zapthink white paper

XML Threat Management

Problems & Solutions

ZapThink, LLC • 108 Woodlawn Road • Baltimore, MD 21210 • info@zapthink.com • www.zapthink.com

XML THREAT MANAGEMENT

PROBLEMS & SOLUTIONS

May 2006

Analyst: Jason Bloomberg

Abstract

XML has become the *lingua franca* of distributed computing—the language at the heart of an increasing percentage of traffic on networks today. As companies implement Service-Oriented Architecture, they build Services, and at the core of the definition of a Service is software that communicates via the exchange of messages—typically in the form of XML. An increasing amount of other network traffic leverages XML, as well.

Any IT security effort must therefore plan for and deal with XML threats, partly because of the explosion in the use of XML, but also because XML's power and flexibility lead to multiple threat categories, including structural threats like oversized and improperly structured XML messages, semantic threats that target the content of the XML message, and more. Companies must implement a comprehensive XML threat management solution that deals with all forms of XML threats, and that solution must operate at wire speed.

Table of Contents

I. The New Type of Threat

Of all the concerns on today's executives' plates, security is the most complex. Security is essentially the mitigation of risk, and risk is something every company has and wants to rid itself of. Risk, however, is a slippery concept, because it deals in possibilities: the possibility of a break-in, or a virus, or a loss of confidentiality, for example. Lowering risk means lowering the probability that such future events might occur; therefore, it's only possible to express the return on investment (ROI) on security investments in terms of how much a security failure would have cost if it had happened, even though it hasn't. Such ROI calculations are particularly difficult, because the greatest risk is one that is completely unexpected—one that by definition wasn't predictable.

Placing XML-related threats into the context of an overall security strategy is critical.

The other main reason that security is such a complex issue is the "twenty doors" problem. Simply securing one part of a system, or a network, or a building provides only false security. Chief Information Officers (CIOs) must consider all possible risks, including those unpredictable ones that are potentially the most costly. Therefore, placing XML-related threats into the context of an overall security strategy is critical. If a company increases its risk by not securing its Web Services, then there's no question that the company must close and lock this door. However, simply securing a company's XML traffic alone can at best provide only a false sense of security.

Every CIO must therefore consider multiple areas within the IT environment, including email, Web applications, remote users, and more, and evaluate the threats that each of these channels presents. When Web Services introduce new application interfaces, for example, the CIO must conduct basic due diligence to do everything they can to protect the business against any new attack vectors that may now exist.

It's also important to realize that existing security approaches are still necessary, but no longer sufficient to address these new attack vectors. User authentication and authorization are not enough, because companies do not authenticate many business transactions, and in any case, authentication mechanisms cannot protect the network, because of the possibility of compromising those mechanisms. Furthermore, Secure Sockets Layer (SSL) is also not sufficient to address XML-based threats, because SSL is a point-to-point technology that inhibits the loosely coupled interactions critical to the operation of Services within a Service-Oriented Architecture (SOA) implementation. SSL also cannot protect against attacks within the enterprise or against the network infrastructure itself.

Today, it's possible to find XML buried within most applications, middleware, and messages on the network all over the IT infrastructure. Even if every modern application were able to protect itself against every XML-related threat, companies would still have to deal with a gigantic legacy problem. There will never be a foolproof guarantee that there won't be vulnerabilities to tainted or malicious XML in some application somewhere on the network.

