Security Considerations in Space and Delay Tolerant Networks

Stephen Farrell, Vinny Cahill
Distributed Systems Group
Department of Computer Science
Trinity College Dublin
{stephen.farrell, vinny.cahill}@cs.tcd.ie

Abstract

This paper reviews the Internet-inspired security work on delay tolerant networking, in particular, as it might apply to space missions, and identifies some challenges arising, for both the Internet security community and for space missions. These challenges include the development of key management schemes suited for space missions as well as a characterization of the actual security requirements applying. A specific goal of this paper is therefore to elicit feedback from space mission IT specialists in order to guide the development of security mechanisms for delay tolerant networking.

1. Introduction

Problems encountered on the Internet today show that security must be a consideration for all networking technologies. Although current space mission networks are significantly less threatened, proper consideration of current Internet threats and countermeasures can help avoid future pitfalls. We can therefore usefully leverage the Internet security knowledge-base built up over the last decade to analyze the security considerations for space missions, and in particular the generalization of those represented by delay tolerant networks (DTN).

Currently, the main venue for DTN related security work is the Internet Research Task Force's (IRTF) [1] Delay Tolerant Networking Research Group (DTNRG) [2]. Our primary objective here is to engender a discussion between space mission IT specialists and those defining security services for DTNs and similar networks.

As with any network, a DTN or space-mission network may use cryptographic data confidentiality and integrity services. However the lack of end-to-end connectivity and the potentially extreme asymmetries in terms of capability and connectivity mean that we need an unusual formulation for such services, where we not only have sources and destinations for data, but potentially different security-sources and security-destinations. We review current work [3, 4, and 5] on this topic and related open issues, for example, key management for high-delay environments where there is currently no clear solution available.

In addition to cryptographic security services, we have learned from the Internet to also pay attention to implementation issues, leading, for example, to firewalls separating the network into different security domains that offer assurance that only approved traffic is present in each domain. However, there is always a trade-off between "being secure" and "being usable" in such environments and the experience on the Internet is that it is generally possible to insert unapproved traffic into any security domain. We therefore also outline some implementation-related vulnerabilities, in particular those that may require specific consideration in space missions.

We also examine some of the potential trade-offs arising as more complex space based networks are more intimately connected to the Internet. For example, space missions require various ways to re-initialize spacecraft, practically all of which form can a denial-of-service (DoS) attack vector whose impact on a space mission could be catastrophic.

The significant benefit of reviewing security services as well as implementation issues is an increased awareness of the possible and actual security considerations arising as space missions make greater and greater use of the terrestrial Internet and associated technologies. Follow-on work could include a characterization of security requirements for future

2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06)
0-7695-2644-6/06 $20.00

space missions, based on current Internet threats but taking into account the issues described here.

The DTNRG is developing two main delay-tolerant protocols, the bundle protocol [6], that defines an overlay network and the Licklider Transmission Protocol (LTP) [7, 8] which is a point-to-point protocol. These form the baseline for the review presented here.

2. LTP Security

Since LTP is a point-to-point protocol its security considerations are simpler than for the bundle protocol, so we begin here. LTP is modeled on the CCSDS file delivery protocol [9] and provides similar services, but is defined in Internet "style" rather than the more "OSI" style of the CCSDS specification. Given that LTP is a point-to-point protocol one would expect that most security considerations could be taken care of at another layer, either above LTP or else in a MAC layer beneath. For this reason LTP does not define a confidentiality mechanism, but only a data integrity mechanism. The reason for including the latter is that LTP could be used in a broadcast environment (e.g. 802.11) where insertion of bogus segments would be a threat worth countering. In fact, the main security consideration with LTP is avoiding DoS attacks, and in particular off-path attacks. Where they are possible, off-path DoS attacks can be mounted from essentially anywhere on the Internet, are harder to trace and allow the attacker to increase the scale of the attack at will. An off-path DoS attack can be quite devastating for a terrestrial system, e.g. where LTP is layered on top of IP or UDP, and would certainly be so for a spacecraft.

Some basic LTP features were, in fact, designed to be robust in the face of such DoS attacks, for example, recommending use of randomized identifiers, rather than starting counters at '1' etc. And in order to make off-path attacks harder, LTP also includes a delay-tolerant cookie-mechanism that can be selectively turned on. This mechanism essentially creates "fresh" shared state between the communicating peers that is presumably unavailable to the off-path attacker. Both the data-integrity and cookie mechanisms are LTP extensions [4], and so are optional to implement.

