

With this second approach, one may never be sure that a taxonomy has been created that covers the subject adequately. How confident could we be in the resulting assessments? Has some important “attack mechanism” been missed?

Obviously, the “bottom up” and taxonomy approaches complement each other. Both should be pursued.7

For further discussion of risks associated with using COTS software, see a recent article by Lindqvist and Jonsson (1998).8

---

**Problem summary:** It is possible -- even likely -- that malicious code can be hidden within major COTS products that are becoming foundations of our DoD command and control systems, and commercially-provided U.S. critical infrastructure systems. Such malicious code would be extremely difficult to detect. We do not have as yet good estimates of the amount of damage or denial of service such malicious code could cause. In addition, adversaries can exploit inadvertent vulnerabilities (bugs, flaws) rather than needing to implant malicious code.

2. CANDIDATE ELEMENTS OF A SOLUTION STRATEGY

We list some specific elements of a solution strategy below. Not all of them are equally practical, or even desirable. We attempt to list all elements that may have some merit, so that they may be compared and discussed. The specific elements listed can support a variety of overall solution strategies.

---

7 A third approach would be to assume that no significant COTS software can be trusted. Under this assumption, one would develop generic procedures to transform COTS into a more secure COTS (SCOTS?). This could include various inspections of the code, “wrapping” it in some sort of protection, providing operational work-arounds in case mischief happens, and so on. This third approach seems practical, but it may work at cross purposes to the whole COTS philosophy (mass-produced, interoperable, at the state-of-the-art, inexpensive, ...).

a. Regulatory/Procurement Initiatives

(1) Require DoD access to COTS source code used in critical applications

This initiative would perhaps restrict the number of COTS products upon which critical infrastructures could be built, but it is not an onerous restriction: Software source code could be provided to the government under a type of escrow with limits on the number of persons or agencies having access, prohibitions on further dissemination of the source code, and so on. This initiative would permit various “red teams” to look for attacks that the code permits, and use of various semi-automated toolkits to scan for abnormal patterns within the code. Even proprietary source code such as that for Windows NT has been disseminated to university research groups and others, so there is precedent for such limited access.

If source code could not be universally obtained, it may be useful to distinguish between operating systems (OSs) and applications (apps). Bad security flaws in OSs may have much broader ramifications, and be harder to fix or work around, than flaws in apps. Source code access should therefore be required, at minimum, for operating systems, and might be foregone if necessary in some cases for application programs.

Our initial assessment is that the DoD should push this element for all it’s worth. The DoD should not give up easily regarding access to source code, particularly for key operating systems.9

One topic requiring further investigation in this regard is the details of DoD’s current source code access situation: To which key COTS systems does DoD lack source code access today? How serious is this lack of access? What are the obstacles preventing access? Another topic requiring further investigation is modalities facilitating greater DoD access to source code (e.g., licensing agreements, non-disclosure agreements, etc.) The views of the various COTS providers on this subject obviously should be solicited as part of this latter investigation.

(2) Buy only from U.S. sources

We list this option for completeness, but it seems quite impractical. Is Sun Microsystems a “U.S. source,” with its development offices in various countries? Is Microsoft, with employees on various types of temporary work visas? Given that the DoD has limited market “pull,” it is unlikely that its procurements could induce software development companies to purge their development staffs of non-U.S.-citizens.

---

9 Besides being an important element of any strategy to make COTS more secure, access to source code also enables or facilitates several of the technical initiatives helping the DoD to live with insecure COTS which we mention later.
(3) Require or encourage better group software development practices

Group software development practices typically have some mechanism for making sure that a given piece of code is "checked out" to only one programmer at a time. However, there are a number of additional practices that are needed in order to make group software development/maintenance activities safer: A) Environments that allow project managers to designate which individuals are allowed to check out various pieces of code, with appropriate authentication. B) A permanent (long term, non-forgeable) record of changes to the code, including a record of who made each change. It might be somewhat harder to buy insiders on a software development team if they know that there is a permanent record of everything that they've done.

We note that it is possible using tools like RCS and CVS to monitor who made what changes to a code base.\textsuperscript{10} This is pretty straightforward. It's also possible to limit developers' access to only certain files, rather than an entire code base. It may therefore not be unreasonable to require vendors (e.g. of COE operating systems and applications) to implement rudimentary source code revision auditing. Many development organizations do this already.

