Starfield Technologies

Automated Domain Validation Testing Procedures

This document describes the systems and processes that Starfield Technologies uses to verify the integrity of our domain validation process.

The information contained in this document is current as of 20-April, 2017.

Procedures

As part of any change to application software in the production environment, automated verification of domain validation processing is conducted as follows:

1. The deployment process, orchestrated by Jenkins, triggers a deployment of code to our pre-production "staging" environment that is intended to closely resemble our production systems while performing operations with untrusted certificate hierarchies.
2. Once the code has been deployed, the orchestration system invokes a series of automated functional tests using Selenium for browser automation.
3. Engineers on the team view the progress and results from the test suite via a dashboard, and if any test fails they determine the cause of the failure.
4. A change order for the production deployment is created and test results are noted. For any tests that fail we attempt to rerun them locally.
5. If a test passes when rerun locally, we take a screenshot of the successful test result. If a test continues to fail we document the failure reasons/justification. The screenshots and test failure analysis are stored in the code repository
6. The change order receives required approvals and the change is deployed to the live production environment.

Test Names	Verification Test Scenarios
KratosDomainValidationTest	Verify system does not allow requests submitted via control panel for contracted root domains
DomainAGDAEHelperTest DomainDAEHelperTest DomainControlTokenHelperTest	Verify system only sends emails for pending domain ownership approval
KratosBlacklistDomainRequestTest KratosDomainValidationTest	Verify internal, public IPs, ipv6, or blacklisted domains can not be submitted
KratosDomainValidationTest	Verify banned TLDs can not be submitted
DomainLogServiceTest	Verify token generation is unique across all approval types
DomainLogServiceTest	Verify token value is 112-bit entropy

KratosDomainValidationTest	Verify system enforces valid domain format, including not allowing internal names
KratosDAEUniqueTokensTest	BR 3.2.2.4.4: Verify that the verification email is only sent to the 5 approved email addresses for a domain
KratosDAEUniqueTokensTest	BR 3.2.2.4.4: Verify unique token is created for each email address
KratosDAEWhoisContactsTest	BR 3.2.2.4.2: Verify that for a subdomain (or wildcard) request emails are sent to the registrant, Technical and Administrative Contacts associated with the base domain
KratosDAEWhoisContactsTest	BR 3.2.2.4.2: Verify that for a subdomain (or wildcard) request emails are NOT sent to the Billing Contact associated with the base domain
KratosDaeNotApprovedTest	BR 3.2.2.4.2: Verify that clicking on the link in the email does not alone approve the domain (user must click approve on the web page)
KratosDaeAuditNoteTest	BR 3.2.2.4.2: Verify that a domain approved via DAE/AGDAE will have an audit note indicating approval method and approval date
KratosDAEWhoisCheckForExpiredTokensTest	BR 3.2.2.4.2: Verify that when a token is expired we requery whois and use those results to send a new token to all email addresses
KratosAgdaeByTldTest	Verify that the base domain in the public suffix list is recognized as a base domain
KratosDaeOptOutTokenExpirationTest	BR 3.2.2.4.2: Verify when customer had opted out of email based validation, emails are not sent
KratosDaeDenyTest	BR 3.2.2.4.2: Verify that clicking deny on the verification page causes the verification to fail
ArtemisResendDAETest	BR 3.2.2.4.2: Verify that a new random value is generated if any email address changes
KratosDAETokensByBaseDomainTest	BR 3.2.2.4.2: Verify system generates same token when domains have common root domain
KratosDAEExpiredTokenNoWhoisTest	BR 3.2.2.4.2: Verify that when a token is expired and no whois results are available then a new token is not generated

KratosDAETokensByBaseDomainTest	BR 3.2.2.4.2: Verify unique token is created for each distinct root domain in SAN
KratosDCEExpiredTokenTest	BR 3.2.2.4.7: Verify that expired tokens fail verification and user is presented with expired token message on screen; a new token is generated
KratosDZCInvalidTokenSetupTest	BR 3.2.2.4.7: Verify that a domain can NOT be validated via BR 3.2.2.4.7 if the text record has been added at dzc.primarydomainName
KratosDZCSubdomainTest	BR 3.2.2.4.7: Verify that a request for a subdomain (www or other) can be verified at the base domain
KratosDZCInvalidTokenSetupTest	BR 3.2.2.4.7: Verify that a domain can NOT be validated via BR 3.2.2.4.7 if the text record has been added at www.primarydomainName
KratosDZCWildcardTest	BR 3.2.2.4.7: Verify that a wildcard request can be validated using BR 3.2.2.4.7
KratosDZCTest	BR 3.2.2.4.7: Verify that verification fails if code is placed in a CNAME record
KratosDZCSubdomainTest	BR 3.2.2.4.7: Verify that the base domain defined in the public suffix list is honored
KratosDZCTest	BR 3.2.2.4.7: Verify that a domain approved via BR 3.2.2.4.7 will have an audit note indicating approval method and approval date
KratosDCETokensByRequestTest	BR 3.2.2.4.7: Verify unique token is created for each certificate request
KratosNotShopperOwnedDomainTest	BR 3.2.2.4.1: verify it fails if domain is not owned by applicant requesting the cert
KratosShopperOwnedDomainTest	BR 3.2.2.4.1: Verify that the domain is approved automatically by the system when it's owned by the applicant submitting the request
KratosWSCTest	BR 3.2.2.4.6: Verify that a domain approved via BR 3.2.2.4.6 will have an audit note indicating approval method and approval date
KratosDCETokensByRequestTest	BR 3.2.2.4.6: Verify unique token is created for each certificate request
KratosWSCSubdomainTest	BR 3.2.2.4.6: Verify that a request for a subdomain or www subdomain can be verified at the "FQDN" (minus the WWW)

KratosWSCInvalidForWildcardsTest	BR 3.2.2.4.6: Verify that a request for a wildcard domain can not be verified using this method
KratosWscUserCertificateTest	BR 3.2.2.4.6: Verify that verification over HTTPS requires a valid trusted certificate (correct domain name, not expired, not self-signed, etc.)
KratosWSCTest	BR 3.2.2.4.6: Verify that domain verification fails if token is placed at www.example.com or www.FQDN
KratosDCEExpiredTokenTest	BR 3.2.2.4.6: Verify that expired tokens fail verification and user is presented with expired token message on screen; a new token is generated
KratosWSCSubdomainTest	BR 3.2.2.4.6: Verify that the base domain defined in the public suffix list is recognized as a base domain
DomainControlHelperTest	BR 3.2.2.4.6: verify that the file can be found/verified at http or https
DomainControlHelperTest	BR 3.2.2.4.6: verify that the file can be found/verified at ports 80 or 443
KratosWSCTest KratosWSCSubdomainTest	BR 3.2.2.4.6: verify that the system looks for a file named godaddy.html or starfield.html for GD
KratosWSCFollowRedirectTest	BR 3.2.2.4.6: verify that the system does not follow redirects
KratosWSCTest	BR 3.2.2.4.6: Verify that domain verification fails if the token in the file is not the assigned token
KratosWSCTest	BR 3.2.2.4.6: Verify that domain verification fails if an invalid file name is used
KratosWSCTest	BR 3.2.2.4.6: Verify that the verification file must be in the /.well-known/pki-validation/ directory of the "FQDN with WWW stripped"
KratosWSCTest	BR 3.2.2.4.6: verify that a random value placed in a file with the name of the random value fails validation