3. Bundle Protocol Security

The bundle protocol, being an overlay network is arguably more vulnerable to attack than LTP, since every potential convergence (i.e. lower) layer protocol's security considerations can apply to the bundle protocol. We first review some of the main threats faced by bundle protocol nodes. Note though, that many of these threats are not specific to the bundle protocol and will be faced by any equivalent DTN protocol.

3.1. Threats

The first set of threats to consider are those coming from network elements that are not directly part of the DTN. As an overlay network, bundles may traverse multiple underlying network elements on each DTN "hop." Of course, any vulnerability in the bundle protocol can be exploited at any of those network elements.

DTN security must take into account the usual range of such potential exploits (masquerading, modified bundles etc.), but compared to most network protocols, as an overlay protocol, the bundle protocol probably represents an easier target. In particular, if it is possible to insert new bundles at such lower-layer "hops," then many DTN nodes will have to be capable of countering such insertions by, where possible, detecting and quickly deleting such spurious bundles.

Conversely, it is also possible to take advantage of lower-layer security services, but this won't necessarily be visible from the DTN layer, and requires coordinated administration in order to be really effective.

Due to the resource-scarcity that characterizes many DTNs, unauthorized access and use of DTN resources can be a serious concern. For example, if an unauthorized application were able to control some DTN infrastructure, perhaps by attacking a routing control protocol, then the resource consumption could be catastrophic for the network.

In addition to these threats, DTN nodes can unwittingly be used to assist or amplify resource consuming behavior for example by not detecting unplanned replays or other misbehaviors. In the Internet, there have been recent cases where so-called amplification attacks [10] make use of larger DNS records - sending a small packet to a DNS server (with a forged origin) can cause the server to send the large packet to a victim.

While DoS attacks can be mounted at any network layer, from physical to application, generally, when developing a new protocol, we should be attempting to do two things. Firstly, we should try to increase the effort to successfully launch an "off-path" DoS attack

by making it hard to "guess" valid values for messages, e.g., through using random values instead of counters for identifying messages. Secondly, we should make it easier to withstand "on-path" DoS attacks by providing a way to choke-off DoS traffic, e.g., by changing to a mode of operation where only fresh, authenticated messages are accepted, and all others are dropped.

In a DTN environment, the generally longer latencies involved will probably act to make DoS attempts more effective, so protocol developers and deployments need to explicitly consider DoS at all times.

As with all networks, security mechanisms will themselves create new DoS opportunities. So whatever services and mechanisms are defined for DTN security should also explicitly consider DoS. For example, mechanisms that involve looking up the status of some key (via some protocol to a key server) based on received messages create new DoS opportunities since such lookups consume resources on both the receiving node and the key server as well as on network nodes in between.

So long as DTN protocols include traffic generated as an artifact of other traffic, then the possibility exists that manipulation of (genuine, forged or modified) bundle content can be used to create a storm of unwanted traffic. Given a DTN operating sufficiently "close to the wire," such traffic could have serious affects.

In particular, the current bundle protocol includes various messages (bundle status reports and custody signals) that can be produced in greater numbers than the original traffic. For example, if a DTN node (or other network element) could modify a single "forwarding report," such that the forwarding of that report bundle will generate another bundle, and if the new bundles' forwarding report bit will also be set, and if the route that these bundles will take includes a loop, then an infinite bundle storm could result. Responding to this potential problem, the bundle specification now includes a requirement that no report should generate another report. This removes the vulnerability, though it does allow implementation bugs to easily recreate the problem. However, it may be wiser to entirely remove some of these supposedly useful reporting "capabilities".

In addition to the resource consuming threats, DTN applications can also be vulnerable to the usual threats against confidentiality and integrity, for example changing the intended destination or a bundle's control fields.

3.2. Security Requirements and Services

This section discusses some of the security requirements that were agreed as priorities during the development of the bundle protocol security specifications. [3, 5]

Traditionally, protocols tend to provide security services that are used either (or both) on a hop-by-hop or end-to-end basis. For DTN security though, we require that these services be usable also between nodes that are not a source or destination, but which can be in the middle of a route.

The example most commonly mentioned for this is where a sensor network uses some satisfactory lower layer security, and has some gateway sensor node, which is more capable and also periodically connected to the Internet, then we may wish to use DTN security services to protect messages between that gateway node and the other DTN sources and destinations on the Internet-side of the gateway. In the case of a confidentiality service, this is clearly useful since bundles which leave the sensor network could be encrypted (by the gateway node) for the final destination. In the case of, say a software download, new code might be integrity protected from the origin to the gateway which is able to check some relevant white or black lists or use some other software authorization scheme that cannot practically be used from a sensor node.