Given DoD's limited buying power (noted earlier), DoD may not be able to "require" these better software development practices on the part of COTS providers. But it certainly should try to find ways to effectively encourage such practices. Finding such effective ways is obviously a topic requiring further investigation.

b. Certification/Standards Initiatives

(1) Have NSA, NIST or a non-governmental organization establish standards and a certification process for COTS used by DoD

Just as a security process was established for "Orange Book" certifications of operating systems, it might be possible to have new standards and certifications for COTS software used in critical systems. However, the certification process takes time. It might often be the case that just when version 4.0 of a product becomes certified for use, version 5.0 would be released, starting the process all over again. Also, it is unclear whether the diversity of operating systems and application programs have enough commonality to allow standard certification processes to be developed and executed.

It is tempting to look at the commercial Underwriters Laboratories "stamp of approval" as a model for a possible non-profit or commercial service that would investigate COTS products for flaws, and issue certifications, perhaps at varying levels of assurance. The DoD procurement process for COTS software could be strongly biased in favor of products having this third-party stamp of approval.

We note, however, that such a certification process doesn't, in itself, change the development of COTS. Such certification would need to be combined with some leverage on the manufacturers - otherwise it would have no effect. Bringing about such a third-party certification process with broad appeal and effective market clout will obviously take time. This is obviously an area requiring further study. Part of this study should focus on

the Underwriters Laboratories model: Which features of this model are transferable to the COTS problem? What are the similarities and differences?

c. Market Forces Initiatives

(1) Join forces with others to increase market clout

It might be possible for DoD to join forces with commercial suppliers of critical infrastructure systems (e.g., the public telephone companies; suppliers of routers and switches; energy and pipeline companies; air and rail transportation firms; banks) to create a "critical mass" of purchasing power that could influence the design and production of more secure and reliable COTS products. We do not, at this time, have figures on how large this combined "secure infrastructure" market might be, but it may well be 10 or even 20 times DoD’s purchasing power – an amount that could begin to influence at least some important market suppliers of COTS operating systems and application programs in widespread use.

In addition to having greater purchasing power, such an "infrastructure market" may also have greater political power insofar as future laws and regulations regarding COTS reliability and security are concerned.\(^\text{11}\)

We believe that commercial suppliers of critical infrastructure systems are obvious allies of the DoD insofar as COTS security and reliability are concerned, and that joining forces with them could provide a big payoff for the DoD, over some period of time. How exactly for the DoD to go about joining forces with these infrastructure providers is obviously an area requiring further investigation. Among other things, this investigation should consider how big various joint markets might be, which infrastructure providers would be DoD’s strongest allies in any such joint venture, and what are the best ways for DoD to enlist these allies.

d. Legal/Liability Initiatives

(1) Change the legal liability situation

We discussed the legal and liability situation in section 1. It would seem that there is little leverage available to DoD at present from legal recourses.

We note, however, that there is some growing feeling that operating systems are a "natural monopoly". That is, market forces will tend toward a monopoly in this area regardless of artificial barriers established to prevent that. Some reasons for this trend might be: (1) compatibility: Everyone wants to exchange files, formats, mail, disks, video, etc. etc. with everyone else, and not have incompatibilities; (2) scale: After a certain size, it's much more efficient to just add functionality to an existing system than to create one from scratch that has to meet all the existing de facto standards, drivers, interfaces, etc. etc. that people want for compatibility; (3) training: Once "everyone" knows how to use some particular interface (e.g., QWERTY, Windows 9x), no matter how ugly, it's easier to adopt it than train them on something really new. If a "natural monopoly" is in fact the case, then so be it -- but then the U.S. government might do what is typically done to

\(^{11}\) The average citizen worries more about the reliability and security of the infrastructures supporting his daily life (e.g., the power grid, telecommunications network, etc.) than he does about the DoD information systems/networks. What the average citizen worries about, sooner or later his elected representatives worry about as well.
monopolies: regulate it. That way, the government would regain at least some control of the operating system situation.

A distinction with legal implications is that between selling and leasing software (and other systems). The trend in software (gradually gaining strength over the past 15 years) has been away from sale (which is governed by rather strong legal precedents, such as the principle of “first sale”) toward leasing (which, as a contractual agreement between private parties, can be almost anything the parties agree to). Revisions to the UCC are now relaxing the actions that are required to bind a party to a contract (by allowing “clicking” or opening a shrink-wrap package to substitute for signing): this is merely the tip of the iceberg, facilitating the widespread use of leasing contracts, which are largely unconstrained by law. It is this trend toward leasing that (so far) allows vendors to escape any responsibility for their products’ working at all, let alone causing unforeseen damage.

