Alex Stamos
Scott Stender

BLACK HAT BRIEFINGS

Attacking Web Services: The Next Generation of Vulnerable Enterprise Apps

Web Services represent a new and unexplored set of security-sensitive technologies that have been widely deployed by large companies, governments, financial institutions, and in consumer applications. Unfortunately, the attributes that make web services attractive, such as their ease of use, platform independence, use of HTTP and powerful functionality, also make them a great target for attack.

In this talk, we will explain the basic technologies (such as XML, SOAP, and UDDI) upon which web services are built, and explore the innate security weaknesses in each. We will then demonstrate new attacks that exist in web service infrastructures, and show how classic web application attacks (SQL Injection, XSS, etc…) can be retooled to work with the next-generation of enterprise applications.

The speakers will also demonstrate some of the first publicly available tools for finding and penetrating web service enabled systems.

Alex Stamos *is a founding partner of iSEC Partners, LLC, a strategic digital security organization, with several years experience in security and information technology. Alex is an experienced security engineer and consultant specializing in application security and securing large infrastructures, and has taught many classes in network and application security.*

Before he helped form iSEC Partners, Alex spent two years as a Managing Security Architect with @stake. Alex performed as a technical leader on many complex and difficult assignments, including a thorough penetration test and architectural review of a 6 million line enterprise management system, a secure re-design of a multi-thousand host ASP network, and a thorough analysis and code review of a major commercial web server. He was also one of @stake's West Coast trainers, educating select technical audiences in advanced network and application attacks.

Before @stake, Alex had operational security responsibility over 50 Fortune-500 web applications. He has also worked at a DoE National Laboratory. He holds a BSEE from the University of California, Berkeley, where he participated in research projects related to distributed secure storage and automatic C code auditing.

Scott Stender *is a founding partner of iSEC Partners, LLC, a strategic digital security organization. Scott brings with him several years of experience in large-scale software development and security consulting. Prior to iSEC, Scott worked as an application security analyst with @stake where he led and delivered on many of @stake's highest priority clients.*

Before @stake, Scott worked for Microsoft Corporation where he was responsible for security and reliability analysis for one of Microsoft's distributed enterprise applications. In this role, Scott drew on his technical expertise in platform internals, server infrastructure, and application security, combined with his understanding of effective software development processes to concurrently improve the reliability, performance, and security of a product running on millions of computers worldwide.

In his research, Scott focuses on secure software engineering methodology and security analysis of core technologies. Most recently, Scott was published in the January-February 2005 issue of "IEEE Security & Privacy", where he co-authored a paper entitled "Software Penetration Testing". He holds a BS in Computer Engineering from the University of Notre Dame.

Attacking Web Services

The Next Generation of Vulnerable Enterprise Applications

Alex Stamos
alex@isecpartners.com

Scott Stender
scott@isecpartners.com

BlackHat 2005

Information Security Partners, LLC
iSECPartners.com

Introduction

- **Who are we?**
  - Founding Partners of Information Security Partners, LLC (iSEC Partners)
  - Application security consultants / researchers
- **Why listen to this talk?**
  - As you'll see, Web Services are being deployed all around us
  - All of this work is based upon our experiences with real enterprise web service applications
  - There are a lot of interesting research opportunities
    - Find out what we don't know
- **The latest version of these slides and tools are available here:**
  - https://www.isecpartners.com/blackhat.html

Introduction: What are Web Services?

- **It's an overloaded term (and a great way to raise VC)**
- **For our purposes, web services are communication protocols that:**
  - Use XML as the base meta-language to define communication
  - Provide computer-computer communication
  - Use standard protocols, often controlled by W3C, OASIS, and WS-I
  - Designed to be platform and transport-independent

Introduction: What are Web Services?