In order to define services that can be used in these ways, we distinguish between the sender of a bundle and the security-sender for an application of one of these services. Similarly, we can distinguish between the bundle recipient and the security-recipient (or security-destination) for a given application of a security service. Basically, the security-sender is the DTN node that applied the security service, and the security-recipient (or security-destination) is the DTN node that is the target for the security service - say the node expected to decrypt or do integrity checking.

The extent to which the various services can be combined for the same or different security senders and destinations is something that is still liable to change. However, we can state a requirement that this should be kept as simple as possible since unwanted complexity in this respect is highly likely to make a DTN harder to manage, and thus less secure. As we'll see below, while this is an easy requirement to state – it can be hard to meet in practice!

Having said that, there may still be good implementation-related reasons to distinguish (at the protocol field level) between uses of these services that are intended to be hop-by-hop (i.e. between this and the next DTN node), as opposed to uses of the services that are intended to be applied across multiple hops. Equally, a protocol might not need to make this distinction, and might only define e.g., one confidentiality service that can be applied multiple times for a single bundle with different combinations of security-sender and security-recipient.

There is one more example of the ways in which DTN security services differ slightly from more "normal" network security services that is worth mentioning here. When a message is authenticated using a digital signature, then in principle, any network element on the path can do some checking of that signature. If the message contains sufficient information (the supposed signer's public key or a resolvable reference thereto), then any node can at least check the cryptographic correctness of the signature.

However, this is typically insufficient to decide how to process the message, since in many environments, basically anyone could insert a public key and a signature thus producing a message that passes this test. So, in most cases, there is some additional check that the signer is authorized, either explicitly by checking that the signer's name or key is authorized for the purpose, or else implicitly by, for example, using a public key infrastructure (PKI) for this purpose (say via an extended key usage extension). It turns out that all practical ways to perform this authorization check are problematic in at least some DTN cases, either due to the lack of availability of an authorization server (say due to simple latency from the verifier to the relevant authorization server), or due to restricted node capabilities (say in the case of a sensor node).

In such cases, it may be sensible for some "bastion" node along the route to do the authorization checks and then to (again explicitly or implicitly) attest that the authorization checks have succeeded. Subsequent nodes may however, for either data integrity or accountability reasons, also validate the cryptographic correctness of the signature. The end result is a mechanism whereby the message has a signature plus some meta-data that are processed by the "bastion" node, whereas the signature is only partly processed by all subsequent nodes.

The bundle protocol provides the ability to encrypt protocol elements so that messages in transit cannot practically be read. However, the extent to which a confidentiality service should be able to be applied to any or all protocol elements is a somewhat open issue. In particular, whether or not source confidentiality should be provided is controversial, with the current protocol not easily supporting this feature.

Any confidentiality service implies a need for key management. However, we don't yet have a DTN-friendly key management scheme. So until we get such a scheme, we expect DTN deployments, will support pre-shared secret-keys, and/or known irrevocable public keys.

Similarly, DTN protocols should provide a way to apply an integrity check to a message so that the identity of the security-sender can be established, and so that changes in sensitive parts of the message can be detected. Again, this implies a need for key management which is not, so far, really met.

The bundle security protocol allows for fairly flexible combinations of application of the confidentiality and integrity services. However, it disallows some insecure combinations, e.g., a plaintext signature that is out of scope of a confidentiality service would allow plaintext guesses to be verified. So we don't want to offer that option, other things being equal.

Since forwarding even a single bundle will consume some network resources, every single DTN node must at least implicitly incorporate some element of policy-based routing. Of course, we do not expect that every single DTN node will be able to handle complex policy decisions. In fact, a DTN node might quite reasonably be programmed to forward all bundles received in a deterministic manner. So, although we require all nodes to implement some policy, that policy can be very simple.

Regardless of how simple, or complex, a node's support for policy-based routing/forwarding might be, DTN implementers should document the relevant aspects of the implementation. In the absence of such documentation, a node might be deployed in an inappropriate context, potentially damaging an entire network.

