



Dynamic Vulnerability Remediation with ModSecurity:

An Incident Response Approach

Ryan C. Barnett
Director of Application Security
ModSecurity Community Manager
Breach Security
Ryan.Barnett@breach.com

OWASP

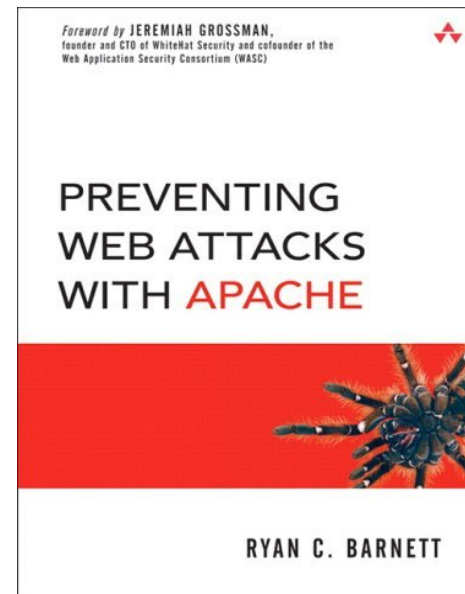
April 16th 2008

Copyright © The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document
under the terms of the OWASP License.

Introduction - Ryan Barnett

Background

- Director of Application Security at Breach.
- ModSecurity Community Manager.
- Background as an IDS/Web Security Admin.
- Author of *Preventing Web Attacks with Apache* (Addison/Wesley, 2006).



Introduction - Ryan Barnett

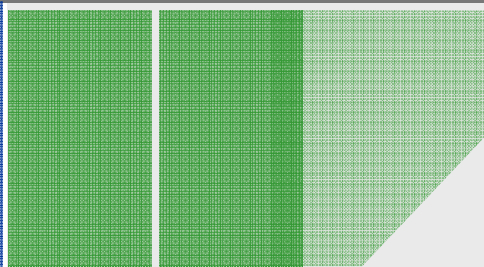
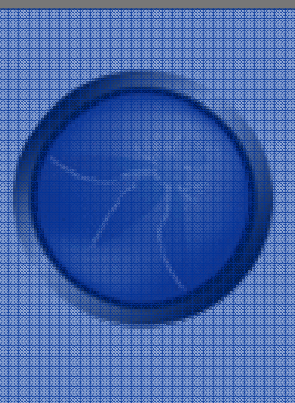
Open Source and Community Projects

- Board Member, Web Application Security Consortium.
- Project Leader, WASC Distributed Open Proxy Honeypots.
- Speaker/Instructor, Open Web Application Security Project
- Courseware Developer/Instructor for the SANS Institute.
- Project Leader, Center for Internet Security's Apache Benchmark.



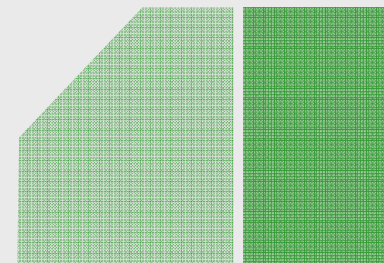
Agenda

- Dynamic Vulnerability Remediation Introduction
- Incident Response Process Approach
 - ▶ Preparation
 - ▶ Identification
 - ▶ Analysis
 - ▶ Virtual Patch Creation
 - ▶ Implementation/Testing
 - ▶ Recovery and Follow-Up
- Examples
 - ▶ Public Vulnerability Announcement
 - ▶ Source Code Review
 - ▶ Vulnerability Assessment
 - ▶ Real Incident: SQL Injection
- Complex Vulnerabilities
- Conclusion/Questions

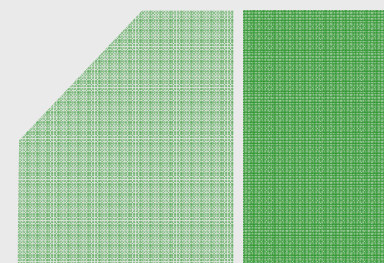
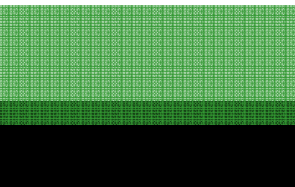
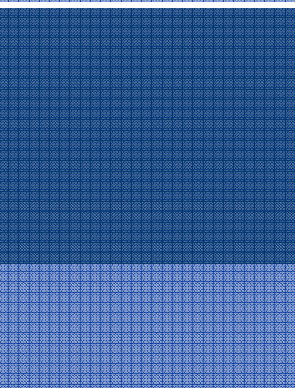


Ryan C. Barnett
Ryan.Barnett@breach.com

Dynamic Vulnerability Remediation with ModSecurity:
An Incident Response Approach



Dynamic Vulnerability Remediation: *What is it?*



The OWASP Foundation
<http://www.owasp.org>

What is Dynamic Vulnerability Remediation?

■ Known by many names

- ▶ Virtual Patching
- ▶ External Patching
- ▶ Just-in-time Patching

■ Definition

- ▶ A policy for an intermediary device (i.e. - Web Application Firewall - WAF) that is able to identify and block attempts to exploit a specific web application vulnerability.

■ Method

- ▶ The WAF analyzes transactions and intercepts attacks in transit, so malicious traffic never reaches the web application.

■ Result

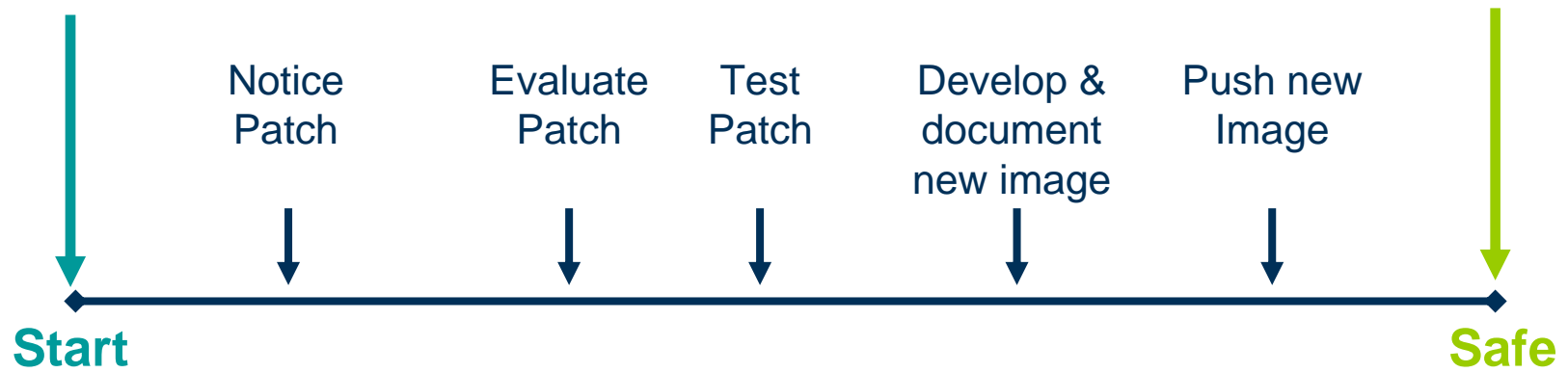
- ▶ Provides protection for a vulnerable web application.

Traditional Patching

Many Challenges and is Time Consuming

Vulnerability
Published and
Patch Released

Last System
Patched &
Rebooted



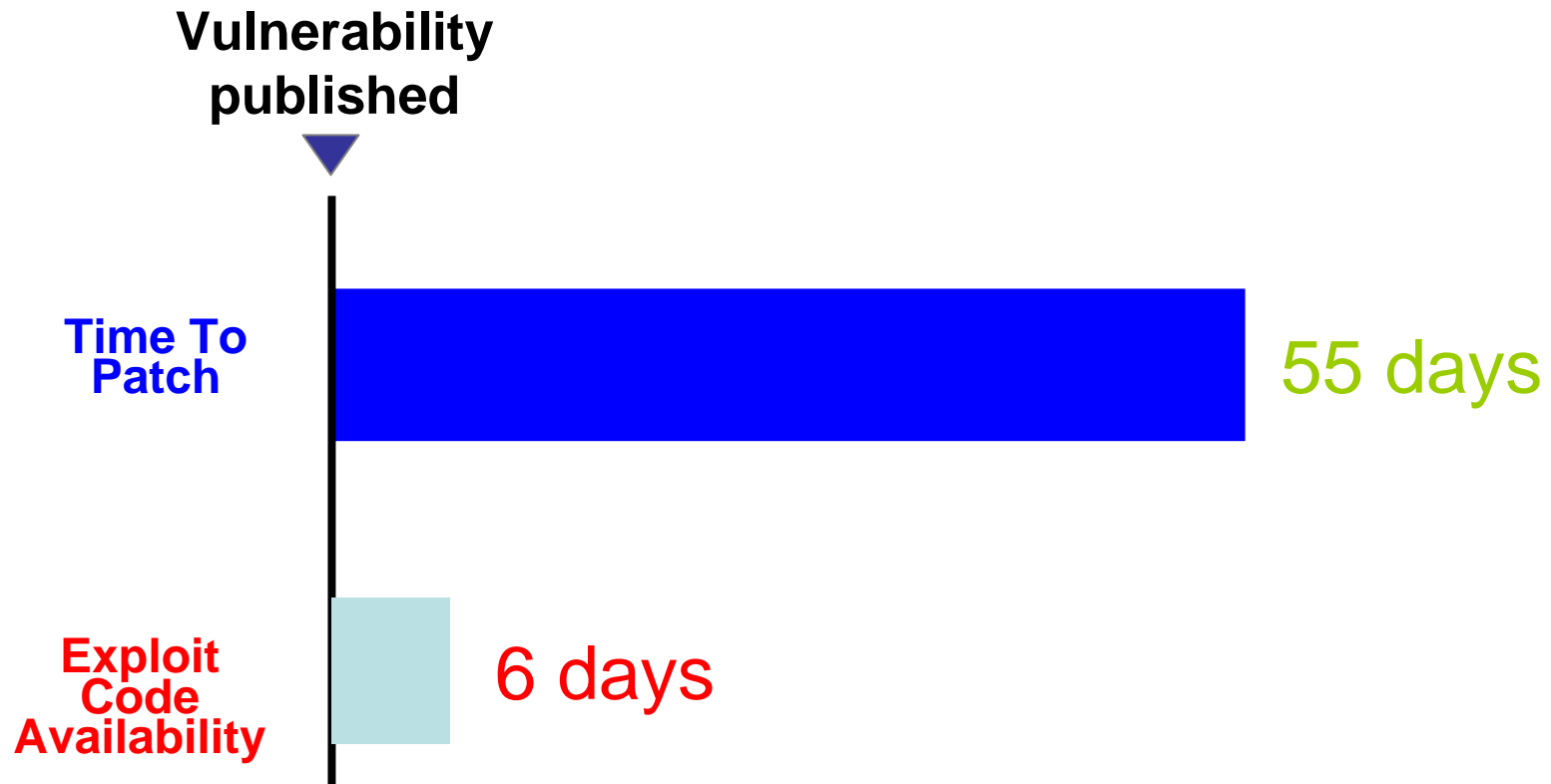
High value systems are difficult to patch:

- Patch may impact the system
- Patches inherently slow and expensive to test
- Most patches not designed to be easily reversible
- Service disruption or machine reboot