- **Why are they so compelling?**
  - Web service standards are built upon well understood technologies
  - Adoption by large software vendors has been extremely quick
  - Web services are sometimes described as a panacea to solve interoperability issues
  - Lots of "magic pixie dust"
  - Are very easy to write:

```
using System.ComponentModel;
using System.Web.Services;
namespace WSTest{
  public class Test : System.Web.Services.WebService
  {
    [WebMethod]
    public string HelloWorld()
    { return "Hello World"; }
  }
}
```

Introduction: What are Web Services?

- **Value to corporate management is easy to understand**
  - Fake quote: "Lets expose our Mainframe APIs through SOAP and use plentiful Java developers on Windows/Linux instead of rare CICS developers on expensive mainframes to extend our system's functionality. If we change our mind about Java, no problem; C#, Perl, Python, C++, and every other language is already compatible with SOAP."

What is this talk?

- **Introduce security risks associated with Web Services**
- **Many of the protocols and issues are familiar**
  - Classic application issues (injection attacks, session management) are still relevant in the WS world
  - Plenty of new protocols and attack surfaces to research
    - **Prediction**: *The next couple of years will see an avalanche of vulnerabilities related to web service issues*
- **This talk is not about WS-Security standards**
  - Standards for crypto, authorization, authentication, etc… are necessary and important
  - Like TLS, standards like this are good building blocks, but do not eliminate vulnerabilities in an application
  - Ex: SSL doesn't protect against SQL injection

Where are Web Services being used?

- **Between Companies (B2B)**
  - Web services are being deployed to replace or supplement older data exchange protocols, such as EDI
  - 3rd party standards limit "Not Invented Here" syndrome
  - Example: Credit Card Clearer -> Bank -> Credit Bureau -> Lender
  - Lots of opportunity for savings here
- **Internal to Companies**
  - All major corporate software vendors have or will offer web service interfaces to their applications
    - IBM, Microsoft, SAP, Oracle
  - Web service standards make connecting systems easy
    - This is great for IT management and productivity
    - This should be scary to security people

iSEC PARTNERS

Where are Web Services being used?

- **In front of legacy systems**
  - Finding people to develop on these systems is hard
  - Reliance on old software and systems restricts growth and improvement of corporate IT systems
  - **Solution**: Web service gateway in front of legacy system
  - IBM is a big mover in this middleware
  - Security in these situations is extremely tricky
- **Between tiers of Web Applications**
  - Front end is HTML/XHTML
  - Backend of SQL is replaced by SOAP
  - WS enabled databases consume these streams
  - Makes "XML Injection" very interesting

iSEC PARTNERS

Where are Web Services being used?

- **On consumer facing web pages**
  - AJAX: Asynchronous JavaScript and XML
    - maps.google.com
  - As APIs to add functionality
    - EBay
    - Google Search
    - Amazon
    - Bank of America

Code Breaks Free…

- **At one point, nobody worried about providing rich functionality to the public Internet**
- **People decided this was a bad idea and put up firewalls**
  - Only HTTP, HTTPS, SMTP allowed from the outside…
- **Web Services tunnel that functionality through ports often deemed "safe"**
- **Rich functionality once again hits the public Internet**
- **Let's propose a new slogan:**

Web Services

We poke holes in your firewall so you don't have to!

Attacks on Web Services

- **Web Services have been designed to be everything-agnostic**
  - Variety of technologies may be encountered at any layer
- **This talk focuses on those commonly encountered**
- **We will discuss security issues at three layers:**
  - Application
  - SOAP
  - XML

iSEC PARTNERS

Application Attacks

- **Every (most) applications accomplish something useful**
  - There is always something to attack
- **Application-specific flaws don't magically go away**
  - Design Flaws
  - Business Logic Errors
  - "Bad Idea" Methods (see UDDI discovery)
- **The same issues (OWASP Top 10) that have plagued us for years still exist**