Some DTN nodes will however, be on boundaries of various sorts, whether they be network-topology related, administrative, networking technology related or simply a case where this node is the first that is capable of handling complex policy decisions. At one stage, these nodes were termed security policy routers, and were considered to be "special" nodes. Our current view though, is that all nodes are in fact policy

routers with some implementing policies which are more complex than others.

We do not, at this stage, require an interoperable way to transfer policy settings between DTN nodes. Such a system could perhaps be developed (though it is an extremely complex task), but pragmatically, for now, the development of a DTN-specific policy language and distribution framework is out of scope of the DTNRG.

DTNs themselves do not appear to generate many new types of policy-based controls - the usual ingress, egress and forwarding types of control can all be applied in DTNs. For example, some "bastion" node might insist on all inbound bundles being authenticated, and might add an authentication element to all outbound elements. So all the usual forms of control can and should be available for use in DTN nodes. No doubt, more will be identified as more DTN deployment experience is gained. The DTN specific policy controls identified thus far, and for which we would recommend support include:

- Time-to-live type controls where we consider the amount of time for which a bundle has been "in-flight"
- Controls to do with "strange" routes, such as those that loop
- Controls handling local or global information about resource constraints in the DTN (e.g., knowledge of a peer's storage capacity)
- Controls related to special types of fragmentation (e.g. so-called reactive fragmentation) which are defined in a DTN

DTN node implementations will also be required to control access to whatever DTN interface they provide so that only authorized entities can act as the source (or destination) of bundles. Clearly, this aspect of access control is an implementation, rather than a protocol issue.

Policy based routing, if not deployed appropriately, may inadvertently create bundle sinkholes. Consider the case in which a bundle is fragmented, and if one fragment of the bundle reaches a router whose policy requires it to see the entire bundle before forwarding, then all fragments of that bundle must also pass through that same router. If they do not, then eventually the fragment at our paranoid router will expire, and ultimately, the entire bundle never arrives at the intended destination. This is clearly a case to avoid - doing so, may however be difficult to arrange without good control over routes.

4. Open Issues in Bundle Security

The bundle security protocol is still very much a work-in-progress and there are some significant open issues remaining to be determined. In this section, we try to explain these sufficiently to get useful input to help with resolving these issues.

We have already mentioned that the level of flexibility to provide is an open issue – the additional flexibility potentially provided, particularly when combined with a planned bundle-in-bundle encapsulation mechanism, does allow for some interesting virtual private network (VPN) models to be set up, but the cost in terms of implementation complexity is high and complexity is the enemy of security. Without specific use cases, it is hard to know how to properly decide this issue.

The second major open issue is key management. There are currently no key management schemes that appear to suit many planned DTN deployments. Mostly, existing schemes require too many roundtrips (e.g., the TLS [11] handshake), or else effectively require hard-coded, hard-to-replace trusted keys to be deployed to all relevant end nodes. The latter is quite practical for experiments, and is currently the best we can do, but ultimately is not scaleable. While there is no expectation that space mission IT specialists can solve this essentially cryptographic issue, there may be some constraints or other requirements that might feed into a solution. For example, what level of "trust" (in terms of keying material) would actually be acceptable when missions from different agencies are interacting?

One possibility in this area could be to adopt a similar approach to that taken in the resurrecting duckling scheme [12] previously suggested for use in ad-hoc networks. In this scheme, nodes that have a "close encounter" with one another would take that opportunity to exchange keying material over what is presumably a relatively trustworthy channel. Those keys can then be stored for later use. If we assume that security-aware DTN nodes will have adequate storage, then perhaps it will, for some time, be possible to flood public keys for all "known" nodes whenever any two nodes have such an encounter. Of course, were such a scheme adopted, it would highlight the problem of key revocation and/or reputation created once there is a way to establish keys for DTN nodes.

A third issue, somewhat less serious, but one that has proven problematic in previous data integrity contexts is how to define a canonical form for bundles so that, e.g., the signer and verifier of a signature use the same

input bytes even if the bundle has been transformed somewhat in transit between them. While the current specification has what seems to be a workable proposal, there are indications [13] that a more compact encoding (a form of header compression) is likely to be required for deep space applications. There is therefore, the requirement that the canonicalization of the "normal" and compressed headers must produce the same result, or else security headers won't be usable across normal/compressed header gateways. As stated above, this is a quite tractable problem, but one that is hard to get right in all contexts, especially were the compressed form of encoding to vary from mission to mission.

