WAF-aiki

Pentest techniques against a Web Application Firewall
Why WAF-aiki?

“One applies aiki by understanding the rhythm and intent of the [opponent] to find the optimal position and timing to apply a counter-technique.”

https://en.wikipedia.org/wiki/Aikido
Rhythm and Intent

- WAF is the “opponent” in that analogy
  - The obstacle thrown in our way
  - The actor throwing its weight around
- Each WAF has distinct rhythm and intent
- We can’t move it out of the way, but we can incorporate its inertia against it!
Why am I talking about this?

• Varied background – defender, attacker, architect, engineer, coder... seen many sides!
• WAF often more nuanced defensive tier
  – Led me to experience uncertainty
• Forced me to up my mental game
Goals for this talk

• Mind games against the defender
• Discussion of common tools available
• Example of a bespoke tool built for an assessment

→ Emotional Preparedness ←
What does a WAF do?

- Attack Signatures
- Cookie Protection
- Forced Browsing
- Format Validation
- Logging & Alerting
- Information Leakage
- Parameter Tampering
- Data Validation
Assumptions for Mind Games

• “Leading the opponent”
• Most WAF are accessible from the Internet
  – Most WAF see attack traffic 24x7
• WAF need tuning to lower noise
  – Defender may not have availability or technical expertise required to do so
Mind Games: Act Stupid

• Stupid approaches can be used to signal harmlessness to the defender
  – Windows attacks against obviously Linux site
  – Repeat, repeat, repeat, repeat, repeat
  – OTS tool profiles

“Anchoring”
Mind Games: Hide Signal In Noise

- Interleave focused traffic in the middle of a tool scan to lower potential detection.
Mind Games: Disclosure

• Discriminating between WAF and Application rejections allows attacker to map defenses.
• Tuned signature sets reduce false positives, but communicate architecture to attacker.
• Broad signature sets indicate noisier logs and (possibly) less prepared defenders.
Mind Games: Confidence Booster

• Like anti-virus, WAF is an attritional defense
• 10% hit rate on OTS scanners
  – That would be 90% traffic permitted through...
• Remember your opponent is fallible!
Tools: Fingerprinting

• Theory: Identify WAF based on response codes, headers, cookies, bodies to multiple stimuli
Beyond Tools: Fingerprinting

• Fingerprint, but maintain healthy skepticism
• Always be alert for WAF-like behavior
  – “I did this and it failed, I did that and it failed differently.”
• Use the source!
  – wafw00f source especially useful for correlation
  – Work passive fingerprinting into your process
Tools: Evasion

• Theory – Sneak traffic past WAF:
  – Trick WAF into trusting us
  – Mangle traffic such that WAF can’t see malice but Application will decode to malicious effect
    • “Impedance Mismatch”
  – Incomplete signature sets
Tools: Evasion Examples

- Burpsuite “Bypass WAF”
- sqlmap Tamper Scripts
- WAFNinja
Beyond Tools: Evasion

• If you can ID the backend, focus on its quirks, because the WAF layer is relatively generic

• *Protocol-Level Evasion of Web Application Firewalls*, Ivan Ristic, 2012
Bespoke Tool: wafiron.py

• Problem: Super-tight WAF policy effectively thwarted attacker
• Theory: Methodically probe and use responses to enumerate the protective limits
• Benefit: Clearer understanding of limits for both testing and reporting purposes
usage: wafiron.py [-h] -u URL -f FAILURE [-g GOOD] [-d DUMP]
Probe the URL to determine WAF limits. The URL must include the word WAFFLE to mark where adaptive changes should be made.

optional arguments:
  -h, --help            show this help message and exit
  -g GOOD, --good GOOD  Known good value for WAFFLE
  -d DUMP, --dump DUMP  Dump request/response pairs to files in dump directory

required arguments:
  -u URL, --url URL     URL of the target; WAFFLE will be replaced to test
  -f FAILURE, --failure FAILURE
                          String indicating WAF blocked the request
# wafiron Algorithm

# Try different bases until you get a good string
good = ''
if args.good:
    good = args.good
if not good:
    good = tryThisCharset(string.digits)
if not good:
    good = tryThisCharset(string.ascii_lowercase)
if not good:
    good = tryThisCharset(string.ascii_uppercase)

# If you did find a good string, extrapolate based on it
if good:
    print('Enumerating acceptable characters based on good string {}'.format(good))
    tryAllChars(good)
$ ./wafiron.py --url 'https://www.example.com/form?txid=WAFFLE' -f 'support ID'
Found minlen 1, maxlen 20 for charset 0123456789
Enumerating acceptable characters based on good string 39217999632332373486 attempted = 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ+/-_~!@#$%^&*()={}[\]|';":?,.><
acceptable = 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_#&=.,

best guess at breakdown:
Digits (0123456789)
Lowers (abcdefghijklmnopqrstuvwxyz)
Uppers (ABCDEFGHIJKLMNOPQRSTUVWXYZ)
Others (.,_#&=)
$
wafiron Dump Files

REQUEST GET https://www.example.com/form?txid=1260864261285084761!
REQHEAD {'Accept-Encoding': 'gzip, deflate', 'Accept': '*/*', 'Connection': 'keep-alive', 'User-Agent': 'python-requests/2.10.0'}
REQBODY None
RESPONSE 200 OK
RSPHEAD {'Content-Type': 'text/html; charset=utf-8', 'Content-Length': '188', 'Cache-Control': 'no-cache', 'Connection': 'close', 'Pragma': 'no-cache'}
RSPBODY b'<html><head><title>Request Rejected</title></head><body>The requested URL was rejected. Please consult with your administrator.<br><br>Your support ID is: 8799856856743714236</body></html>'
wafiron Advantages

• Automated enumeration of limits saves time
  – Less time wasted manually experimenting
  – Avoid following dead-end paths
• Determined gaps may be usable as findings
• Documentation of accuracy of controls
  – Win for both defender \textit{and} attacker
wafiron Available

https://www.swynwyr.com/wafiron

• Improvements to be made...
  – Extend probing to POST, Headers
  – More sophisticated WAF response specification
Summary

• WAFs are varied and complex, but calculable
• Their nature is to speak to the attacker
  – Listen and heed them!
  – Let your flexibility answer their inertia