iSEC PARTNERS

Application Attacks

- **SQL Injection**
  - Most web service applications are still backed by databases
  - SOAP/XML provide means to escape/obfuscate malicious characters
- **Overflows in unmanaged code**
- **Mistakes is authorization/authentication**
- **XSS**
  - Rich data representation allows charset games with browsers
  - Technologies such as AJAX allow new possibilities in XSS attacks
    - Creating a well formed SOAP request can be difficult
  - Attacks against other interfaces (such as internal customer support) more likely

iSEC PARTNERS

Our Friend: CDATA Field

- **XML has a specific technique to include non-legal characters in data, the CDATA field**
  - Developers assume that certain data types cannot be embedded in XML, and these assumptions can lead to vulnerabilities
  - When querying a standard commercial XML parser, the CDATA component will be stripped
    - The resulting string contains the non-escaped dangerous characters
    - Existence of CDATA tags is hidden from developer
  - Where is your input filtering?
- **Where to use this?**
  - SQL Injection
  - XML Injection
  - XSS (Against a separate web interface)
- **Example:**

```
<TAG1>
<![CDATA[<]]>SCRIPT<![CDATA[>]]>
    alert('XSS');
<![CDATA[<]]>/SCRIPT<![CDATA[>]]>
</TAG1>
```

iSEC PARTNERS

SOAP Attacks

- **SOAP is a standard which defines how to use XML to exchange data between programs**
  - Designed to capture RPC-style communication
  - Generally over HTTP/S, but this isn't required
    - MQ, SMTP, Carrier Pigeon
- **The "magic" of Web Services begins**
  - Programming infrastructure turns 9-line code sample into full-fledged web service
  - Ease of deployment sometimes masks deeper security issues
    - Serialization
    - Schema Validation
  - Attacks against layers of the stack are often left open

iSEC PARTNERS

SOAP Attacks

- **SOAP Interfaces are described using Web Services Description Language (WSDL)**
  - WSDLs can be quite complicated
  - Generally not created or consumed by human being
    - Auto-generated by WS framework
    - No access controls generally enforced on WSDLs
  - Requesting a WSDL can be as simple as adding a ?WSDL argument to the end of the URL
    - http://labs.isecpartners.com/blackhat.html?WSDL

iSEC PARTNERS

Example WSDL: EBay Price Watching

```
<?xml version="1.0"?>
<definitions name="eBayWatcherService"
   targetNamespace=
      "http://www.xmethods.net/sd/eBayWatcherService.wsdl"

     xmlns:tns="http://www.xmethods.net/sd/eBayWatcherServi
     ce.wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <message name="getCurrentPriceRequest">
      <part name="auction_id" type = "xsd:string"/>
   </message>
   <message name="getCurrentPriceResponse">
      <part name="return" type = "xsd:float"/>
   </message>

   <portType name="eBayWatcherPortType">
      <operation name="getCurrentPrice">
         <input
            message="tns:getCurrentPriceRequest"
            name="getCurrentPrice"/>
         <output
            message="tns:getCurrentPriceResponse"
            name="getCurrentPriceResponse"/>
      </operation>
   </portType>
```

```
<binding name="eBayWatcherBinding"
     type="tns:eBayWatcherPortType">
       <soap:binding
          style="rpc"

     transport="http://schemas.xmlsoap.org/soap/http"/
     >
       <operation name="getCurrentPrice">
          <soap:operation soapAction=""/>
          <input name="getCurrentPrice">
             <soap:body
                use="encoded"
                namespace="urn:xmethods-EbayWatcher"

     encodingStyle="http://schemas.xmlsoap.org/soap/en
     coding/"/>
          </input>
          <output name="getCurrentPriceResponse">
             <soap:body
                use="encoded"
                namespace="urn:xmethods-EbayWatcher"

     encodingStyle="http://schemas.xmlsoap.org/soap/en
     coding/"/>
          </output>
       </operation>
    </binding>

...
```