5. Implementation Issues

It is probably true to say that in the 1990s many IT security specialists considered that most threats could be addressed through the use of the type of cryptographic services described above. That is no longer the case – there is now a broad recognition that implementation and deployment related problems are often more significant, even if countering those threats may also require the usual set of cryptographic based security services. We could give numerous examples, but will limit ourselves to just a few, that are, hopefully, more relevant for the space IT community.

Even networks that run according to very strict policies about fire-walling, use of VPNs etc. are now much more vulnerable due to host mobility. In a couple of reported cases [14, 15], source code has been leaked from IT companies, possibly due to malware running on an employee's home computer that was sometimes connected to the Internet, and sometimes to the corporate VPN. Such cases demonstrate that, in a complex network, it is at least quite hard to properly protect even your most precious digital possessions.

And the situation here is going to get worse, as, for example, the range of devices with both network connectivity and general purpose computing capability continues to grow with PDAs, USB sticks etc. all becoming potential attack vectors on an otherwise controlled network. So, even networks running with an "air-gap" may become nearly as vulnerable as those with less apparently rigorous security policies.

So, perhaps one should start from the premise that while one has to have sensible ingress and egress controls on networks, it is simply not possible to fully control the traffic entering the network and there will always be the possibility that a bad actor can insert some traffic into any part of our network. The security goal must then include damage limitation and recovery, rather than simply being one of prevention. One should also note that lower-layer encryption is not a useful counter to many of these threats, since they generally involve subverting an authorized host that is allowed to insert traffic into the encrypted part of the network.

In a space context, presumably the most controlled part of the network will be in space. However, from a security point of view, perhaps the deep space network (DSN) is the most critical part of the infrastructure, since it supports so many missions, and being on Earth, will be connected to the Internet. The extent to which DSN components will be accessible from the Internet will be a crucial factor in determining the level of exposure of other space mission nodes.

Similarly, one can expect a trend to allow instrument principal investigators (PIs) more direct control over their instruments during the mission. For cost reasons, there will probably be an unstoppable momentum to provide such control from the PI's home institution, (if that hasn't already happened). At some point, this leads to a dependency on the PI's home network security and so a weakest link argument will apply. This situation is of course, directly analogous to the source code leakage examples above.

One relatively common reaction to the current Internet security situation is that since there is no money to be made by hacking into "my system" (in this case space communications systems), that means that the threat level is much lower. While there is an element of truth here, it is, however, the case that innocent bystanders' systems are often attacked in order to make money elsewhere. In a recent court case [16] the defendants were charged with crippling a hospital network affecting intensive care systems. The defendants allegedly spread a program in an attempt to build a "botnet" so they could sell advertising. The adversary is no longer a vandal, but is becoming more and more capable, and perhaps, more willing to knowingly break laws in order to make money.

One of the most common forms of implementation problem seen on the Internet is related to code quality, and presumably affects all systems, including flight software, as well as ground-based systems. Whether or not this software is significantly less vulnerable due to significantly more stringent testing is not clear to the authors. For example, one could speculate that an architecture that has more statically allocated blocks of memory is perhaps an easier target for someone searching for an interesting buffer overflow.

It is also quite conceivable that flight software although extremely well tested against random errors (e.g., memory errors) or functional failures (e.g., partial loss of power) could remain highly vulnerable to directed attacks (e.g., integer underflow, buffer overrun or similar attacks). The argument here is similar to the argument that while scale-free networks (like the Internet) are very robust against random failures, they remain highly vulnerable to intelligently directed attacks [17]. If one believes that unauthorized traffic will, in fact, eventually penetrate to all parts of all networks, and if there are such vulnerabilities, then there is likely to be trouble ahead. How soon this may occur and with how much impact is of course hard to say.

Many space protocols were designed (and quite properly so, at the time) without considering the possibility that such unauthorized traffic could be present on the network. For example, the CCSDS proximity-1 protocol [18] includes a way for a peer to reset the clock of its communicating partner. An API exposing such a feature could fairly be called a "dangerous implement" [19], and is generally to be avoided in Internet security. With control over such an API, it would be fairly trivial to mount many attacks on a spacecraft, e.g., resetting its clock so that only the attacker knows when the spacecraft will in future be contactable.

Lastly, regardless of what development practices are adopted, be they high-level common criteria review [20], of closed-source right up to fully open-source technology, or anything in between, it appears that there is a requirement for some method of relatively frequent distribution of security patches. Microsoft's "patch Tuesday" [21] is probably the best known of these schemes. Whether or not some such scheme would work at all for space missions is an open question – clearly there is a long history of updating flight software during missions, but this is often done in a much more controlled and mission-specific manner. If DTN (or any networking technology) became ubiquitous, would that require that such updates be synchronized across many missions at once? And if so, would that be feasible?