The security environment for XML and Web Services

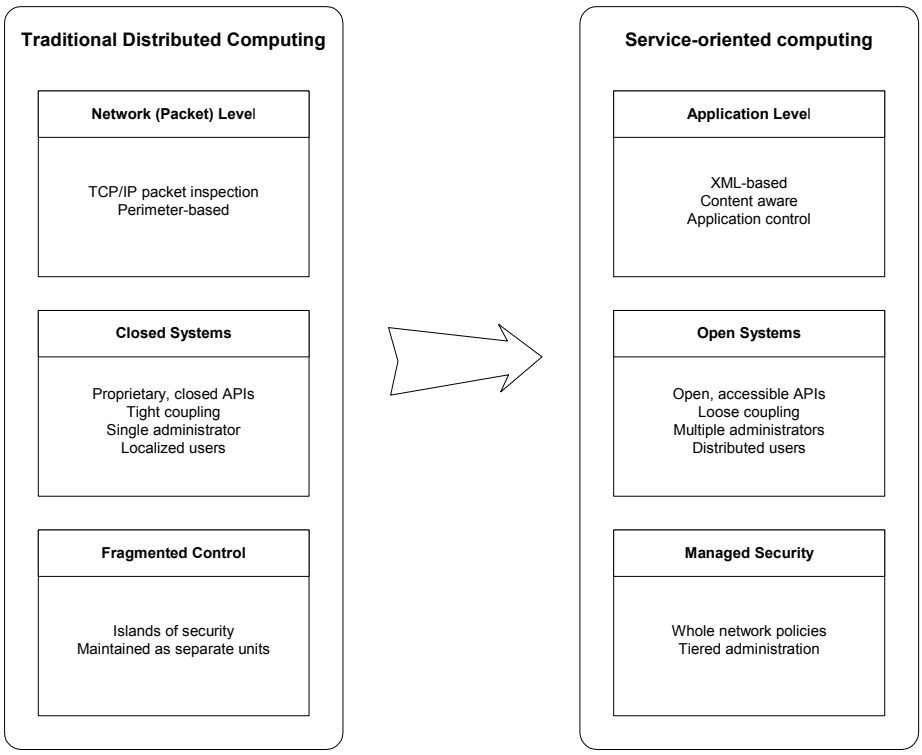
IT security solutions address how to mitigate risks in the enterprise at all levels. Locking 19 entrances to a building does not provide security if there are 20 entrances, and the same is true for IT. Therefore, discussions of security must cover all aspects of security, including human, physical, network, and application security. It's important to place XML threats, including those relating to Web

Services, within the overall context of IT security; after all, a hacker doesn't care if a particular vulnerability is Web Service-related.

Compounding the threats that XML introduces in the enterprise is the fact that the broader trend to SOA as an organizing principle for all of IT introduces an additional level of challenges for IT security. Security for traditional distributed computing depended on islands of security, where networks had their own perimeter security which considered users within the network as trusted. However, this "trusted vs. untrusted" dichotomy breaks down when a company has followed the SOA best practice of abstracting heterogeneous IT functionality as looselycoupled Services, because users can access Services located on various systems across one or more enterprises. The concept of trusted groups no longer has meaning; instead, enterprises must institute policies that apply to their entire enterprise network (including participants invited from outside), and administer that security in a tiered, or hierarchical fashion.

Securing such open, loosely coupled systems, therefore, requires a more sophisticated security approach than traditional environments. Different systems now have different policies and possibly different security mechanisms. This transition from distributed computing to Service-oriented computing introduces many different changes to the context for IT security, as Figure 1 below illustrates:

Figure 1: The Security Context for Service-Oriented Computing



Source: ZapThink

SOA changes the way companies must deal with security. In particular, perimeter-based security is no longer sufficient when companies must inspect traffic on a per-message basis; security based on closed system interfaces is not appropriate for looselycoupled, composite Services that span a wide range of systems and applications; and it doesn't make sense to isolate each system for

the purposes of security administration. Instead, companies must establish policies and contracts to apply to Services no matter where they are in the network.

XML: A new threat vector

While it is certainly possible to implement SOA without the use of XML, there's no question that XML is becoming increasingly widespread in today's distributed computing environments, even as companies begin the trek toward enterprise SOA. Within the broader context of IT security, therefore, addressing XML threats becomes a critical task—one of the 20 doors that companies must close, if you will.

XML messages are threat vectors for the enterprise, just as email messages and Web pages are.

In fact, XML messages are threat vectors for the enterprise, just as email messages and Web pages are. XML messages can carry familiar threats like worms and viruses, and in fact may provide additional approaches for hiding and injecting worms and viruses as compared to email. For example, not only can a Web Services request carry executable files, but such requests can actually call for the execution of those files.

Furthermore, some of XML's greatest strengths are actually its greatest weaknesses when it comes to security. XML is a text-based language, making it easier for hackers to understand. XML is also by nature extensible, in that every XML message contains its own metadata, essentially providing a roadmap for threats. And finally, XML exposes information about how a business operates by the very fact that it exchanges information in documents adhering to known schemas—known not only to the intended parties in the exchange, but often to the general public as well.

XML, therefore, greatly simplifies a type of threat known as the *insider attack*, which is an attack by hackers who maliciously use knowledge about the business processes they obtain though probing the processes' public interfaces. The Web Service public infrastructure and XML's inherent transparency also leads to attacks based on exploiting weaknesses in the application, attacks based on flooding the network with traffic, and attacks based on finding an entry point for inserting a malicious agent.