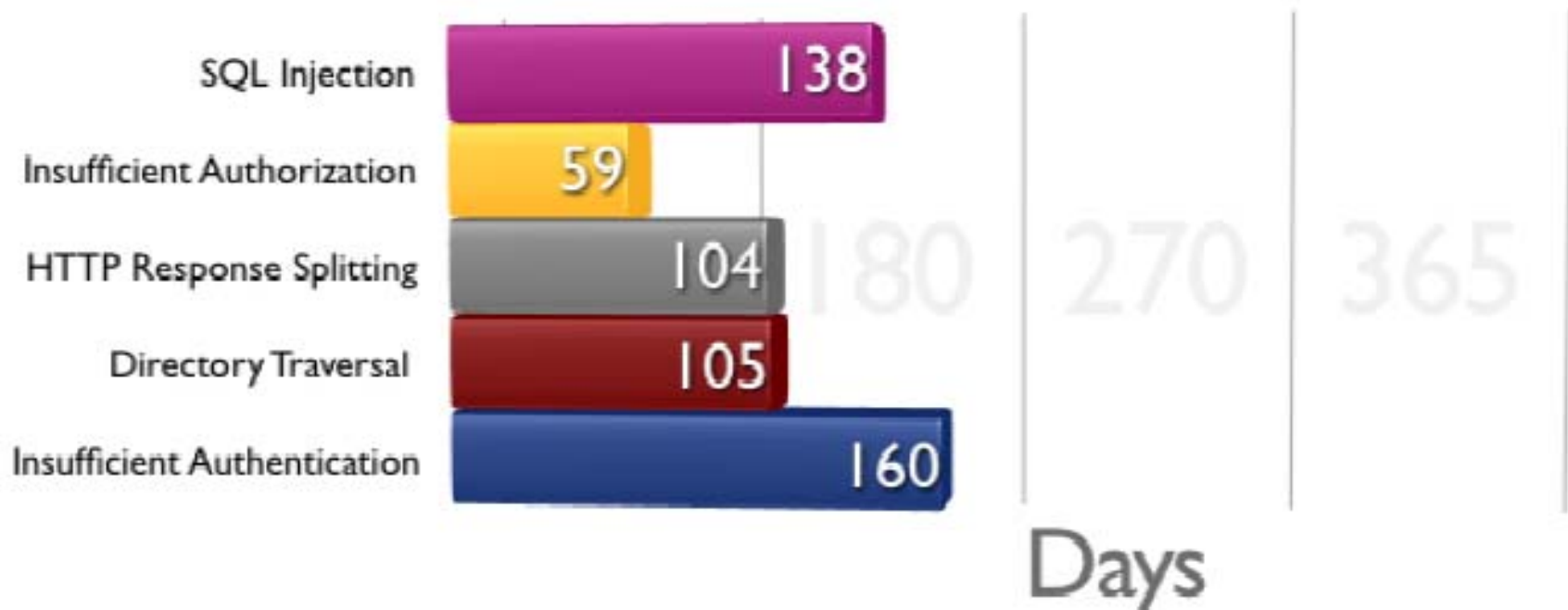
Traditional Patching:

A Race Against the Clock



Vulnerability Scanning Statistics

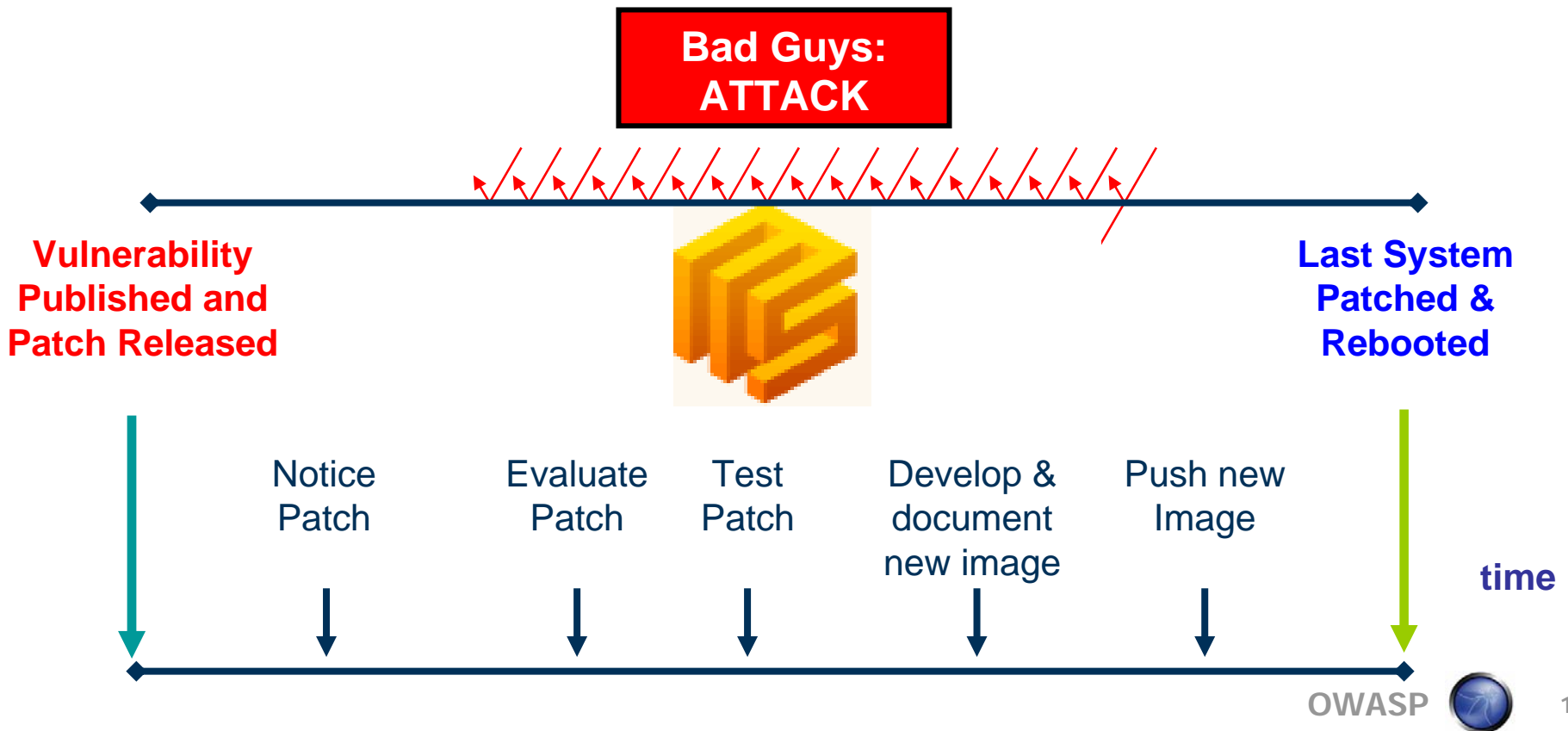
- Average # of days for the top 5 URGENT severity vulnerabilities to be fixed



- Traditional code fixes take too long...

Dynamic Vulnerability Remediation Concept

Preventing Exploitation During Patching

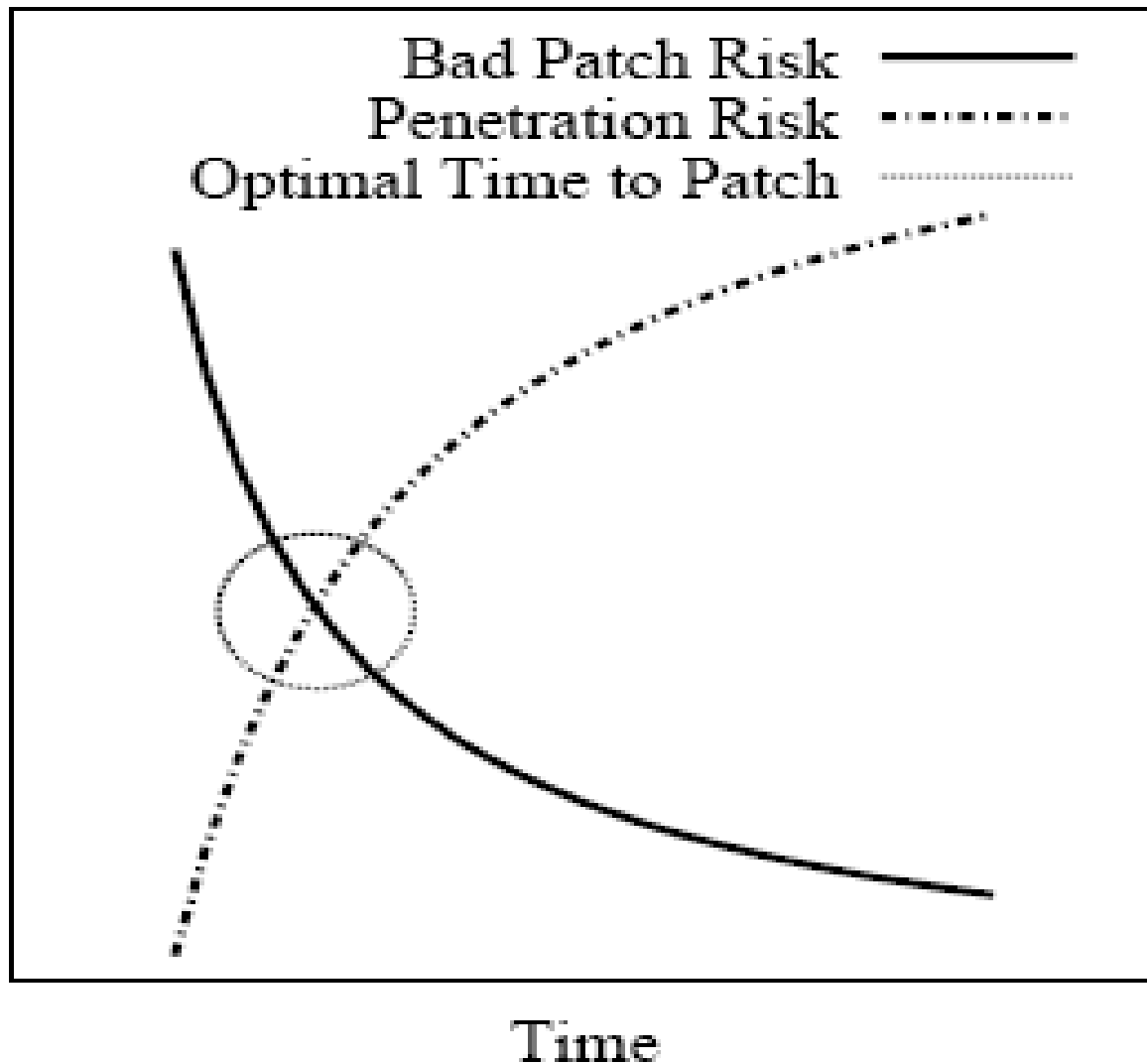


Value (1)

- Scalable solution as it is implemented in few locations vs. installing patches on all hosts.
- Reduces risk until a vendor-supplied patch is released or while a patch is being tested and applied.
- **Less likelihood of introducing conflicts as libraries and support code files are not changed.**

Traditional Patching Trade-Off:

Applying a Bad Patch vs. Exploit Exposure

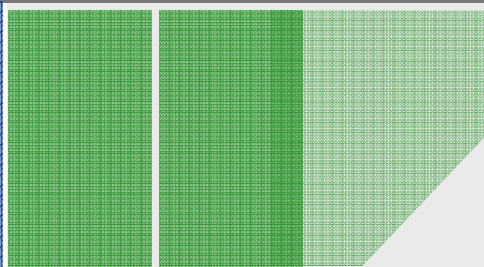
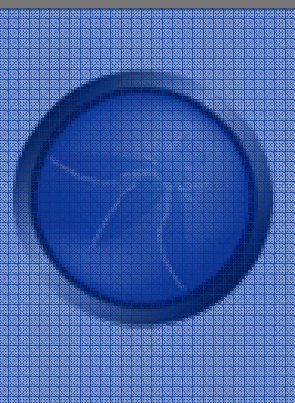


Value (2)

- Protection for mission-critical systems that may not be taken offline.
- Reduced or eliminated time and money spent performing emergency patching.
- **Allows organizations to maintain normal patching cycles.**

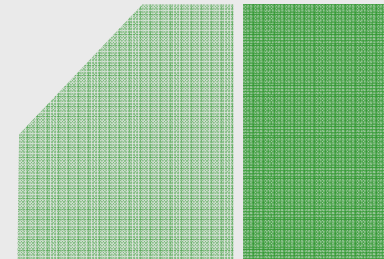
Why ModSecurity?