6. Conclusions/Questions

We have given an overview of the current status of work on security for delay-tolerant networks and listed a number of open issues where we would like to get input from space mission IT specialists as to their requirements, if any, for each. As participants in the development of the bundle security protocol, the questions we would therefore ask the space IT community are:

- In the face of increasing connectivity between spacecraft communications and the terrestrial Internet, what types of VPN setup would make sense for the space missions?
- What are the relevant trust relationships existing between agencies and how might those be reflected in terms of a key management scheme?
- To what extent could current Internet security practices benefit future space IT communications systems?
- Would a "patch Tuesday" be desirable for space communications systems? If not, how should security updates be handled?

Our hope is that getting answers to these questions from the space IT community will ensure that the bundle security mechanisms will meet real mission needs.

7. References

[1] Internet Research Task Force, http://www.irtf.org/

[2] Delay Tolerant Networking Research Group, http://www.dtnrg.org/

[3] S. Farrell, S. Symington, H. Weiss, "Delay-Tolerant Networking Security Overview" draft-irtf-dtnrg-sec-overview-01.txt, March 2006, work-in-progress.

[4] S. Farrell, M. Ramadas, S. Burleigh, "Licklider Transmission Protocol – Extensions" draft-irtf-ltp-extensions-03.txt, March 2006, work-in-progress.

[5] S. Symington, S. Farrell, H. Weiss, "Bundle Security Protocol Specification" draft-irtf-dtnrg-bundle-security-01, March 2006, work-in-progress.

[6] Scott K., Burleigh, S. "Bundle Protocol specification" Internet-Draft, draft-irtf-dtnrg-bundle-spec-04.txt, November 2005, work-in-progress.

[7] Burleigh, S., et al, "Licklider Transmission Protocol – Motivation" Internet Draft, draft-irtf-dtnrg-ltp-motivation-02.txt, March 2006, work-in-progress.

[8] Ramadas, M., et al, "Licklider Transmission Protocol – Specification" Internet Draft, draft-irtf-dtnrg-ltp-04.txt, March 2006, work-in-progress.

0-7695-2644-6/06 $20.00 © 2006 IEEE

[9] CCSDS File Delivery Protocol (CFDP). Recommendation for Space Data System Standards, CCSDS 727.0-B-3 BLUE BOOK Issue 3, June 2005.

[10] Paxson, V. "An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks" Computer Communication Review, 2001, VOL 31; PART 3, pages 38-47, Association of Computing Machinery. ISSN 0146-4833.

[11] Dierks, T. and C. Allen, "The TLS Protocol - Version 1.0" RFC 2246 , January 1999.

[12] Stajano, F. "The Resurrecting Duckling – What Next?" 8th International Workshop on Security protocols, Lecture Notes in Computer Science, Springer-Verlag, 2000.

[13] Burleigh, S. "DTN for Space – JPL Implementation" DTNRG meeting presentation at IETF-65, http://www3.ietf.org/proceedings/06mar/slides/DTNRG-0.ppt

[14] Schneier, B. "Microsoft Source Code Leak" Crypto-Gram newsletter, 15 March 2004. http://www.schneier.com/crypto-gram-0403.html

[15] Broersma, M. "Cisco suffers source code leak" TechWorld, 17 May 2004. http://www.techworld.com/security/news/index.cfm?NewsID=1572

[16] Williams, C. "Three charged with Seattle hospital botnet attack" The Register, 14 February 2006. http://www.theregister.co.uk/2006/02/14/seattle_hospital_botnet/

[17] Zhao, L. "Attack vulnerability of scale-free networks due to cascading breakdown" PHYSICAL REVIEW E **70**, 035101(R) (2004).

[18] CCSDS Proximity-1 Space Link Protocol—Data Link Layer. Blue Book. Issue 3. May 2004.

[19] Bishop, M. "Robust Programming" course notes, http://nob.cs.ucdavis.edu/classes/ecs153-2000-04/Pdf/robust.fm.pdf

[20] NIST Common criteria web site http://niap.nist.gov/cc-scheme/index.html

[21] Goodfellow, B. "Patch Tuesday" TheTechGap.com, 11 January 2005. http://www.thetechgap.com/2005/01/strongpatch_tue.html