Software products, both commercial and open source, also provide a new route for various XML threats. People have reported buffer overflow attack vulnerabilities in products as diverse as IBM DB2 and Sun Solaris. Apache Xerces and Microsoft ASP.NET have suffered from denial of service attacks. People have also reported external entity attacks against products like PeopleSoft and Apache Axis. The list of products vulnerable to various forms of XML threats is actually quite long, and while vendors (and open source communities) continually work to mitigate such threats, there's no question that this category of problem will continue for the foreseeable future.

How standard threats map to the XML world

While XML is a relatively new threat vector in the enterprise, it's important to remember that older, familiar attacks can now leverage XML as well. For example, common Web vulnerabilities including buffer overflow attacks, denial of service attacks, and Trojan horses can now take advantage of the XML threat vector to open up new areas of vulnerability.

A *buffer overflow* is an anomalous condition in which a process attempts to put more data into a buffer than memory allows, resulting in the extra data overwriting adjacent memory locations. Such overflows can cause processes to crash or produce incorrect results. Intentionally malicious inputs can trigger buffer overflows, leading to a variety of unexpected results. As a result, buffer overflows are popular among hackers because they form the basis of many exploits. Buffer overflows can work regardless of the type of data in the buffer, so XML-based buffer overflows work the same as overflows of other kinds of data.

The next form of attack, the *denial of service* (DoS) attack, is more insidious. DoS is an attack on a computer system or network that causes a loss of service by consuming the bandwidth or other resources of the target network or the computational resources of the target systems. DoS attacks depend upon large quantities of data, and as with buffer overflows, the format of the data is typically irrelevant—XML works as well as any other form. Distributed denial of service (DDoS) attacks, which leverage multiple distributed systems to deliver the attack, can also depend upon any data format. *XML denial of service* (XDoS) attacks often also overwhelm the XML parser, the SOAP stack or other XML processing components, preventing the availability of the Web Service or other interface to legitimate users.

The third common form of attack, the *Trojan horse*, is a malicious program disguised as legitimate software. There are two common types of Trojan horses: otherwise useful software a hacker corrupts with malicious code that executes when the target program executes, or a standalone program that masquerades as something else in order to trick the user into running it. Because Trojan horses rely upon other software to execute, they can be the conduit for other forms of attack, like buffer overflows. It is also possible for a Trojan horse to infect XML processes in various applications from Internet browsers to databases.

Threats unique to XML

In addition to common threats like buffer overflows and Trojan horses, there are also threats that are unique to XML, including semantic threats, encoding attacks, and certain external threats. In addition, XML-based Web Services, Asynchronous JavaScript and XML (AJAX), and other newer XML standards and approaches are exposing mission-critical applications to new XML-based security threats. The fact that Web Services have become pervasive throughout IT for both enterprise and telecommunications applications has also increased the risk of XML-based attacks. It is critical, therefore, for existing security systems to keep up with the new attacks that can now occur due to the prevalence of XML on the network.

It is critical for existing security systems to keep up with the new attacks that can now occur due to the prevalence of XML on the network.

Semantic threats in particular are an important category of threat, because there are numerous openings to exploit the exposure of operational knowledge using a public semantic infrastructure. Semantic threats include code injection and SQL injection attacks, among others. A related semantic threat unique to XML is *XML injection*, in which a malicious party inserts XML into the document. This malicious XML may actually be valid XML, but contain content that will break the parser or, like SQL injection, elicit a response containing confidential information,

for example. Basically, any threat that manipulates the representation of the XML document to change the semantics of that document constitutes a semantic threat.

In fact, it's possible to institute a "man in the middle" attack by compromising the routing information contained in an XML message. This XML routing information helps direct XML traffic through the IT environment by allowing an XML intermediary to assign routing instructions to a document. If attackers compromise the intermediary, they will be able to insert bogus instructions to re-route a confidential file. The attackers can then strip out the malicious instructions before forwarding the document to its destination, with the recipient being none the wiser that there was a malicious intermediary that intercepted the message.

Semantic threats also include a range of data integrity and confidentiality attacks, including message or data tampering. Message tampering means the modification of parts of a request or response in-flight, while data tampering involves exploiting weaknesses in the access control mechanism, permitting the attacker to make unauthorized calls to the Web Service to alter data.