SOAP Exposure

- **Attack:** *WSDLs give away all of the sensitive information needed to attack a web application*
  - This includes "hidden" or debug methods that developers might not want exposed
  - These method have always existed
    - Real danger with applications "ported" to web services from normal web interface
- **Companies have always had "cruft" systems that are protected by obscurity**
  - **You** know about that 1:00AM FTP batch job your company does unencrypted over the Internet. Do you want everybody in this room to know about it?
  - Extranets, customer portals, one-off links to other businesses
  - These secret attack surfaces will be exposed through standardization on web service infrastructures
- **Defense: Manually review WSDLs to look for dangerous functions**
  - We've heard of people manually editing them out. Automagic processes might restore those
  - Debug functionality MUST be removed in a repeatable manner before deployment to production
    - "Secure development lifecycle" is not just marketing BS

SOAP Attacks

- **SOAP Headers**
  - Provide instructions on how a message should be handled
    - Often not necessary in basic applications
    - Still parsed/obeyed by WS frameworks
    - So many standards, so many attack surfaces
  - **Attack**: DoS in Processing Instructions
  - **Attack:** Source routing used to bypass security checks

- **SOAPAction Header**
  - Sometimes needed, sometimes filtered to attempt to remove soap requests. Often not required at all.
  - **Attack**: Bypass protections that rely on SOAPAction

SOAP Attacks

- **Session management**
  - SOAP, like HTTP, is stateless!
  - Developers need to program their own state mechanism. Options include:
    - In-line SessionID, defined
    - Cookie in header
  - SOAP is transport independent, so a message should be able to be passed without session information from the transport, such as a HTTP cookie
    - Often used, but it's a hack
    - **Attack**: Cookies might be stripped at the web server, or not properly routed to the part of the app where decisions are being made. Watch out!
  - New WS-I cryptographic standards might allow developers to bootstrap state
  - Classic state attacks work
    - Predictable IDs are still predictable
    - But, XSS can't easily access in-band stateID
  - **Attack:** SOAP, being stateless, might make applications vulnerable to replay attacks
    - Need to make sure cryptographic protections also include anti-replay

XML Introduction

- **What is XML?**
  - A standard for representing diverse sets of data
- **Representing data is hard work!**
  - Binary Data
  - Internationalization
  - Representing metacharacters in data
  - Defining and Validating schemas
  - Parsing mechanisms
- **Attacks**
  - Source-specified code page masks malicious characters
  - Complex/large DTD takes down parser
  - Injection attacks

XML Introduction

- **Based on a few basic but strict rules:**
  - Declarations
  - Tags must open and close
  - Tags must be properly nested
  - Case sensitive
  - Must have a root node
- **Why do we care about the rules?**
  - Attacking web services generally means creating valid XML
  - If your XML doesn't parse right, it gets dropped early on
  - Fuzzing XML structure might be fun, but you're only hitting the parser
- **Simple example:**

```
<product itemID="1234">
   <manufacturer>Toyota</manufacturer>
   <name>Corolla</name>
   <year>2001</year>
   <color>blue</color>
   <description>Excellent condition, 100K miles</description>
</product>
```

XML Introduction - Parsing

- **XML Documents are defined by:**
  - DTD: Old Standard
  - XSD: Current Method
  - **Attack**: Reference external DTD, allows tracking of document and parsing attacks
- **There are two common XML parsers used across platforms**
  - **SAX**: State-oriented, step-by-step stream parsing
    - Lighter weight, but not as intelligent
    - **Attack**: User controlled data overwrites earlier node.
  - **DOM**: Complicated, powerful parsing
    - **Attack**: DoS by sending extremely complicated, but legal, XML.
      - Creates huge object in memory
    - Why use other types of floods to attack? XML parsing gives a much larger multiplier
- **XPath engines provide query interface to XML documents**
  - Like other interpreted query languages, XPath injections are possible.
- **Always a bad idea: *custom* parsers**
  - "I can use a RegEx for that"
  - It is common to simulate SAX parsers as they are simple conceptually.
  - Plenty of devils in the details: XML tags inside CDATA block, Entity substitution