- Free - ☺
- Deep understanding of HTTP and HTML
 - ▶ Breaking up to individual fields: headers, parameters, uploaded files.
 - ▶ Validation of field attributes such as content, length or count
 - ▶ Correct breakup and matching of transactions and sessions.
 - ▶ Compensation for protocol caveats and anomalies, for example cookies.
- Robust parsing:
 - ▶ Unique parameters syntax
 - ▶ XML requests (SOAP, Web Services)
- Anti Evasion features:
 - ▶ Decoding
 - ▶ Path canonizations
 - ▶ Thorough understanding of application layer issues: Apache request line delimiters, PHP parameter names anomalies.
- Rules instead of signatures:
 - ▶ Sessions & state management, Logical operators, Control structures.

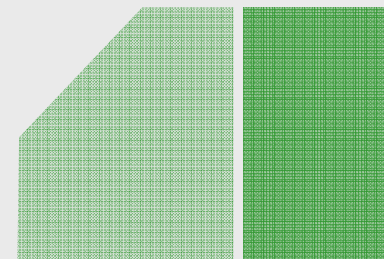


Dynamic Vulnerability Remediation with ModSecurity: *An Incident Response Approach*

Ryan C. Barnett
Ryan.Barnett@breach.com



Incident Response Phase 1: *Preparation*



Preparation Tasks:

You Can't Patch What You Don't Know

- Ensure that you are signed up for on all vendor alert mail-lists for commercial/open source software that you are using.
 - ▶ This should include the SANS @Risk weekly newsletter (<http://www.sans.org/newsletters/risk/>) as it includes Web Application vulnerability sections.

Summary of Updates and Vulnerabilities in this Consensus

Platform	Number of Updates and Vulnerabilities
-----	-----
Windows	4 (#1, #7, #9)
Other Microsoft Products	7 (#2, #5, #6, #10)
Third Party Windows Apps	16
Linux	3
Solaris	1
Unix	2
Novell	1
Cross Plat	11 (#3, #4, #8)
Web Application - Cross Site Scripting	19
Web Application - SQL Injection	28
Web Application	25
Network Device	

Preparation Tasks:

Deploy ModSecurity In Advance

- As time is critical during incident response, it would be a poor time to have to get approvals to install new software.
 - ▶ You can install ModSecurity in embedded mode on your Apache servers, or
 - ▶ Install ModSecurity on an Apache reverse proxy server. The advantage with this deployment is that you can create fixes for non-Apache servers.
- Even if you do not use ModSecurity under normal circumstances, it is best to have it “on deck” ready to be enabled if need be.

Preparation Tasks:

Pre-Authorization

- Virtual Patches need to be implemented ASAP so the normal governance processes and authorizations steps for standard software patches need to be expedited.
- Since virtual patches are not actually modifying source code, they do **NOT** need to have the same amount of regression testing as normal software patches.
- The authorization process should be similar to how your organization handles updates to AV/NIDS signatures.

Preparation Tasks:

Increase Audit Logging (1)

- The Common Log Format (CLF) that is extensively used by web servers does not contain enough detail to accurately identify or confirm exploit attempts.
- Critical data such as the full Request Headers and Request Body (such as POST payloads) are not normally logged.
 - ▶ For instance, here is an example log entry in CLF format –

```
80.87.72.6 - - [22/Apr/2007:18:55:53 --0400] \
"POST /xmlrpc.php HTTP/1.1" 200 293
```

- ▶ What was in the POST Payload???

Preparation Tasks:

Increase Audit Logging (2)

- SecAuditEngine handles the creation of audit logs.
- Possible values are:
 - ▶ On - log all transactions by default – can potentially consume a lot of resources.
 - ▶ Off - do not log transactions by default.
 - ▶ RelevantOnly - by default only log transactions that have triggered a warning or an error, or have a status code that is considered to be relevant (see SecAuditLogRelevantStatus).
- Recommend On for the following situations
 - ▶ Initial WAF deployment/testing.
 - ▶ Sensitive areas of web application.
 - ▶ Trap and Trace during Incident Response – use “ctl:auditEngine=On” when a rule fires.
- Auditing optimization options such as excluding “static” content can help to reduce load.

Preparation Tasks:

Increase Audit Logging (3)

--ddb9bf17-A--

[22/Apr/2007:18:55:53 --0400]

dGgsYX8AAAEABJkpY8AAACG 80.87.72.6 41376

192.168.1.133 80

--ddb9bf17-B--

POST /xmlrpc.php HTTP/1.1

TE: deflate,gzip;q=0.3

Connection: TE, close

Host: www.example.com

User-Agent: libwww-perl/5.805

Content-Length: 201

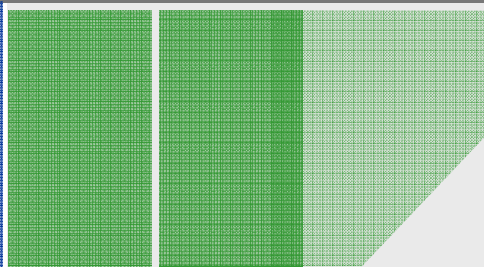
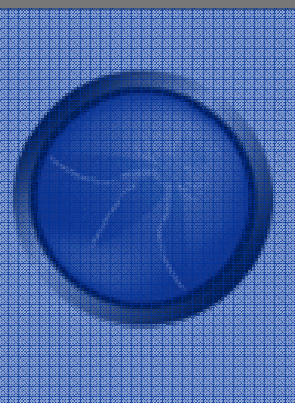
--ddb9bf17-C--

<?xml

version="1.0"?><methodCall><methodName>test.method
</methodName><params><param><value><name>', ' ')) ;ec
ho '_begin_';echo `id;ls /;w`;echo
'_end_';exit; /*</name></value></param></params></m
ethodCall>

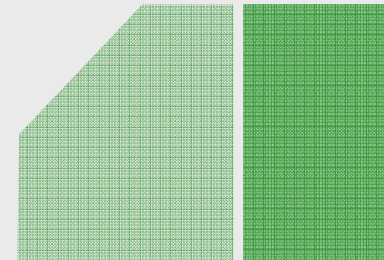
**POST Payload is
now available and
shows signs of OS
Command
injections.**



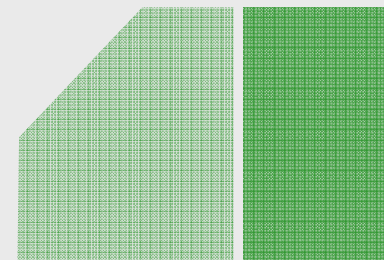


Dynamic Vulnerability Remediation with ModSecurity: *An Incident Response Approach*

Ryan C. Barnett
Ryan.Barnett@breach.com



Incident Response Phase 2: *Identification*



Vulnerability Discovery:

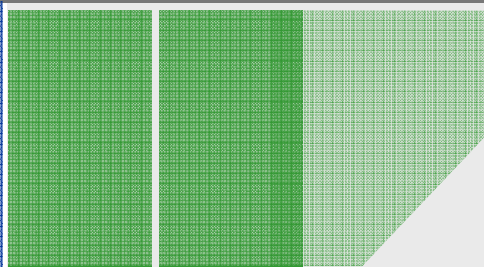
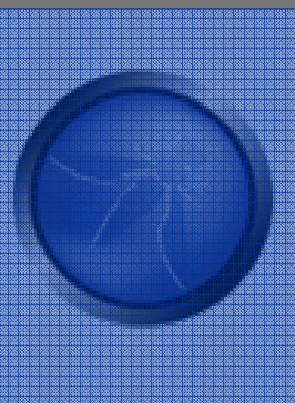
Proactive Identification

- Occur when an organization takes it upon themselves to assess their web security posture and conducts the following tasks:
 - ▶ Vulnerability assessment (internal or external) and penetration tests
 - ▶ Source code reviews
 - ▶ These tasks are extremely important for custom coded web applications.
- Output
 - ▶ Reports details on vulnerabilities.
- Action
 - ▶ Immediately create Virtual Patches.
 - ▶ Initiate normal source code fix SDLC

Vulnerability Discovery:

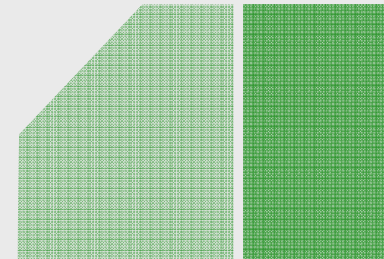
Reactive Identification

- Vendor contact (e.g. pre-warning)
 - ▶ Occurs when there a vendor discloses a vulnerability for commercial web application software that you are using.
- Public disclosure
 - ▶ Public vulnerability disclosure for commercial/open source web application software that you are using.
 - ▶ Threat Level is increased as more people know about the vulnerability.
- **Security incident**
 - ▶ Most urgent situation.
 - ▶ Remediation must be immediate.
 - ▶ Blocking only the source IP is not always possible as you may prevent legitimate users from accessing the application.
 - ▶ WAF rules are more flexible – it is not necessarily *where* you are coming from but *what* you are doing

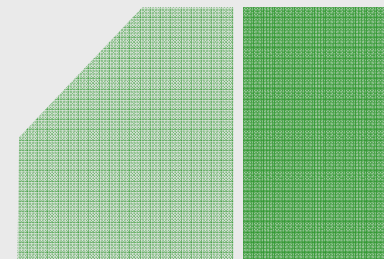


Dynamic Vulnerability Remediation with ModSecurity: *An Incident Response Approach*

Ryan C. Barnett
Ryan.Barnett@breach.com



Incident Response Phase 3: *Analysis*



Vulnerability Analysis Phase (1)

■ *What is the name of the vulnerability?*

- ▶ This means that you need to have the proper CVE name/number identified by the vulnerability announcement, vulnerability scan, etc...

■ *What is the impact of the problem?*

- ▶ It is always important to understand the level of criticality involved with a web vulnerability. Information leakages may not be treated in the same manner as an SQL Injection issue.

■ *What versions of software are affected?*

- ▶ You need to identify what versions of software are listed so that you can determine if the version(s) you have installed are affected.

■ *What configuration is required to trigger the problem or how to tell if you are affected by the problem?*

- ▶ Some vulnerabilities may only manifest themselves under certain configuration settings.