WSDL scanning or enumeration is another semantic threat. Included in every Web Services Description Language (WSDL) file is a mechanism for dynamically describing the parameters a Web Service exposes. Attackers might cycle through the various command combinations to discover unintentionally related or unpublished Web Service capabilities, enabling them to pass unauthorized or malicious requests to the Service.

Encoding attacks are threats that take advantage of naïve or broken XML parsers that are unable to handle encodings correctly. One type of encoding attack is a coercive parsing attack, which attempts to exploit legacy interfaces to XML components in a heterogeneous infrastructure. The attack tries to overwhelm a system's processing capabilities, leading to a DoS attack, or it may install malicious code in a Trojan horse attack.

Structural threats are threats related to the structure of the XML document. Included among structural threats are *oversized payloads*: since XML is verbose, normal XML messages can easily be quite large. It can be difficult to tell, therefore, whether a particularly large message or XML file is normal, or whether it is a sign that an attacker is attempting to manipulate the parser to execute a DoS attack. *Recursive payloads* are another structural threat. Because XML can nest elements within a document to address complex relationships, it's possible that attackers can attempt to break an XML parser by creating a file with thousands of nested elements, or with elements that are nested in an invalid way.

Grammar validation threats are threats related to schema validation; for example, *schema poisoning*. XML schemas model an XML document's grammar and template structure, and parsers rely upon schemas to interpret Web Service messages and other XML documents. Since schemas can provide processing instructions to parsers, they are vulnerable to tampering. An attacker may attempt to compromise the schema file and replace it with a similar but modified one at a different location.

In fact, the relationship between XML message parsing and validation leads to a range of vulnerabilities. The challenge here is that with naïve XML validation implementations, validation must typically wait until the parsing step completes (known as the parsing precondition), so an attack that targets the parser may never reach the validation step. Smarter approaches to validation, however, can begin schema validation while the parsing is still taking place. But even these

smarter approaches are vulnerable, because software will still parse any pathological node before it validates it. The solution is for the validation solution to perform special checks on the XML message before any parsing takes place.

Additional threats include *external entity threats* that manipulate the XML processor to access code at remote locations improperly. An example of an external entity threat is a *malicious include*, which causes a Web Service to include invalid external data in its output or return privileged files from the server.

II. Addressing Current and Future XML Threats

While it is clear that XML introduces new threats within the IT environment, it's important to point out that XML is part of the solution as well.

While it is clear that XML introduces new threats within the IT environment, it's important to point out that XML is part of the solution as well. As XML becomes the dominant messaging paradigm, and companies implement clear semantics and a requirement that messages be correct, the Internet overall can become a safer place to exchange messages, and in general, to do business. In some ways, managing XML threats can address persistent security issues that were actually open before the advent of XML. Because it's now necessary to deal with security at the content level, *XML Threat Management* (XTM) can ensure greater security overall than the traditional TCP/IP firewalls and virus scanning approaches in use today.

The basics of XML Threat Management

To harden a system that provides or consumes Web Services or otherwise exchanges XML messages, it's imperative for any XTM solution to perform several important threat prevention steps, as outlined in Table 1 below:

Table 1: The Basics of XML Threat Management

XML Check	Description	Performance Cost	Key Requirement
XDoS Attacks	Checks for wellformedness and performs structural sanity checks against parsing attacks	Very Low	Policy
Anomaly Detection	Detects messages that deviate from statistical norm	Low	Training
XPath	Checks for specific constructs like SOAP envelopes	Medium	XPath rule sets
Schema Validation	Determines if the message is a valid representation of an existing schema	High	Effective, comprehensive schemas
Content Signatures	Tests content values and ranges	Very High	Application-specific rule sets

Source: Tarari and ZapThink

There is a hierarchy of cost in processing and also a difference in the cost of the development and maintenance of the rules necessary to protect XML.

Basically, XTM consists of all the tasks a company should perform to guard against XML threats when the input is not under the company's control. Note from the table above that each type of XML check has a different performance cost, as well as a different key requirement. It's important, therefore, to keep in mind that there is a hierarchy of cost in processing and also a difference in the cost of the development and maintenance of the rules necessary to protect XML. Checks like XDoS attacks and anomaly detection are more automatic, XPath checks require the mostly manual construction of rule sets, schema validation requires schema creation and management, and content checks require

application-specific rule sets that may be quite detailed. As a result, the organizational costs of different XTM tasks can be every bit as significant as the performance costs.