iSEC PARTNERS

XML Attacks

- **Emerging attack class: XML Injection**
  - Occurs when user input passed to XML stream
  - XML parsed by second-tier app, Mainframe, or DB
  - XML can be injected through application, stored in DB
    - When retrieved from DB, XML is now part of the stream

```
<UserRecord>
  <UniqueID>12345</UniqueID>
  <Name>Henry Ackerman</Name>
<Email>hackerman@bad.com</Email><UniqueID>0</UniqueID><Email>hackerman@bad.co
    m</Email>
  <Address>123 Disk Drive</Address>
  <ZipCode>98103</ZipCode>
  <PhoneNumber>206-123-4567</PhoneNumber>
</UserRecord>
```

SAX Parser Result*: UniqueID=0*

iSEC PARTNERS

Web Services DoS

- **Like all DoS, looking for multiplier advantage**
  - **CPU Time**
    - Extremely deep structures require CPU time to parse and search
    - References to external documents
      - Cause network timeout during parsing, may block process
    - Creating a correct DOM for complex XML is not trivial
  - **Memory Space**
    - Deep and broad structures
    - Large amounts of data in frequently used fields will be copied several times before being deleted
    - Memory exhaustion can be difficult against production systems, but creating garbage collection / VM overhead might slow the system
  - **Database Connections**
    - Despite low CPU/mem load, filling the DB request queue can wait state an application to death
    - Need to find a good SOAP request that does not require auth, but results in a heavy DB query
      - Perfect example: Initial User Authentication
    - Common database can be a single point of failure for multiple application servers

Web Services DoS

- **In any WS DoS case, there are important details to make the attack effective**
  - Legality of SOAP request
    - Matches DTD/XSD Syntax. This might not preclude embedding complex structures!
    - Matches real SOAP Method
      - Anything that “burrows” deeper into the application stack causes more load
      - Especially important when attacking databases
    - Might need a valid session ID
      - Authenticate once with a real SOAP stack, then use the SessionID/cookie into the static attack
  - Speed
    - Making a request is relatively heavy compared to other DoS
      - Requires a real TCP connection
      - Don’t use a SOAP framework. Most of the multiplier is lost
      - Need to listen for response for some attacks
    - We often run into limitations of the underlying script framework
      - Native framework would increase effectiveness of DoS