Vulnerability Analysis Phase (2)

■ *Is proof of concept exploit code available?*

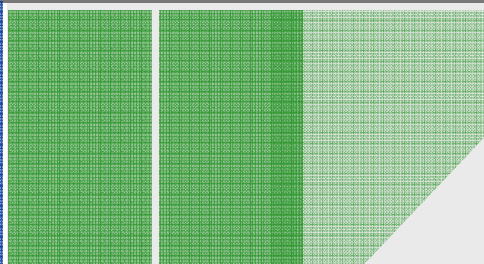
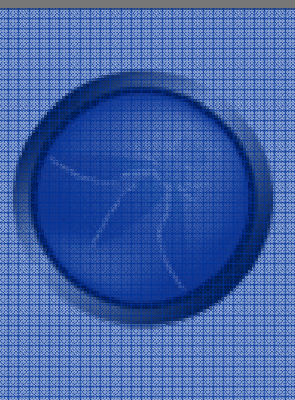
- ▶ Many vulnerability announcements have accompanying exploit code that shows how to demonstrate the vulnerability. If this data is available, make sure to download it for analysis. This will be useful later on when both developing and testing the Virtual Patch.

■ *Is there a work around available without patching or upgrading?*

- ▶ This is where Virtual Patching actually comes into play. It is a temporary work-around that will buy organizations time while they implement actual source code fixes.

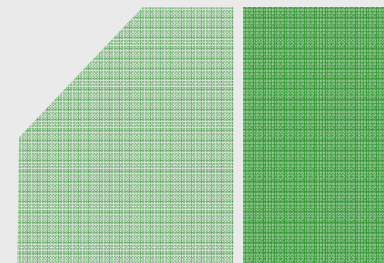
■ *Is there a patch available?*

- ▶ Unfortunately, vulnerabilities are often announced without an accompanying patch. This leaves organizations exposed and is why Virtual Patching has become an invaluable tool.
- ▶ If there is a patch available, then you initiate the proper patch management processes and simultaneously create a Virtual Patch

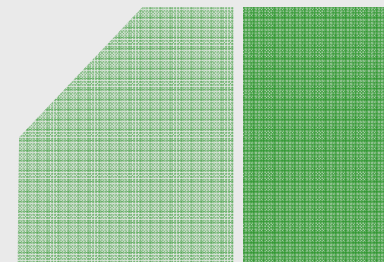


Dynamic Vulnerability Remediation with ModSecurity: *An Incident Response Approach*

Ryan C. Barnett
Ryan.Barnett@breach.com



Incident Response Phase 4: *Virtual Patch Creation*



Virtual Patch Writing Considerations

■ Minimize false negatives

- ▶ Do not miss attacks, even when the attacker intentionally tries to evade detection.
 - Attackers may try common evasion techniques such as using various encoding schemes or including null bytes.
 - Must therefore include normalization functions before applying rules.

■ Minimize false positives

- ▶ Do not ever block legitimate traffic under any circumstances.
- ▶ Most false positives arise due to one of the following:
 - A weaknesses in the engine or signature language that prevents the detection logic from being implemented with adequate precision.
 - Signatures being written without regard for false positives (in many cases it is a sloppiness problem).

Virtual Patch Goal:

Minimize False Negatives

- During vulnerability research, you must identify all of the *mandatory conditions* for an attack to succeed.
 - ▶ When testing proof-of-concept exploit code, if the attack succeeds even when a particular variable is set to a random value, that variable is not important for the patch creation.
- Given a set of criteria that must be satisfied for an attack to succeed, it is possible to describe patch logic that has zero false negatives.
 - ▶ Meaning an attack simply cannot succeed unless the associated request has exactly the characteristics that the patch is looking for.

Poor Rule Writing

Resulting in False Negatives

```
SecDefaultAction
    "log,deny,phase:2,status:500,t:urlDecodeUni,t:htmlEntityDe
    code,t:lowercase"
# WEB-CGI csSearch.cgi arbitrary command execution attempt
SecRule REQUEST_URI "/csSearch\.cgi\?" chain
SecRule REQUEST_URI "\`"

#generic SQL injection sigs using PCRE
SecRule REQUEST_URI|ARGS|REQUEST_BODY
    "/\w*(\x27|\')(\\x6F|o|\\x4F)(\x72|r|\x52)/ix"

#PHPNuke general SQL injection
SecRule REQUEST_URI "/modules\.php\?.*name=.*UNION.*SELECT"
```

Use of lowercase transformation function however the rule is written in upper-case

Converted Snort Rule – can't specify PCRE flags in this way.

Does the application accept POST requests?

An SQL injection does not have to use SELECT or UNION

Virtual Patch Goal:

No False Positives

- At this stage, the rule writer attempts to identify at least one characteristic that would *never* occur in normal traffic.
- A zero false negative patch is also a zero false positive patch if it is comprised of a characteristics that are both:
 - ▶ Anomalous compared to normal traffic, and
 - ▶ Critical to the attack's success
- Examples:
 - ▶ SQL Injection Attacks: special characters such as ' and %27 are provided in a particular value in particular web request.
 - ▶ PHP Remote File Include Attacks: a remote URL is provided in a particular value in a particular Web request.
 - ▶ Buffer Overflows: too much of a certain kind of data is provided to a specific variable in a particular parameter.

Virtual Patch Terminology:

Negative/Positive Security

- **Negative Security** is looking for what is dangerous such as known web attack signature strings or character sets outside of the normal alpha-numeric ASCII range

- ▶ **Example Vulnerability**

- If a semi-colon is passed to parameter A of application B, then an attacker can inject OS commands.

- ▶ **Example Negative Security Virtual Patch**

- Would be to look for a semi-colon being passed to parameter A in application B.

- **Positive Security** is the security model employed to validate acceptable input for all portions of the application

- ▶ **Example Vulnerability**

- If a semi-colon is passed to parameter A of application B, then an attacker can inject OS commands.

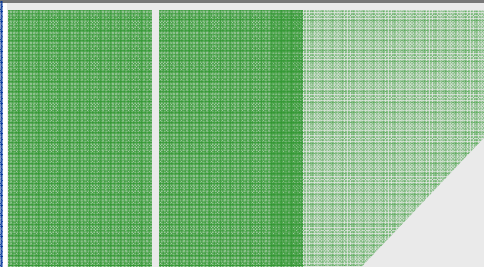
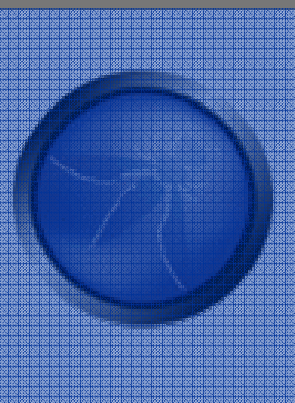
- ▶ **Example Positive Security Virtual Patch**

- Would be to enforce only digits for parameter A in application B.

Negative Security vs. Positive Security:

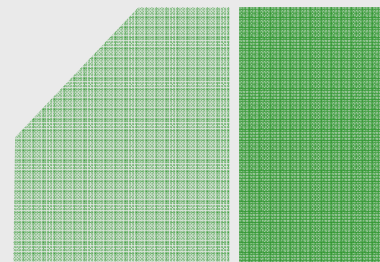
Which is Better?

- A Virtual Patch can employ either a negative or positive security model.
- Negative Security Rules
 - ▶ Can usually be implemented more quickly.
 - ▶ The issue is that evasions are more likely.
- Positive Security Rules
 - ▶ A positive security model provides better protection, however, it is often a manual process and thus is not scalable and difficult to maintain for large/dynamic sites.
 - ▶ A positive security model can be selectively employed when a vulnerability alert identifies a specific location with a problem.

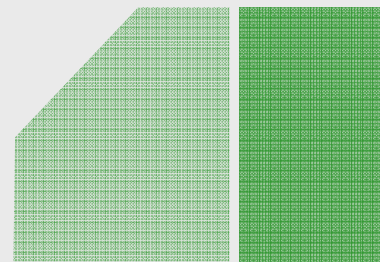


Dynamic Vulnerability Remediation with ModSecurity: *An Incident Response Approach*

Ryan C. Barnett
Ryan.Barnett@breach.com



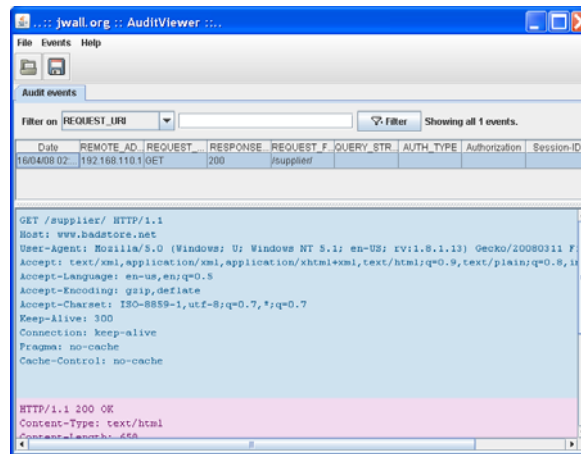
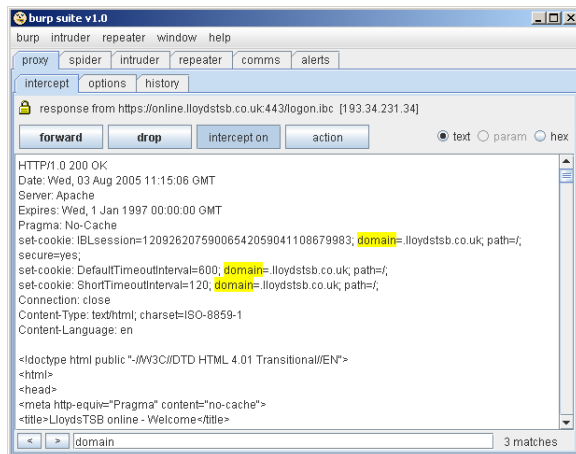
Incident Response Phase 5: *Implementation/Testing*



Implementation/Testing:

Testing Tools

- In order to accurately test out the Virtual Patch, it may be necessary to use an application other than a web browser.
- Some useful tools are –
 - ▶ Command line web clients such as Curl and Wget.
 - ▶ Local Proxy Servers such as WebScarab and Burp Proxy.
 - ▶ ModSecurity AuditViewer – can re-inject audit log data.
- These tools will allow you to manipulate the request data in any way desired.



curl
groks those URLs



Testing the Virtual Patch

- ## ■ You can use curl to send a test exploit request

```
$ curl -d "username=`perl -e 'print "0"x250'`"
http://www.example.com/isqlplus/login.uix
```

- This will result in the following request

```
POST /isqlplus/login.uix HTTP/1.1
User-Agent: curl/7.15.4 (i686-pc-cygwin) libcurl/7.15.4
  OpenSSL/0.9.8d zlib/1.2.3
Host: www.example.com
Accept: */*
Content-Length: 259
Content-Type: application/x-www-form-urlencoded
```

```
username=000000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000000000000  
0000000000000000000000000000000000000000000000000000000
```

Verify Patch Rule Triggered

- You should receive a 403 Forbidden Status Code
- This will also generate the following error log message

```
[Sat Jun 09 08:45:32 2007] [error] [client  
192.168.1.103] ModSecurity: Access denied with code  
403 (phase 2). Match of "rx ^(\w{0,32})$" against  
"ARGS:username" required. [file  
"/usr/local/apache/conf/rules/modsecurity_crs_15_cus  
tomrules.conf"] [line "1"] [msg "Oracle iSQLPlus  
login.uix username positive policy violation"]  
[hostname "www.example.com"] [uri  
"/isqlplus/login.uix"] [unique_id  
"hf3JssCoD4QAAApC88AAAAB" ]
```

ModSecurity Debug Log

- In order to verify exactly how your new rule is working, you should review the SecDebugLog file.
- The Debug log provides details on the rule processing order.
- You will most likely need to increase the SecDebugLogLevel directive setting to get enough detail to validate the patch processing.
- You can selectively increase the logging based on source IP address so that you don't impact performance on the web server.