XTM in the Context of XML Security

While it's important for any organization to place XTM in the context of an overall XML security plan and indeed in a comprehensive IT security strategy, certain aspects of XML security fall outside of the strict definition of XTM. XTM basically assumes a security layer that provides identity and access management as well as encryption, decryption, and digital signing. XTM therefore addresses the gaps in a broader XML security infrastructure to protect against the specific threats outlined above. Nevertheless, it's also important to monitor for other types of attacks as well, in conjunction with an overall effort to ensure compliance with service-level agreements that may or may not be security-specific.

The overall approach to dealing with the broad range of XML security issues is through the use of an *XML gateway*, which is typically a hardware device that is able to perform all the necessary security steps in a high-performance manner. Such XML gateways must ensure message integrity by encrypting sensitive data and verifying signatures on incoming requests. The standards *XML Encryption* and *XML-DigitalSignature* are the primary mechanisms of ensuring message integrity in the XML/Web services world. Such devices must also shield themselves from DoS attacks as well as protect underlying Services against such attacks. XML gateways must also offer content-based defenses that stop attacks including semantic threats like SQL injection and external entity attacks. Such gateways therefore should offer basic XML security as well as XTM capabilities in order to provide for comprehensive protection.

Applying the security tradeoff to XML threats

There's no such thing as a threat-free environment, or perfect threat prevention.

Providing sufficient XTM capabilities is both processor- and network-intensive, leading to hardware-based XML appliances that can both accelerate brute force processing steps like encryption and decryption, as well as enable efficient approaches to applying the XTM techniques described in the section above. Nevertheless, such devices still succumb to the enterprise security tradeoff, which points out that perfect security is impossible, or in other words, infinitely expensive. After all, there's no such thing as a threat-free environment, or perfect threat prevention. It's important, therefore, for companies to decide how they can approach XTM in a realistic way, given their budgetary and time constraints.

The issue of where and how to address XML threats is therefore a challenging task. Some parts of the XTM solution should run at network speeds and provide some measure of security at any point in the network. In the core of the network it's important both to run processes at network speed and also to have a high tolerance (no false positive indications of threats), while stopping most threats.

The perimeter of the network, however, requires a higher level of threat protection, including the sorting of traffic that requires higher levels of verification, depending on its context. It's especially important to protect traffic going both into and out of Web servers, in case the server has been compromised. It's also important to consider security within the perimeter to double-check for malformed traffic to guard against compromised applications and Trojan horses.

Reducing the risks of advanced XML applications

The transaction-intensive environment that Web Services, AJAX, and other XML-based standards and approaches support requires a secure, reliable network

with authorized, accurate data flowing through it. Because XML-centric applications are vulnerable to variants of many of the same exploits that are already familiar to the enterprise, protecting the network from such XML-borne exploits is clearly business-critical.

One important area of concern is at the browser interface. Because so many of today's applications rely upon HTTP and HTML, they are typically stateless and require round-trip page updates whenever the user requires new information. AJAX helps to solve this round-trip efficiency problem allowing the client to communicate with the server asynchronously, without interrupting the user. However, while AJAX enables the Web to be a standalone software development platform, it also brings performance and security considerations that both developers and companies implementing AJAX applications must deal with.

AJAX brings performance and security considerations that both developers and companies implementing AJAX applications must deal with.

Many of the XML threats discussed above can leverage AJAX. For example, an AJAX Trojan horse can mount a DoS attack on the enterprise. In other cases, when the user is taking advantage of a virtual private network (VPN), an AJAX-based attack can easily flood the VPN channel with XML data, immobilizing the mobile VPN user, or a malicious mobile user can provide the conduit for an attack against the enterprise's Web server or its applications. As a result, such AJAX-based attacks can threaten the individual remote user or the entire enterprise. It is often the case that XML schemas do not exist for the XML data that AJAX generates, so schema validation is usually useless in preventing AJAX applications that run amok. With AJAX, as with many other types of dynamic XML applications, effective threat mitigation other than schema validation is critical.

The fact that XML messages can consume many times the bandwidth of traditional binary data formats means that AJAX-based applications often lead to system-wide performance degradation. In fact, the processing overhead of AJAX data can still compromise the schema validation defense. An XML gateway that supports both baseline XML threat detection and schema validation should have the XML acceleration horsepower to deal with the volume of traffic that AJAX applications will generate.