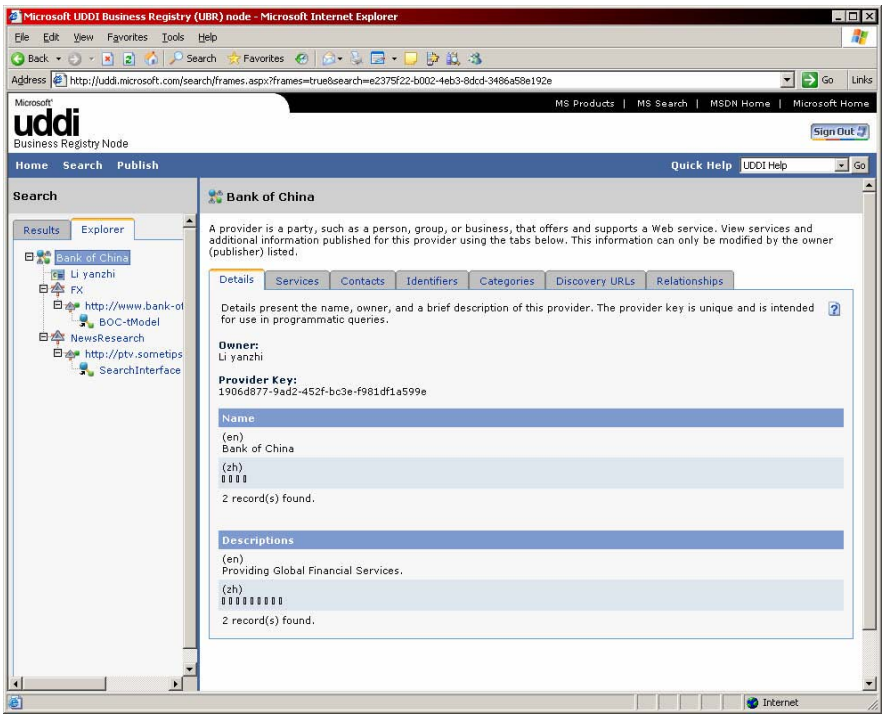
Web Service DoS: The Aftermath

- **We are currently researching some more possibilities**
  - Attacks against XPath equivalent to recent RegEx DoS
  - Using HTTP 1.1 pipelining to speed attack
  - SOAP equivalents of “teardrop” attacks against state: multiple fragmented requests
- **Defense isn’t easy**
  - Application server vendors need to add DoS into negative QA testing
    - There doesn’t seem to be much customer demand yet
  - Need to check complexity before parsing
    - Secure SOAP handler
    - XML “Firewall”
  - Use strict XML Schema verification
  - Don’t forget the “nooks and crannies” attackers can shove code into
    - SOAP Headers

Web Service Discovery Methods

- **UDDI**
  - Registries that list web services across multiple servers
  - Auto-magically works on some systems, such as .Net
  - Multiple authorities have created classification schemes
    - Winner is not yet clear
  - Not necessary to expose to world
    - B2B services that were always insecure were at least secret
    - Now advertised to entire world
    - UDDI servers support authentication and access control, but this is not the default (or common) configuration
  - **Attack**: UDDI points an attacker to all the information they need to attack a web service
- **UDDI Business Registry (UBR)**
  - Four major servers, run by IBM, Microsoft, SAP, and NTT
  - Has beautiful, searchable interface to find targets
    - Obviously, also searchable by web services
  - **Attack**: No binding authentication of registry
    - New WS-Security standards are building a PKI to authenticate UBR->Provider->Service
    - Confusion might be an attackers friend
  - Who needs nmap? UBR points you right to the source!

UBR Example



Web Service Discovery

- **Other 3rd Party Registries**
  - http://www.xmethods.com/ has an excellent list of fun services
- **DISCO / WS-Inspection**
  - Lightweight versions of UDDI
  - Provides information about a single server's web services
- **We have created a discovery tool: WSMap**
  - Scans a defined set of IPs for known app server URLs
  - "Tickles" WS endpoint with SOAP requests to generate telltale error
  - Looks for WSDLs
  - (Almost) identifies the application server

Attack Tree: Tying it all Together

- Navigate to UBR, ask for a site
- Attach to UDDI, ask for WSDL
- Examine WSDL, find dangerous methods
- Use fuzzer to test methods, find XML Injection
- Profit!

OWASP Top 10 – Still Relevant?

1. **Unvalidated Input**
2. **Broken Access Control**
3. **Broken Authentication and Session Management**
4. **Cross Site Scripting (XSS) Flaws**
5. **Buffer Overflows**
6. **Injection Flaws**
7. **Improper Error Handling**
8. **Insecure Storage**
9. **Denial of Service**
10. **Insecure Configuration Management**

The answer to all of these is YES.

Conclusion

- **Web Services are powerful, easy-to-use, and open.**
  - AKA: they are extraordinarily dangerous
  - Many crusty corporate secrets will now be exposed
- **Lots of security work still required**
  - Analysis of rapidly developing Web Services standards
    - WS-Security
    - WS-Routing
    - WS-"Everything"
  - Attack Tools
    - Better proxies
    - More efficient DoS
    - Better automated discovery
  - Define best practices for development
    - "XML Firewall" vendors want this to be a hardware solution
    - Like all good security, it really needs to be baked into the product by the engineers closest to the work
  - PKI Infrastructure for Authentication
    - Who will control the cryptographic infrastructure?

Web Services Security

Q&A

Alex Stamos
alex@isecpartners.com
Scott Stender
scott@isecpartners.com