Sample Debug Log Data

Recipe: Invoking rule 82211d8.

Executing operator !rx with param "^(POST)\$" against REQUEST_METHOD.

Target value: POST

Operator completed in 17 usec.

Rule returned 0.

No match, not chained -> mode NEXT_RULE.

Recipe: Invoking rule 82214b0.

Rule returned 0.

No match, not chained -> mode NEXT_RULE.

Recipe: Invoking rule 82360d0.

Executing operator !rx with param "^(\\w{0,32})\$" against
ARGS:username.

Target value:

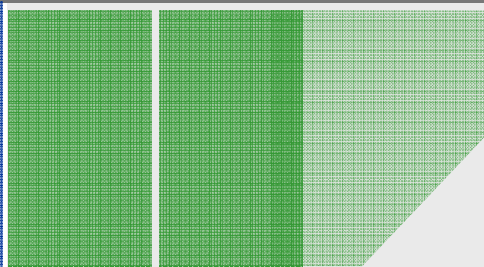
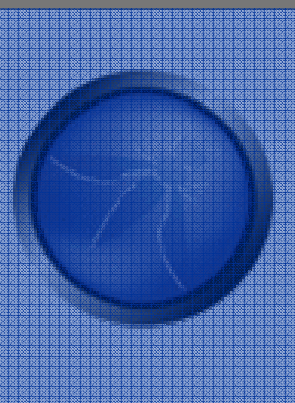
[illegible]

Operator completed in 13 usec.

Rule returned 1.

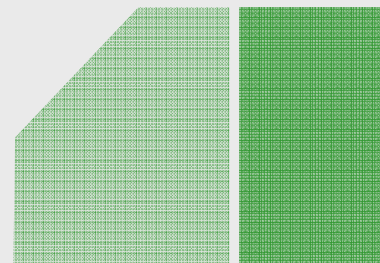
Match, intercepted -> returning.

```
Access denied with code 501 (phase 2). Match of "rx ^(\w{0,32})$"
against "ARGS:username" required. [id "1"] [msg "Postparameter
username failed validity check. Value domain: Username."] [severity
"ERROR"]
```

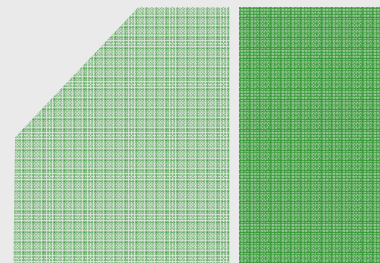



Dynamic Vulnerability Remediation with ModSecurity: *An Incident Response Approach*

Ryan C. Barnett
Ryan.Barnett@breach.com



Incident Response Phase : *Recovery/Follow-Up*



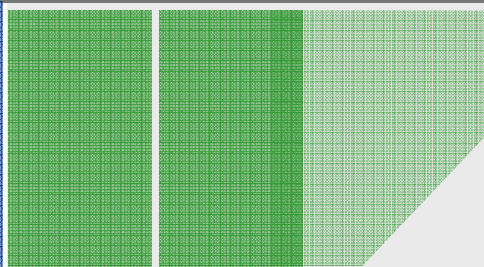
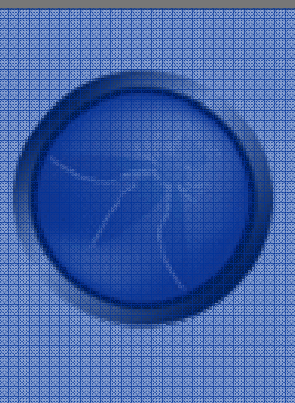
Recovery and Follow-Up

■ *Track Virtual Patches*

- ▶ Although you may need to expedite the implementation of Virtual Patches, you should still track them in your normal Patch Management processes.
- ▶ This means that you should create proper change request tickets, etc...

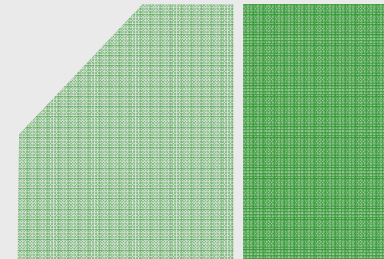
■ *Periodic Re-Evaluations*

- ▶ You should have periodic re-assessments to verify if/when you can remove previous Virtual Patches once the web application code has been updated with the real software patch.
- ▶ Many people opt to keep Virtual Patches in place due to better identification/logging vs. application or db capabilities.

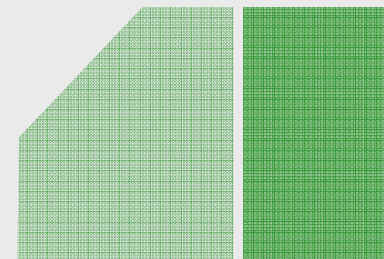


Dynamic Vulnerability Remediation with ModSecurity: *An Incident Response Approach*

Ryan C. Barnett
Ryan.Barnett@breach.com



Example : Public Vulnerability Announcement
*Google Mini Search Appliance IE Parameter
Cross-Site Scripting Vulnerability*



SANS @Risk Announcement

- 07.41.32 - CVE: Not Available
- Platform: Web Application - Cross Site Scripting
- Title: Google Mini Search Appliance IE Parameter Cross-Site Scripting
- Description: Google Mini Search Appliance is an integrated hardware and software enterprise search solution. The application is exposed to a cross-site scripting issue because it fails to sanitize the "ie" input parameter in the "search" script. Google Mini Search Appliance version 3.4.14 is affected.
- Ref: <http://www.securityfocus.com/bid/25894>

SecurityFocus Vulnerability Data

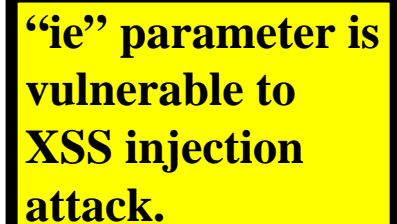
[info](#)[discussion](#)[exploit](#)[solution](#)[references](#)

Google Mini Search Appliance IE Parameter Cross-Site Scripting Vulnerability

An attacker can exploit this issue by enticing an unsuspecting user to follow a malicious URI.

The following proof-of-concept URI is available:

`http://www.example.com/search?ie=[Evil Code]`
`&site=x&output=xml_no_dtd'&client=x&proxystylesheet=x'`



“ie” parameter is vulnerable to XSS injection attack.

ModSecurity Core Rules:

Generic XSS Rules

SecRuleEngine DetectionOnly

...

SecRule REQUEST_FILENAME|ARGS|ARGS_NAMES|REQUEST_HEADERS|\

XML:/*|!REQUEST_HEADERS:Referer

```
"(?:\b(?:(:type\b\W*?\b(?:text\b\W*?\b(?:j(?:ava)?|ecma|v
b)|application\b\W*?\b(?:java|vb))script|c(?:opyparentfol
der|reatetextrange)|get(?:special|parent)folder)\b|on(?:(:
:mo(?:use(?:o(?:ver|ut)|down|move|up)|ve)|key(?:press|down
|up)|c(?:hange|lick)|s(?:elec|ubmi)t|(:un)?load|dragdrop|
resize|focus|blur)\b\W*?=|abort\b)|(:l(?:owsrc\b\W*?\b(?:
(?:java|vb)script|shell)|ivescript)|(:href|url)\b\W*?\b(?:
(?:java|vb)script|shell)|background-
image|mocha):|s(?:(:type\b\W*=.*\bexpression\b\W*|etimeo
ut\b\W*?)\(|rc\b\W*?\b(?:(:java|vb)script|shell|http):)|a
(?:ctivexobject\b|lert\b\W*?\(|))|<(:(:body\b.*?\b(?:back
groun|onloa)d|input\b.*?\btype\b\W*?\bimage|script|meta)\b
|!\[CDATA\(|(:\.(?:(:execscrip|addimpor)t|(:fromcharco
d|cooki)e|innerHTML)|\@import)\b)" \
```

```
"deny,capture,ctl:auditLogParts=+E,log,auditlog,msg:'Cross
-site Scripting (XSS) Attack. Matched sign ature
<{%TX.0}>',,id:'950004',severity:'2'"
```

ModSecurity Core Rules:

Targeted XSS Blocking

SecRuleEngine DetectionOnly

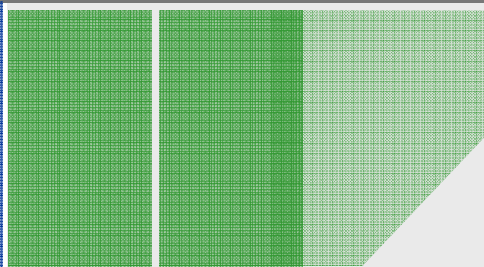
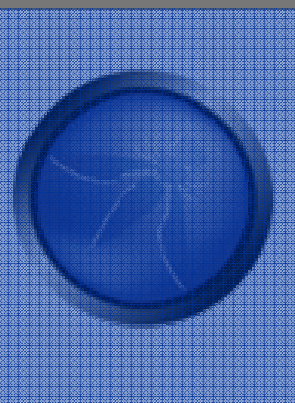
<Location /search>

SecRule ARGS:ie

```
"(?:\b(?:(:type\b\W*?\b(?:text\b\W*?\b(?:j(?:ava)?|ecma|vb)|application\b\W*?\b(?:java|vb))script|c(?:opyparentfolder|reatetextrange)|get(?:special|parent)folder)\b|on(?:(:mo(?:use(?:o(?:ver|ut)|down|move|up)|ve)|key(?:press|down|up)|c(?:hange|lick)|s(?:elec|ubmi)t|(:un)?load|dragdrop|resize|focus|blur)\b\W*?=|abort\b)|(:l(?:owsrc\b\W*?\b(?:(:java|vb)script|shell)|ivescript)|(:href|url)\b\W*?\b(?:(:java|vb)script|shell)|backgroundimage|mocha):|s(?:(:type\b\W*=.*\bexpression\b\W*|etimeout\b\W*?)|(rc\b\W*?\b(?:(:java|vb)script|shell|http):)|a(?:ctivexobject\b|lert\b\W*?\b|))|<(:(:body\b.*?\b(?:backgroun|onload|input\b.*?\btype\b\W*?\bimage|script|meta)\b|!\[CDATA\[)|(:\.(?:(:execscrip|addimpor)t|(:fromcharcod|cooki)e|innerhtml)|\@import)\b)" \
```

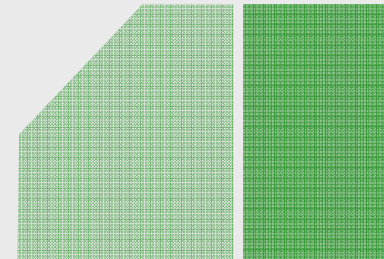
"deny,capture,ctl:ruleEngine=On,ctl:auditLogParts=+E,log,auditlog,msg:Google Mini Search Appliance IE Parameter Cross-Site Scripting Attack. Matched signature
<%{TX.0}>',id:'100000',severity:'2'"

</Location>

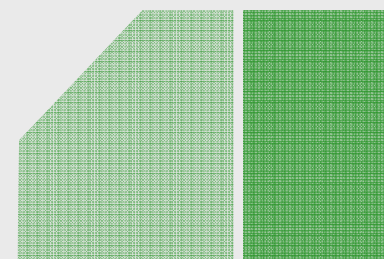


Dynamic Vulnerability Remediation with ModSecurity: *An Incident Response Approach*