III. Tarari's Approach

San Diego-based Tarari, Inc. designs and produces *Tarari Content Processors* that accelerate and offload compute-intensive, complex algorithms. The Content Processors can accelerate a range of network security and digital media applications, as well as XML and Web Services operations. Tarari's product line primarily includes application-specific integrated circuits (ASICs), and Tarari also offers boards and software acceleration components for network equipment, appliance, and server vendors as well as for independent software vendors and enterprises.

Tarari Content Processors offer XML Threat Management (XTM) capabilities, including line-speed, zero-hour and signature-based detection of attacks carried in XML-based protocols, packets, and messages. Tarari designed its chips to allow original equipment manufacturers (OEMs) to add XTM capabilities into network security equipment such as firewalls, VPNs, intrusion prevention systems (IPSs), unified threat management (UTM) solutions, access control tools, application networking devices and application accelerators. The Tarari Content Processors simply snap in to networking, appliance, blades and server systems.

Tarari's XTM Content Processor guarantees message security for top XML threats.

Tarari's *XTM Content Processor* (XTM-CP) guarantees message security for top XML threats, snaps into network security equipment, and is 100% hardware, which means it is invulnerable to known attacks directed at software infrastructure. It supplements existing defenses with a full-spectrum approach

for networks carrying XML. Tarari also offers patent-pending algorithms that are able to detect XML-borne attacks with greater protection and higher throughput than XML firewalls alone can.

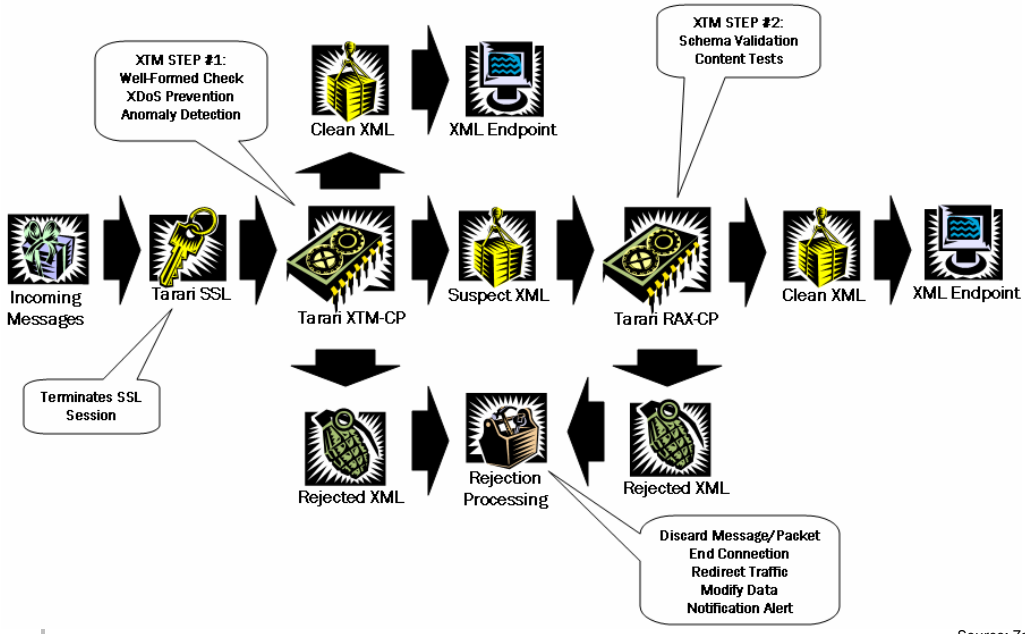
In addition, Tarari offers the XML *RAX Content Processor* (RAX-CP), which can complement XTM, adding layered capabilities for any complex XML processing task, including schema validation and content-signature verification as well as content-based routing, parsing, XPath evaluation, SOAP termination, XML Security, and XSLT transformation. RAX-CP uses Tarari's *Random Access XML* (RAX) processing methodology to accelerate and simplify XML processing by enabling direct access to any data within an XML message. The access requires neither conventional parsing nor creating and traversing an XML tree structure, RAX-CP also leverages Tarari's ASIC to support transparent hardware acceleration.

Protection for thousands of applications

The secret to Tarari's XTM approach is the way they provide two layers of protection.