There hasn’t (to our knowledge) been a test case charging gross negligence/malfeasance against any software company. We suspect that the courts might throw out any anti-liability clauses in the license agreement, especially if an employee of the software company inserted malicious code into a program. Of course, in situations involving national security, the possibility of collecting compensatory damages after the fact is not really very satisfying.

Any change in the legal situation significantly increasing the liability of COTS providers insofar as security breaches are concerned would obviously further the cause of reducing COTS vulnerabilities. This could have a big payoff, but clearly requires a major, long-term effort. The best ways for the DoD to approach such an effort -- and who to join forces with in so doing -- require further investigation.12

e. Technical/Procedural Initiatives

(1) Conduct R&D leading to enhanced “toolkits” capable of analyzing COTS source or binary code

 Millions of lines of source code cannot be scanned manually. One must rely on semi-automated tools and aids to uncover code that may be abnormal or malicious in some manner. It is likely that NSA or other agencies have some toolkits to aid in such processes. This existing “state of the art” should be investigated, and enhanced in any way possible to make the examination of large quantities of COTS code more timely and practical. This initiative would work in conjunction with initiative a(1) above, requiring DoD access to source code used in critical applications.

These toolkits are a component of a larger initiative that is very important -- perhaps one of the best solution tactics currently known: the serious use of "red team" activities. One of the best ways to find how bad the COTS problem is, is to have 20 great programmers/hackers study a COTS package, bang on it, decompile it, study the source code (if available), and in general work with it to see how many flaws or vulnerabilities they find, with what levels of seriousness. (I.e., what their implications might be if exploited in operational systems using this COTS package.) This would be one of the most cost-effective approaches to learning more about the depth of the problem, and their

12 Lawsuits arising out of the Year 2000 problem could well spawn legal developments helpful in increasing COTS developers’ responsibilities and liabilities for their products. The DoD should monitor any such developments as part of its long-term legal strategy in this area.
activities would be enhanced by a "toolkit" of techniques, scans, pattern recognizers, decompilers and disassemblers to aid in their work.

Note, however, that red team activities are most effective in discovering inadvertent programming errors that result in security vulnerabilities. In order to prevent evil from entering a code base, it is best to use code reviews, source code controls, and auditing.

"Red teams" need not all be formally assembled. Informal hacker attacks occur spontaneously (and for free) all the time. We suggest a variation of the red team approach, in which analysts (possibly working with hackers) attempt to show how observed hacking activities could have or might be used to produce more serious, targeted effects. If particularly scary scenarios are unearthed in this way, they might then be tried out to prove the point (if the point is not obvious enough to those concerned), but it may be that hiring hackers to do what they are already doing for free is unnecessary and perhaps unenlightening, as long as the implications of existing hacking attacks are studied carefully and systematically.

One interesting component of a toolkit might be a program "profiler." The typical profiler generates information about the execution of a program -- specifically, what fraction of the CPU cycles are expended within various modules. Profilers can in principle generate a wide range of other types of information, e.g., a list of all files opened, network activity, etc. This information might be of some value for identifying suspicious behavior in an application. We note that within the Sun Solaris (Unix-based) operating system, it is currently possible to profile code without modifying the source and obtain information on how many times each basic block executes. There is also a run-time tool "truss" on System V UNIX (e.g. Solaris) that traces all system calls made by an executable. Effective use of most profilers requires access to source code, and use of a compiler of one's own choosing. (A caution, however: Profiling and black-box testing are good ways of flushing out bugs, but they won't likely uncover evil code carefully hidden in highly complex COTS software by a clever and resourceful adversary.)

Another component of a toolkit involves various types of testing of the COTS product. Three important types of tests are: (1) "black box," in which the product is treated and exercised strictly through its input/output interfaces, to observe its behavior under a variety of circumstances; (2) system-level fault injection, to deliberately perturb the system with non-standard inputs in an attempt to check for abnormal behavior in abnormal circumstances; and (3) operational system testing, in which the COTS product is exercised and tested in the larger environment within which it will reside during operational use. See a recent article by J. Voas (1998) for further details on these testing distinctions and their uses.13

We expect that NSA is currently investigating toolkits such as those described here. However, more can probably be done in this area -- and more can certainly be made available to personnel throughout DoD concerned with the security of the COTS software they operate. An investigation is required outlining the elements of an R&D program leading to more effective toolkits.14 This effort should use the current DoD (i.e., NSA) state-of-the-art as its point of departure.