Ryan C. Barnett
Ryan.Barnett@breach.com



Example : Source Code Review *Buffer Overflow/Authentication Bypass*

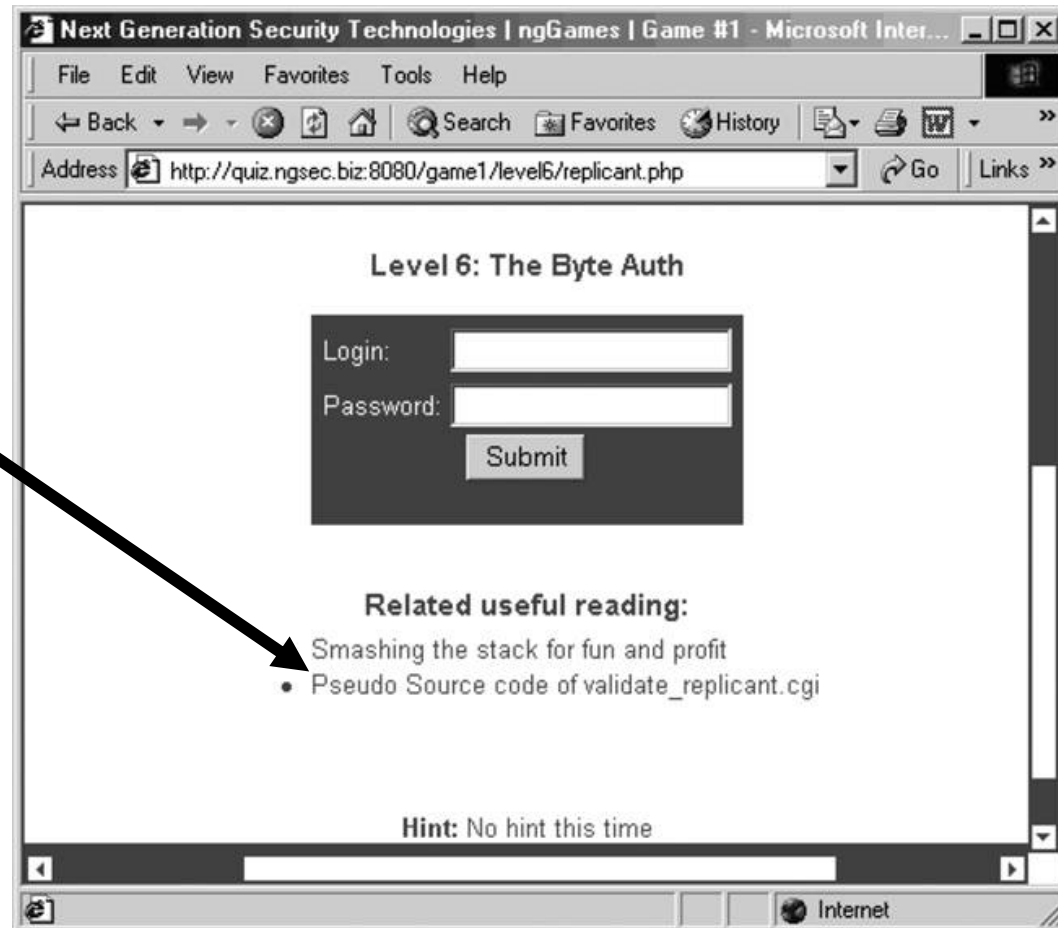


Source Code Review:

Authentication Bypass

- Let's say that a source code review was conducted on the login page of your

Example Pseudo code shows the vulnerability.



Problems In The Code

```
void show_error(void) {  
    // AUTHENTICATION ERROR  
    exit(-1);  
}  
int main(int argc, char **argv) {  
    char error_on_auth='1';  
    char user[128];  
    char pass[128];  
    char *ch_ptr_begin;  
    char *ch_ptr_end;  
  
    if ((strcmp(user,GOOD_USER)==0) && (strcmp(pass,GOOD_PASS)==0))  
        error_on_auth='0';  
  
    if (error_on_auth=='0')  
        // AUTHENTICATION OK!!  
  
    } else {  
  
        // AUTHENTICATION ERROR  
        show_error();  
  
    }
```

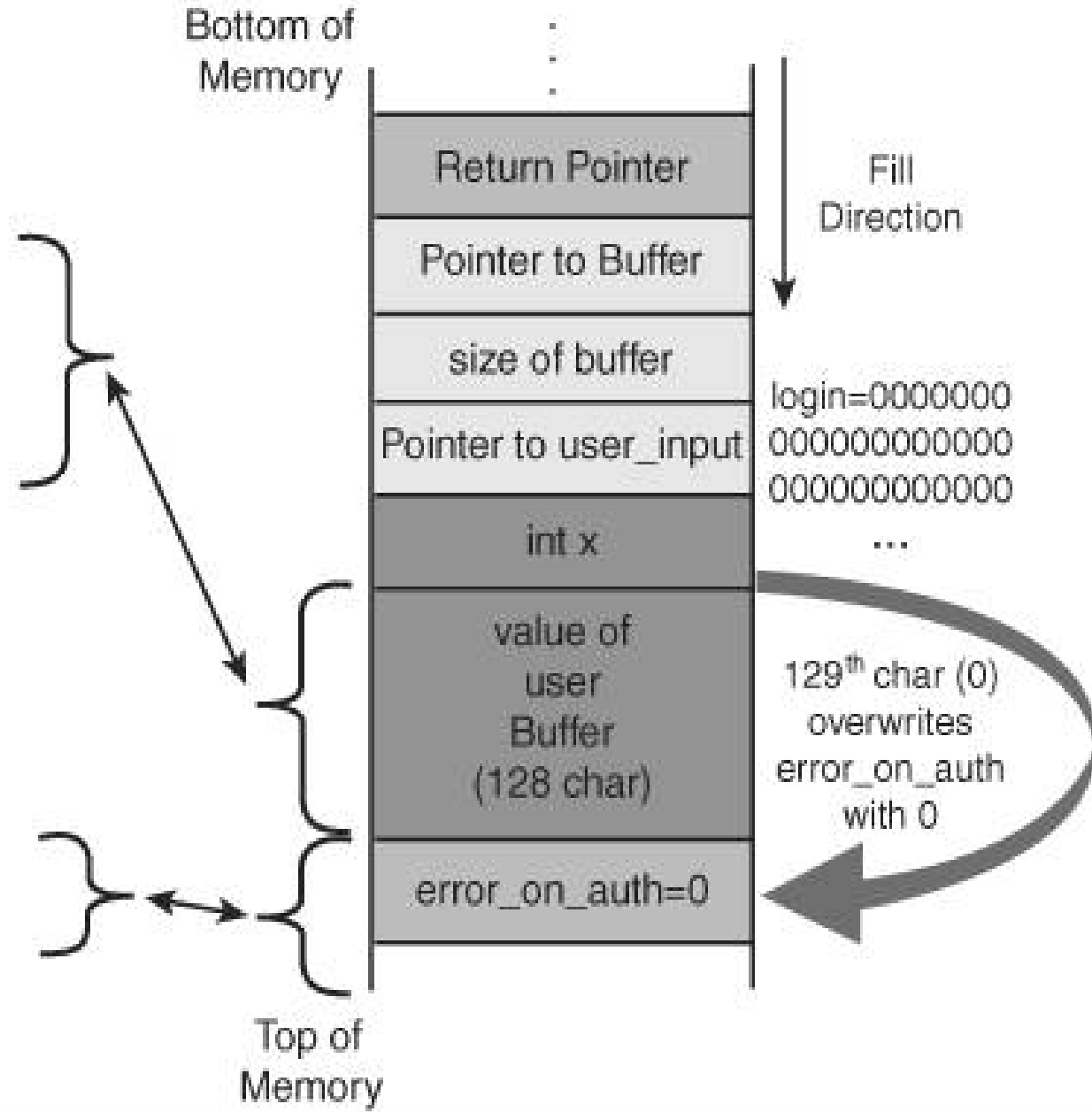
user and
error_on_auth
variables are declared
next to each other

If error_on_auth
variable is equal to 0,
then the user is
authenticated



Byte Auth Stack View

```
int main (int argc, char  
** argv) {  
char error_on_auth='1';  
char user [128];  
--CUT--  
*(ch_ptr_end++)='\0';  
strcpy(user,ch_ptr_begin);  
--CUT--  
if  
((strcmp(user,GOOD_USER)==0  
) &&  
(strcmp(pass,GOOD_PASS)==0)  
) error_on_auth='0';  
  
if (error_on_auth=='0') {  
  
// Authentication OK!!
```



Negative Security Virtual Patch

- Only apply this rule to the proper CGI script
- Inspect the "login" argument
- Block if the parameter payload is greater then 128 characters in length

```
<Location /cgi-bin/validate_replicant.cgi>  
SecRule ARGS:login "^.{128,}$"  
</Location>
```

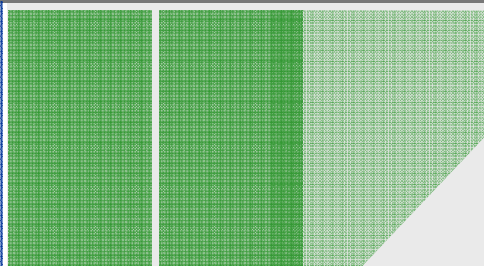
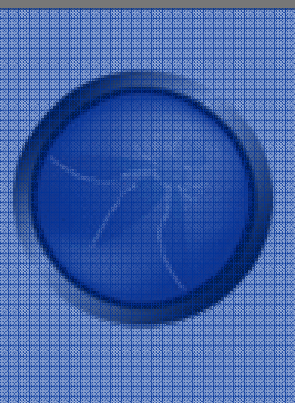
ModSecurity 2.5 Version

```
<Location /cgi-bin/validate_replicant.cgi>  
SecRule ARGS:login "@gt 128" t:length  
</Location>
```

Positive Security Virtual Patch

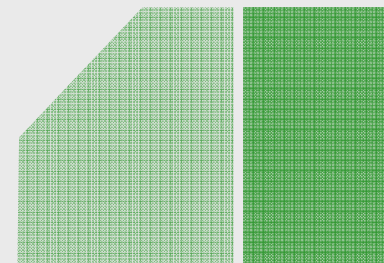
- Only apply this rule to the proper CGI script
- ARGS should only appear on POST payload and not in a Query_String
- Ensure that there are only 2 arguments supplied and that only 1 argument is named "login"
- Inspect the "login" argument
 - ▶ Block if the parameter payload is not an upper/lowercase letter between 0 and 25 characters in length
- Apply anti-evasion functions

```
<Location /cgi-bin/validate_replicant.cgi>  
SecRule &ARGS_GET_NAMES "@gt 0"  
SecRule &ARGS_POST_NAMES "!@eq 2"  
SecRule &ARGS:login "!@eq 1"  
SecRule ARGS:login "!^[a-zA-Z]{0,25}$" \  
"deny,log,t:urlDecodeUni,t:htmlEntityDecode, \  
t:lowercase,t:removeWhitespace,t:removeComments"  
</Location>
```