The secret to Tarari's XTM approach is the way in which they provide two layers of protection, as shown in Figure 2 below. First, the Tarari XTM-CP checks incoming messages for wellformedness, prevents XDoS attacks, and detects anomalies, all at wire speed. XTM-CP eliminates malicious or unwanted content before it reaches the applications that will process the XML. XTM-CP can also stop some attacks on the first packet with active anomaly detection and full streaming operation. Tarari XTM-CP also protects against AJAX-borne threats by vetting XML traffic in all AJAX HTTP requests and responses, stopping out-of-control AJAX applications.

Figure 2: XTM-Enabled Network Security with Tarari



Source: ZapThink

In some cases, these tests are sufficient, but in other cases, the Tarari XTM-CP passes off the message to the RAX-CP, which performs schema validation and several other content-based checks. Tarari's two-tier approach also helps to address the relative performance and organizational costs of the various XML checks described in Table 1 above, as well as their relative effectiveness. For

example, schema validation may not be practical for every document in the network because of performance limitations, and in any case, schema validation alone only applies to the subset of XML documents that have schemas. Those malicious documents that schema validation can detect may be only a small portion of the threats to an enterprise.

It's a common misconception that the combination of authentication and schema validation is sufficient for addressing XML threats. In fact, since most schemas allow for an infinite number of different document instances, it's possible for a hacker to create XML documents that will readily pass the schema validation test but still wreck havoc on various systems. For example, since most schemas allow for unlimited repeatable elements in some content models, it is a simple matter to create a document that will be schema-valid and still exhaust available processing resources. This problem is why Tarari always recommends performing baseline checks of XTM first, even when schema validation is also necessary. XDoS checks, well-formed checking, and anomaly detection guarantee that the message is safe to process, whether for schema validation or any other XML process.

Conventional models for identifying and processing XML in the network are insufficient to address the diversity of XML's uses.

Furthermore, conventional models for identifying and processing XML in the network are insufficient to address the diversity of XML's uses. Not only do many schemas simply not exist for many applications of XML, but applications embed the XML itself into data streams in a variety of ways, including base64 binary encodings, URI-encodings in AJAX applications, as islands in graphic, video, HTML and other document formats, as conversational fragments in the Jabber community's Extensible Messaging and Presence Protocol (XMPP), and so on.

XML appears in a large number of enterprise applications as well. Programs with reported XML vulnerabilities include Microsoft SQL Server, Microsoft Internet Explorer, Microsoft ASP.Net, Oracle 9i, Adobe Reader, Apache Axis, PeopleSoft, IBM DB2, MySQL, and Sun J2EE, all of which transport and embed XML in many different ways. To address this issue, Tarari XTM scans data streams for encoded and embedded XML and performs basic checks which ensure that the XML cannot break receiving applications, something no firewall following the cooperative processing model of authenticate/validate can do. Tarari XTM offers protection for thousands of applications, no matter how they use XML.

Protection of the network and servers from XML zero-hour attacks

Schema validation can only eliminate threats from documents which are not schema-valid. Well-formed checking can only eliminate threats from documents which are not well-formed. XDoS checks can only eliminate threats from documents which have known XDoS attack signatures. XML's inherent extensibility offers the hacker many different ways to mount attacks, including novel attacks that such defenses cannot identify. Tarari XTM-CP stops malicious XML traffic that XML firewalls can't block, by means of an anomaly detection algorithm that lets only good messages through. The algorithm blocks attacks that use malicious XML, even though the XML is technically well-formed, valid, and free of XDoS signatures. In fact, anomaly detection is the only known method for detecting this *zero-hour attack*, which is an attack based on a previously unknown vulnerability and is hence typically immune to defenses against attacks with known attack signatures.

Furthermore, Tarari XTM is similar to many advanced intrusion detection systems (IDS) and intrusion prevention systems (IPS) that also offer two types of protections. Advanced IDS/IPS offer signature scanning to detect viruses as well as anomaly detection. Anomaly detection alone may detect zero-hour attacks, but may also produce false positives, while signature scanning misses new

attacks but rarely produces false positives. When the two approaches work together, it's possible to catch both known and zero-hour attacks with a very low rate of false positives. The combination of both approaches therefore provides the best protection overall. Tarari has adapted this two-pronged approach to XML threat management, combining well-formed and XDoS checking, parallel to IDS/IPS threat signature scanning and anomaly detection in a single process.