---


14 The Digital Millennium Copyright Act, currently under consideration in Congress to implement treaties signed at the World Intellectual Property Organization's (WIPO) Geneva conference on digital information
(2) Add security controls to COTS software

We earlier mentioned as an overall approach to the COTS problem that one should “assume that no significant COTS software can be trusted.” This element implements that approach by studying means for “wrapping” or otherwise encapsulating, filtering or isolating critical COTS software programs. DARPA has funded a substantial research program based on the concept of “wrapping” untrusted system components.

We believe this approach is an important component of a COTS security strategy, and that much more can be done in this area. What’s needed here is: (1) an inventory of existing techniques and products (e.g., wrappers, firewalls, filters) to identify gaps in the current menu of available security controls; (2) an outline of the R&D program required to fill those gaps; and (3) a specification of the elements of a DoD program requiring use of such security controls in all “critical” COTS applications.

(3) Improve procedures for responding to security flaws

Part of a solution strategy would be to find a way to make the "cycle time" of fixing or blocking a bug, once known, faster than the cycle time of perpetrators exploiting that bug or flaw. Note that the total cycle time includes obtaining a bug fix from the COTS manufacturer (or else creating a work-around or reconfiguration around the problem oneself), disseminating the fix to the field installations, and insuring that the fix is installed throughout DoD. One possible area for research concerns ways of fixing security problems “on the fly.” Though this is routinely done in the mainframe world, it is still relatively rare in the workstation/desktop/server world. If a system must be taken down and rebooted to make a patch, then users will object to having this happen too often or during peak hours, leaving a window of opportunity for attackers. Technical solutions to this problem should not be all that difficult.

We believe that much more can be done regarding the transmission and implementation of security fixes; DoD needs to work with the COTS vendors on this problem. A first step would be to study what procedures for rapid response DoD has control over, and how they can be improved. A second step would be to work with the COTS vendors on joint improvements.

(4) Consider disabling or deinstalling portions of COTS products

Many COTS operating systems and application programs contain “bells and whistles” not needed by critical DoD or infrastructure operations. For example, browsers run Java or ActiveX scripts; some applications create and maintain automatic Internet links to external information sources; application programs such as Word and Excel allow execution of macros. Those additional facilities create additional opportunities for security-related flaws. It is often possible to disable selected features in these products without the need for reprogramming, by configuring the product accordingly. DoD might provide a “configuration service” that provides default secure configurations for popular COTS products, so that the default “out of the box” settings minimize opportunities for security problems. In general, the simpler and less complex a product, the greater the chance for channeling it into acceptable and understood modes of behavior.\(^\text{15}\)

and copyrights in December 1996, could limit, impede or even prohibit the analysis of copyrighted source code. DoD should monitor developments in this area.
\(^\text{15}\) This strategy was suggested in an interview with John McDermid, University of York, UK, entitled “The Cost of COTS,” by Nancy Talbert, IEEE Computer, 31(6), June 1998, pp. 46-52.
This should be an important element of any DoD strategy to reduce COTS vulnerabilities. The first step should be to inventory the features of key COTS products used by the DoD that can be disabled by various configuration controls and identify the security implications and performance tradeoffs of disabling various of these features.

The various initiatives listed above fall into two general categories: those that try to make existing COTS products more secure, and those that allow us to live with insecure COTS products. The following table provides a summary of the security components above, and categorizes them using this distinction. Note that the technical initiatives tend to aim at the need to work with insecure COTS products, and the other initiatives concentrate on making COTS products more secure.

<table>
<thead>
<tr>
<th>Elements of a Security Strategy</th>
<th>Makes COTS More Secure</th>
<th>Works With Insecure COTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a. Regulatory/Procurement Initiatives</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) Require DoD access to COTS source code</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>(2) Buy only from US sources</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(3) Require better software development practices</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td><strong>b. Certification/Standards Initiatives</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) Establish certification process</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><strong>c. Market Forces</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) “Join forces” to increase market clout</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td><strong>d. Legal/Liability Initiatives</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) Change the legal liability situation</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><strong>e. Technical/Procedural Initiatives</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) Conduct R&amp;D on more effective “toolkits”</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>(2) Add security controls to COTS software</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>(3) Improve procedures for responding to flaws</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>(4) Disable portions of COTS products</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

*Solution Summary:* There are two basic approaches to “managing” this problem: making COTS used by the DoD more secure; and learning to live with insecure COTS. Things can be done in both of these areas. We have identified a number of candidate elements supporting each of these approaches. The specific elements listed can support a variety of overall solution strategies.