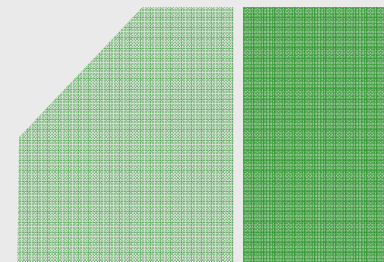
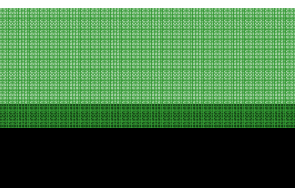
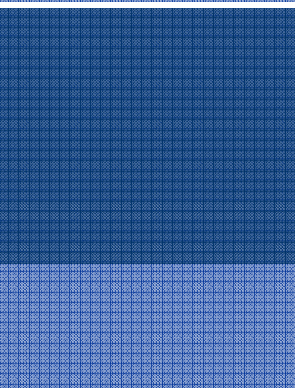


Dynamic Vulnerability Remediation with ModSecurity: *An Incident Response Approach*

Ryan C. Barnett
Ryan.Barnett@breach.com



Example : Vulnerability Scan Results *OS Command Injection in Web Services*



Vulnerability Scanning Vendor Report

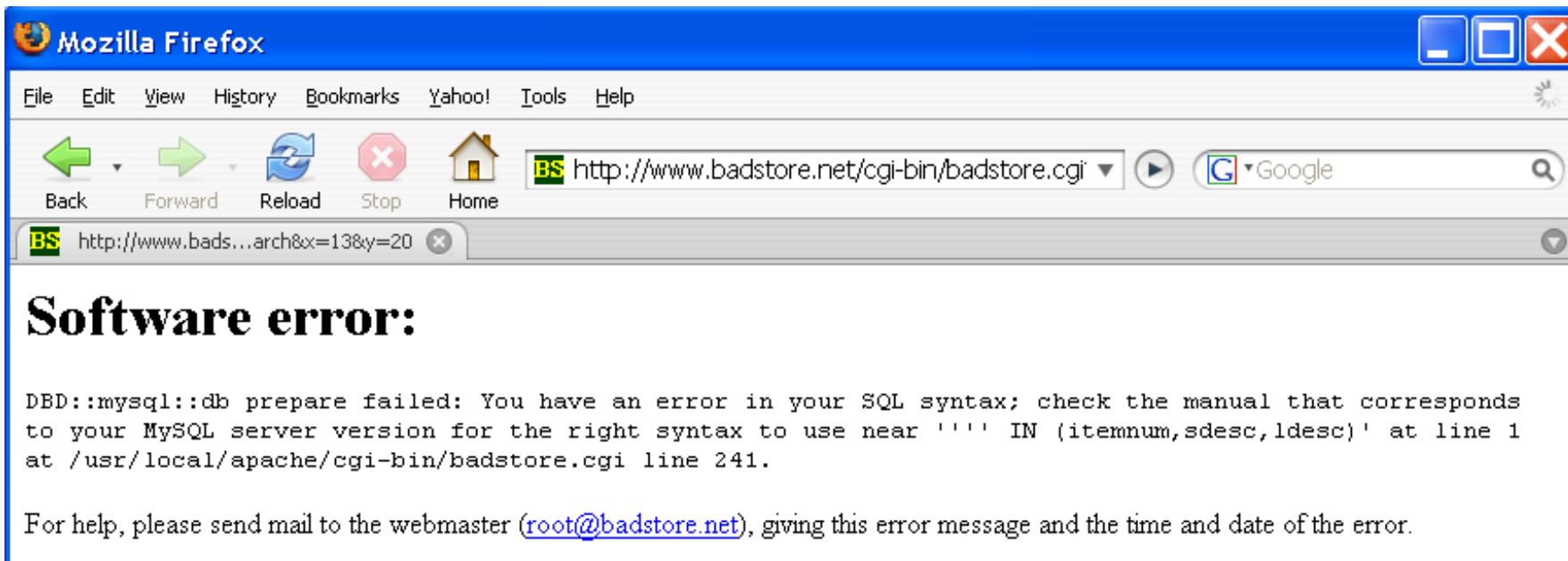
- Let's say that your vulnerability scanning vendor identifies an SQL Injection problem with the "search" function of your application.

If you inject a meta-character (') into the search field, it responds with a DB Error



Exploit: /cgi-bin/badstore.cgi?searchquery=%27&action=qsearch&x=13&y=20

Resulting Page Shows DB Error Message



The screenshot shows a Mozilla Firefox browser window. The address bar displays the URL `http://www.bads...arch&x=13&y=20`. The main content area displays a database error message in a monospaced font.

Software error:

`DBD::mysql::db prepare failed: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' IN (itemnum,sdesc,ldesc)' at line 1 at /usr/local/apache/cgi-bin/badstore.cgi line 241.`

For help, please send mail to the webmaster (root@badstore.net), giving this error message and the time and date of the error.

ModSecurity Alerts on the DB Error Message

ModSecurityProM1100

BREACH

Dashboard Alerts Sensor Sites Search Reports Administration About

HTTP Transaction: 4387 (2008-02-19 08:49:30)

Site	Badstore
Hostname	www.badstore.net
Method	GET
URI	/cgi-bin/badstore.cgi


Alerts

Parameters

Request

Response

Rules

<input type="checkbox"/>	ID	Severity	Message
1 <input type="checkbox"/>	970003	 WARN (4)	SQL Information Leakage Warning. Pattern match "(?:\b(?:?:s(?:?:elect list because it is not contained in(?:?:an aggregate function and there is no(?:?:either an aggregate function or the) GROUP BY clause(?:?:applied argument is not a valid(?:?:(?:?:M(?:?:S(?:?: Postgre)SQL(?:?:O(?:?:racle DBC))))S(?:?:yntax error converti ...)" at RESPONSE_BODY.

* Select an ID to create an exception

Create Exception

Transaction ID	4387 < Previous Next > << Back to list
Sensor	M1100 - Default sensor
Sensor Tx ID	JFEmHX8AAAEAAFPvYHYAAEA
Timestamp	2008-02-19 08:49:30 (received at 2008-02-19 08:49:31)
Duration	346.18 msec
Source	192.168.110.1 / 3987 (NX)
Destination	192.168.110.140 / 80
Server	ModSecurityProxy/1.6.0 (Unix) mod_ssl/1.6.0 OpenSSL/0.9.8g
Producer	ModSecurity for Apache/2.5.1-breach2 (http://www.modsecurity.org/); Enhanced ruleset/1.6.1.
Session ID	-
User ID	-



Positive Security Virtual Patch

ModSecurityProM1100

[Dashboard](#) [Alerts](#) [Sensor](#) [Sites](#) [Search](#) [Reports](#) [Administration](#) [About](#)

Edit Site - Advanced


Custom Directives

Custom Rules Before:

```
SecRule REQUEST_FILENAME "@streq /cgi-bin/badstore.cgi" "phase:2,t:none,chain,block"
SecRule ARGS_GET:searchquery|"!^\\d{1,4}$" "ctl:ruleEngine=On"
```



Searchquery Data Is Now Validated








ModSecurityProM100



Dashboard Alerts Sensor Sites Search Reports Administration About

All Active Alerts

Update & Close Hold Remove Hold Delete

Group by: None Refresh every 60 seconds Update / Refresh

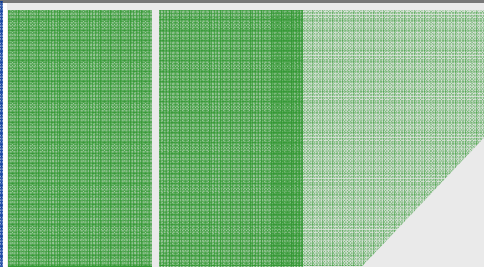
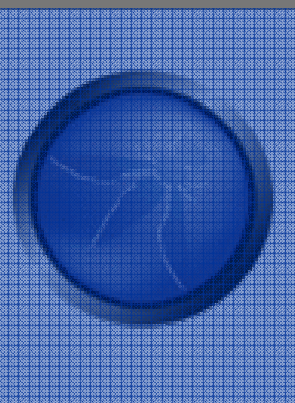
Severities:  Emergency  Alert  Critical  Error  Warning  Notice  Info

<input type="checkbox"/>	▼ ID	Date/Time	Source/Port	Hostname/Method/URI	Severity
<input type="checkbox"/>	4389	2008-02-19 09:08:36	192.168.110.1 PORT: 4798	HOSTNAME: www.badstore.net METHOD: GET URI: /cgi-bin/badstore.cgi  SearchQuery Parameter Violation.	 ERROR (3)

Results 1 - 1 of 1.

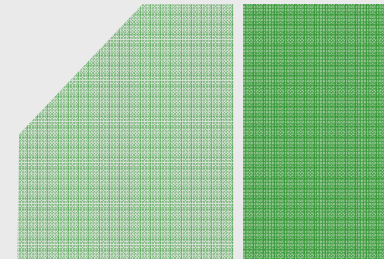
Resolution: Not resolved Category: Undetermined Comment:

Update & Close Hold Remove Hold Delete

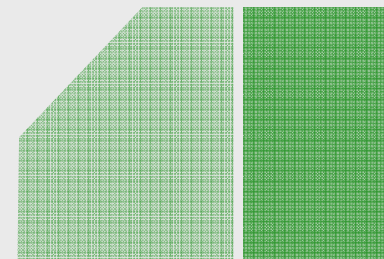


Dynamic Vulnerability Remediation with ModSecurity: *An Incident Response Approach*

Ryan C. Barnett
Ryan.Barnett@breach.com



Example : Real Customer Incident *SQL Injection*



eCommerce Customer Complaints

- Several of eCommerce customers notified the fraud division.
- They complained that their credit card info was being stolen from their site.
- The eCommerce company could not track down the problem due to poor audit logging.
 - ▶ They could find no evidence of tampering in the MS-SQL DB logs
- They contacted Breach and we deployed our ModSecurity appliance in DetectionOnly mode.
- We quickly identified the problem...

SQL Injection: Reconnaissance Probe

Request Details

```
GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20@@version%2b'/'%2b@ \
@servername%2b'/'%2bdb_name()%2b'/'%2bsystem_user))--sp_password HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
User-Agent: Microsoft URL Control - 6.00.8862
Host: www.example.com
X-Forwarded-For: 222.252.135.128
Connection: Keep-Alive
Cache-Control: no-cache, bypass-client=222.252.135.128
```

SQL Injection String

Request Details

```
GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20@@version%2b'/'%2b@ \
@servername%2b'/'%2bdb_name()%2b'/'%2bsystem_user))--sp_password HTTP/1.1
```

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*

User-Agent: Microsoft URL Control - 6.00.8862

Host: www.example.com

X-Forwarded-For: 222.252.135.128

Connection: Keep-Alive

Cache-Control: no-cache, bypass-client=222.252.135.128

Targeting Database Variables

Request Details

```
GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20@@version%2b'/'%2b@
@servername%2b'/'%2bdb_name()%2b'/'%2bssystem_user))--sp_password HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
User-Agent: Microsoft URL Control - 6.00.8862
Host: www.example.com
X-Forwarded-For: 222.252.135.128
Connection: Keep-Alive
Cache-Control: no-cache, bypass-client=222.252.135.128
```


DB Audit Log Evasion Attempt

Request Details

```
GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20@@version%2b'/'%2b@ \
@servername%2b'/'%2bdb_name()%2b'/'%2bsystem_user))--sp_password HTTP/1.1
```

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*

User-Agent: Microsoft URL Control - 6.00.8862

Host: www.example.com

X-Forwarded-For: 222.252.135.128

Connection: Keep-Alive

Cache-Control: no-cache, bypass-client=222.252.135.128

SQL Injection Response

Response Details

HTTP/1.1 500 Internal Server Error

Content-Length: 598

Content-Type: text/html

Cache-control: private

Set-Cookie: ASPSESSIONIDCCQCSRDQ=EHEPIKBBBFLOFIFOBPCJDBGP; path=/
Connection: close

<p>Microsoft OLE DB Provider for ODBC Drivers e \

<p>

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax \

.May 3 2005 23:18:38

.Copyright (c) 1988-2003 Microsoft Corporation

.Standard Edition on Windows NT 5.2 (Build 3790: Service Pack 1)

/EXAMPLE_SQL/OPT/OPT2' to a column of data type int.