Differentiators of the Tarari offering

The Tarari XTM silicon engine has a maximum throughput of up to 4 Gbps, and with multiple cards can handle a 10 G line rate. The solution provides the same active protection against XML-based attacks that IPS, firewalls, and unified threat management (UTM) devices offer against other types of network threats. Tarari XTM handles large numbers of connections in concurrent, interleaved streams, managing XML fragments. It interfaces directly to network processors handling transport-level packets. This network-oriented solution provides protection beyond what even XML firewalls can deliver.

Tarari XTM recognizes dozens of familiar XDoS attacks, including recursive and oversized payloads and well-formed checking. In addition, Tarari's patent-pending XML anomaly detection learns to recognize threat-bearing messages without known threat signatures. Furthermore, Tarari's implementation is in silicon, providing immunity against numerous exploits of software infrastructure, as opposed to other devices that offer their core intellectual property in software and are therefore vulnerable to the very threats they protect against. The final differentiator is the straightforward administration and plug-in capability that Tarari XTM provides.

IV. The ZapThink Take

At the core of any Service-Oriented Architecture implementation are the Services that abstract application functionality and data in the enterprise, and at the core of the definition of a Service is software that communicates via the exchange of messages. The vast majority of such messages within any of today's SOA implementations are XML-based, and an increasing amount of other network traffic leverages XML, as well. There's no question that XML is becoming as ubiquitous as the underlying protocols TCP/IP and Ethernet.

It's clear, therefore, that any IT security effort must plan for and deal with XML threats, partly because of the explosion in the use of XML, but also because of XML's core properties. After all, XML is text-based, metadata-rich, and both human readable and often executable. XML's power and flexibility are also its greatest vulnerabilities, leading to multiple threat categories.

Any IT shop that doesn't address the full range of XML threats is inviting trouble.

Any IT shop that doesn't address the full range of XML threats is inviting trouble. Hackers are both persistent and intelligent, and will probe for all types of vulnerabilities. If companies don't implement limits on oversized and improperly structured XML messages, then structural threats are likely to succeed. On the other hand, if they don't implement schema validation and content filtering, then semantic threats may be successful. And most importantly, if their XML Threat Management slows down the network, then the business will still suffer the effects of XML threats, regardless of whether those threats are present. Companies require XTM that deals with all forms of XML threats at wire speed.

Copyright, Trademark Notice, and Statement of Opinion

About ZapThink, LLC

ZapThink is an IT advisory and analysis firm that provides trusted advice and critical insight into the architectural and organizational changes brought about by the movement to XML, Web Services, and Service Orientation. We provide our three target audiences of IT vendors, service providers and end-users a clear roadmap for standards-based, loosely coupled distributed computing – a vision of IT meeting the needs of the agile business.

ZapThink helps its customers in three ways: by helping companies understand IT products and services in the context of Service-Oriented Architecture (SOA) and the vision of Service Orientation, by providing guidance into emerging best practices for Web Services and SOA adoption, and by bringing together all our audiences into a network that provides business value and expertise to each member of the network.

ZapThink provides market intelligence to IT vendors and professional services firms that offer XML and Web Services-based products and services in order to help them understand their competitive landscape, plan their product roadmaps, and communicate their value proposition to their customers within the context of Service Orientation.

ZapThink provides guidance and expertise to professional services firms to help them grow and innovate their services as well as promote their capabilities to end-users and vendors looking to grow their businesses.

ZapThink also provides implementation intelligence to IT users who are seeking guidance and clarity into the best practices for planning and implementing SOA, including how to assemble the available products and services into a coherent plan.

ZapThink's senior analysts are widely regarded as the "go to analysts" for XML, Web Services, and SOA by vendors, end-users, and the press. Respected for their candid, insightful opinions, they are in great demand as speakers, and have presented at conferences and industry events around the world. They are among the most quoted industry analysts in the IT industry. ZapThink was founded in November 2000 and is headquartered in Baltimore, Maryland.

ZAPTHINK CONTACT:
ZapThink, LLC
108 Woodlawn Road
Baltimore, MD 21210
Phone: +1 (781) 207 0203
Fax: +1 (786) 524 3186
info@zapthink.com

ZapThink, LLC • 108 Woodlawn Road • Baltimore, MD 21210 • info@zapthink.com • www.zapthink.com