500 Status Code and DB Errors

Response Details

HTTP/1.1 500 Internal Server Error

Content-Length: 598

Content-Type: text/html

Cache-control: private

Set-Cookie: ASPSESSIONIDCCQCSRQ=EHEPIKBBBFLOFIFOBPCJDBGP; path=/
Connection: close

<p>Microsoft OLE DB Provider for ODBC Drivers e \n
rror '80040e07'

<p>

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax \n
error converting the nvarchar value 'Microsoft SQL Server 2000 - 8.00.2039 (Int \n
el X86)

.May 3 2005 23:18:38

.Copyright (c) 1988-2003 Microsoft Corporation

.Standard Edition on Windows NT 5.2 (Build 3790: Service Pack 1)

/EXAMPLE_SQL/OPT/OPT2' to a column of data type int.

Includes Results of Variable Query

Response Details

HTTP/1.1 500 Internal Server Error

Content-Length: 598

Content-Type: text/html

Cache-control: private

Set-Cookie: ASPSESSIONIDCCQCSRDQ=EHEPIKBBBFLOFIFOBPCJDBGP; path=/
Connection: close

<p>Microsoft OLE DB Provider for ODBC Drivers e \n
rror '80040e07'

<p>

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax \n
error converting the nvarchar value 'Microsoft SQL Server 2000 - 8.00.2039 (Int \n
el X86)

.May 3 2005 23:18:38

.Copyright (c) 1988-2003 Microsoft Corporation

.Standard Edition on Windows NT 5.2 (Build 3790: Service Pack 1)

/EXAMPLE_SQL/OPT/OPT2' to a column of data type int.

SQL Injection: Stealing Customer Data

Request Details

```
GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20top%201%20convert(v \
archar,isnull(convert(varchar,OR_OrderDate),'NULL'))%2b'/'%2bconvert(varchar,isnull(conver \
t(varchar,OR_OrderID),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_FirstName \
),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_LastName),'NULL'))%2b'/'%2bco \
nvert(varchar,isnull(convert(varchar,OR_OrderAddress),'NULL'))%2b'/'%2bconvert(varchar,isn \
ull(convert(varchar,OR_OrderCity),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar, \
OR_OrderZip),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_OrderState),'NULL' \
))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_OrderCountry),'NULL'))%2b'/'%2bconver \
t(varchar,isnull(convert(varchar,OR_CCardName),'NULL'))%2b'/'%2bconvert(varchar,isnull(con \
vert(varchar,OR_CCardType),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_CCar \
dNumberenc),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_CCardExpDate),'NULL \
'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_CCardSecurityCode),'NULL'))%2b'/'%2b \
convert(varchar,isnull(convert(varchar,OR_Email),'NULL'))%2b'/'%2bconvert(varchar,isnull(c \
onvert(varchar,OR_Phone1),'NULL'))%20from%20Orders%20where%20OR_OrderID=47699)--sp_passwo \
rd HTTP/1.1
```

Targeting Credit Card Data

Request Details

```
GET /cart/loginexecute.asp?LoginEmail='%20or%201=convert(int,(select%20top%201%20convert(v \
archar,isnull(convert(varchar,OR_OrderDate),'NULL'))%2b'/'%2bconvert(varchar,isnull(conver \
t(varchar,OR_OrderID),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_FirstName \
),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_LastName),'NULL'))%2b'/'%2bco \
nvert(varchar,isnull(convert(varchar,OR_OrderAddress),'NULL'))%2b'/'%2bconvert(varchar,isn \
ull(convert(varchar,OR_OrderCity),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar, \
OR_OrderZip),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_OrderState),'NULL' \
))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_OrderCountry),'NULL'))%2b'/'%2bconver \
t(varchar,isnull(convert(varchar,OR_CCardName),'NULL'))%2b'/'%2bconvert(varchar,isnull(con \
vert(varchar,OR_CCardType),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_CCar \
dNumberenc),'NULL'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_CCardExpDate),'NULL \
'))%2b'/'%2bconvert(varchar,isnull(convert(varchar,OR_CCardSecurityCode),'NULL'))%2b'/'%2b \
convert(varchar,isnull(convert(varchar,OR_Email),'NULL'))%2b'/'%2bconvert(varchar,isnull(c \
onvert(varchar,OR_Phone1),'NULL'))%20from%20Orders%20where%20OR_OrderID=47699)--sp_passwo \
rd HTTP/1.1
```



Response Includes Customer Data

Response Details

HTTP/1.1 500 Internal Server Error

Content-Length: 573

Content-Type: text/html

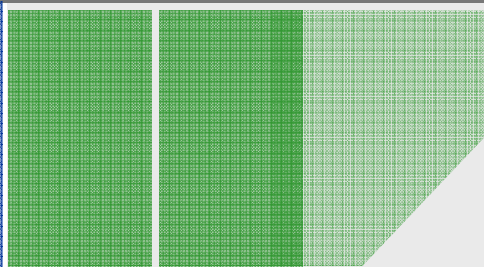
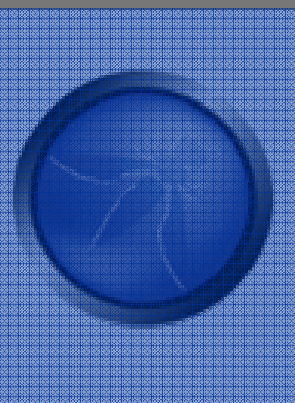
Cache-control: private

Connection: close

```
<font face="Arial" size=2>
<p>Microsoft OLE DB Provider for ODBC Drivers</font> <font face="Arial" size=2>e \
rror '80040e07'</font>
<p>
<font face="Arial" size=2>[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax \
error converting the varchar value 'Feb 13 2007 12:00AM/47699/John/Doe/128 Da \
niel Someplace Dr /City/06354/DC/US/John C Doe Jr/ /k&#151;Utdw&#136;i&#132;&#1 \
41;&#133;qzzv/02/2009/4792/jdoe@email.net/888.555.7578' to a column of data t \
ype int.</font>
<p>
<font face="Arial" size=2>/cart/loginexecute.asp</font><font face="Arial" size=2 \
```

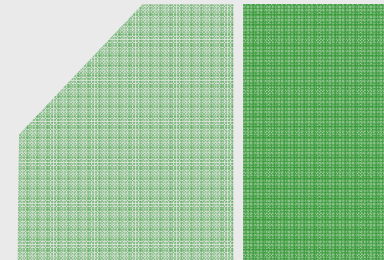
Positive Security Virtual Patch

```
<Location /cart/loginexecute.asp>  
SecRule ARGS:LoginEmail "!^([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.|([a-zA-Z0-9_\-]+)\.)))([a-zA-Z]{2,4}|[0-9]{1,3})$" \\  
"phase:2,capture,log,deny,status:403,msg\\  
:'Email Input Data Violation:\\  
%{TX.0}' "  
</Location>
```

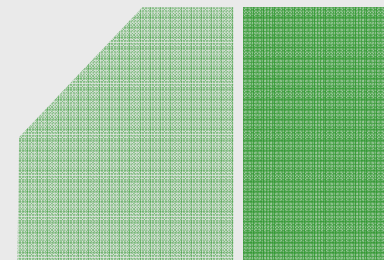



Dynamic Vulnerability Remediation with ModSecurity: *An Incident Response Approach*

Ryan C. Barnett
Ryan.Barnett@breach.com



Complex Vulnerabilities: *Stateful Rules*



What about Complex Vulnerabilities?

- We have been focusing on Atomic-based rules – which is making decisions based on one single transaction.
 - ▶ Command Injection type vulnerabilities are relatively easy to address with small virtual patches.
- Stateful-based rules – which must correlate data from multiple transactions can also be used.
 - ▶ These can include issues such as Brute Force Attacks, Session Hijacking and Business Logic Flaws.
 - ▶ These are more challenging, however, the biggest hurdle is first identifying the problem.
- ModSecurity 2 advanced features can be utilized
 - ▶ Persistent Collections (initcol and setsid)
 - ▶ Set/Update/Decrease arbitrary variables

Tracking Form-based Authentication Failures

- **Goal** - You want to be able to track failed form-based Authentication requests. If they exceed a threshold, then temporarily suspend access and redirect the client to a friendly webpage.
- **Problem** – You need to be able to do the following:
 - ▶ Identify when form-based authentication attempts fail.
 - ▶ Track the failed attempts across multiple requests.
 - ▶ Set a temporary blocking period.
- **Solution** – Use the “`initcol`” action to create a persistent collection based on the client’s IP address and user-agent string, the “`RESPONSE_BODY`” variable location to identify failure text within the html payload and the “`setvar`” action to keep track of the number of failures.

Example Login Failure Message



[Sign Up](#) | [Log In](#) | [Help](#) | [Security Center](#)

Welcome

Send Money

Request Money

Merchant Services

Auction Tools

Member Log-In

Secure Log In 



Your sign in information is not valid. Please try again.

Registered users log in here. Be sure to [protect your password](#).

Email Address:

[Forgot your email address?](#)

Password:

[Forgot your password?](#)

Brute Force Detection Ruleset

Monitoring

- ▶ Capturing the username
- ▶ Login Failures

Protection

- ▶ Brute force detection
- ▶ Scanners and automation detection
- ▶ Misdemeanor scoring

Comparison Operators

State Collection

```
SecAction phase:1,no log,pass,initcol:ip=%{REMOTE_ADDR}_%{HTTP_USER_AGENT}
SecRule IP:SCORE "@ge 20" "phase:1,pass,log,setvar:ip.blocked=1,expirevar:ip.blocked=600"
SecRule IP:SCORE "@gt 100" "phase:1,pass,log,setvar:ip.drop=1,expirevar:ip.drop=100"
```

Drop Action

Inspect HTML

```
SecRule IP:DROP "@eq 1" "phase:1,drop,log,msg:'Brute Force Attack Identified'"
SecRule IP:BLOCKED "@eq 1" "phase:1,deny,log,status:302,redirect:http://www.site.com/"
<Location "login.jsp">
SecRule RESPONSE_BODY "your sign in information is not valid" "phase:4,no log,tlower
setvar:ip.score=+1,expirevar:ip.score=600"
</Location>
```

Rate control

Conclusion

- There is a tremendous need for Virtual Patching:
 - ▶ Vulnerability disclosure is increasing.
 - ▶ Automated exploit code is often released in days.
 - ▶ Organizations have many systems that need to be patched.
 - ▶ Patching processes are often slow.
- Virtual Patching helps to address these issues as it is able to be quickly implemented in a WAF
 - ▶ This provides immediate protection from remote exploitation.
 - ▶ Servers do not have to be taken offline for patching.
 - ▶ There is less chance of service interruption that often happens when traditional patches are installed.
- ModSecurity is an excellent application to implement Virtual Patches - www.modsecurity.org

Questions?

Thank you!

Ryan C. Barnett

Business: Ryan.Barnett@breach.com

Personal: RCBarnett@gmail.com