|  |  |  |
| --- | --- | --- |
| WHStore:watsonhall:groups:owasp:OWASP_Image_Toolbox:owasp_logo_122106.eps  **OWASP Cornucopia is a mechanism to assist software development teams identify security requirements in Agile, conventional and formal development processes.**  Author  Author  Colin Watson  Project Leaders  Colin Watson and Darío De Filippis  Reviewers  Tom Brennan, Johanna Curiel and Timo Goosen  Acknowledgments  Project Leaders  Reviewers  ${Common\_T00050}  ${Common\_T00060}  ${Common\_T00070}  ${Common\_T00080}  ${Common\_T00090}  Acknowledgments | |  |
| Introduction  ${Common\_T002210}  Although the idea had been waiting for enough time to progress it, the final motivation came when SAFECode published its Practical Security Stories and Security Tasks for Agile Development Environments in July 2012.  The Microsoft SDL team had already published its super Elevation of Privilege: The Threat Modeling Game (EoP) but that did not seem to address the most appropriate kind of issues that web application development teams mostly have to address.  Although the idea had been waiting for enough time to progress it, the final motivation came when SAFECode published its Practical Security Stories and Security Tasks for Agile Development Environments in July 2012.  Cornucopia Ecommerce Website Edition is based the concepts and game ideas in EoP, but those have been modified to be more relevant to the types of issues ecommerce website developers encounter.  It attempts to introduce threat-modelling ideas into development teams that use Agile methodologies, or are more focused on web application weaknesses than other types of software vulnerabilities or are not familiar with STRIDE and DREAD.  Cornucopia Ecommerce Website Edition is referenced as an information resource in the PCI Security Standard Council’s Information Supplement PCI DSS E-commerce Guidelines, v2, January 2013.  The card deck (pack)  These provided five suits, and a sixth called “Cornucopia” was created for everything else:  These provided five suits, and a sixth called “Cornucopia” was created for everything else:   * • Data validation and encoding (VE) * • Authentication (AT) * • Session Management (SM) * • Authorization (AZ) * • Cryptography (CR) * • Cornucopia (C)   Smilar to poker-playing cards, each suit contains 13 cards (Ace, 2-10, Jack, Queen and King) but, unlike EoP, there are also two Joker cards.  The content was mainly drawn from the SCP.  Mappings  The other driver for Cornucopia is to link the attacks with requirements and verification techniques.  An initial aim had been to reference CWE weakness IDs, but these proved too numerous, and instead it was decided to map each card to CAPEC software attack pattern IDs which themselves are mapped to CWEs, so the desired result is achieved.  Each card is also mapped to the 36 primary security stories in the SAFECode document, as well as to the OWASP SCP v2, ASVS v3.0.1 and AppSensor (application attack detection and response) to help teams create their own security-related stories for use in Agile processes. |  | Game strategy  Apart from the content differences, the game rules are virtually identical to those for EoP.  Printing the cards  Check the Cornucopia project page for how to obtain pre-printed decks on glossy card.  The cards can be printed from this document in black & white but are more effective in color.  The cards in the later pages of this document have been laid out to fit on one type of pre-scored business A4 card sheets.  This appeared to be the quickest way to initially provide to create playing cards quickly.  Avery product codes C32015 and C32030 have been tested successfully, but any 10 up 85mm x 54 mm cards on A4 paper should work with a little adjustment.  Other stationery suppliers like Ryman and Sigel produce similar sheets.  These card sheets are not inexpensive, so care should be taken in deciding what to print and using what media and printer type.  The cards can of course just be printed on any size of paper or card and then cut-up manually, or a commercial printer would be able to print larger volumes and cut the cards to size.  The cut lines are shown on the penultimate page of this document, but Avery also produce a landscape A4 template (A-0017-01\_L.doc) that can be used as a guide.  Printing and cutting up can take an hour or so, and using a faster printer helps.  Try to print add higher quality to increase legibility.  An optional card back design (in OWASP tartan) has been provided as the last page of this document.  There is no special alignment needed.  Dual-sided printing needs special care taken.  You could customize the card faces or the backs for your own organization’s preferences.  Customization  After you have used Cornucopia a few times, you may feel that some cards are less relevant to your applications, or the threats are different for your organization.  Edit this document yourself to make the cards more suitable for your teams, or create new decks completely.  Instructions  The text on each card describes an attack, but the attacker is given a name, which are unique across all the cards.  The name can represent a computer system (e.g. the database, the file system, another application, a related service, a botnet), an individual person (e.g. a citizen, a customer, a client, an employee, a criminal, a spy), or even a group of people (e.g. a competitive organization, activists with a common cause).  The attacker might be remote in some other device/location, or local/internal with access to the same device, host or network as the application is running on.   * The attacker is always named at the start of each description<https://lists.owasp.org/mailman/listinfo/owasp_cornucopia> * An example is: William has control over the generation of session identifiers.<https://www.owasp.org/index.php/OWASP_Cornucopia>   This means the attacker (William) can create new session identifiers that the application accepts.  The attacks were primarily drawn from the security requirements listed in the SCP, v2 but then supplemented with verification objectives from the OWASP “Application Security Verification Standard for Web Applications”, the security focused stories in SAFECode’s “Practical Security Stories and Security Tasks for Agile Development Environments”, and finally a review of the cards in EOP. |

|  |  |  |
| --- | --- | --- |
| Lookups between the attacks and five resources are provided on most cards:  • Requirements in “Secure Coding Practices (SCP) - Quick Reference Guide”, v2, OWASP, November 2010  https://www.owasp.org/index.php/File:OWASP\_SCP\_Quick\_Reference\_Guide\_v2.pdf  • Verification IDs in “Application Security Verification Standard (ASVS) for Web Applications”, OWASP, v3.0.1, 2016 (excluding sections 18 and 19)  https://www.owasp.org/images/3/33/OWASP\_Application\_Security\_Verification\_Standard\_3.0.1.pdf  • Attack detection points IDs in “AppSensor”, OWASP, August 2010-2015  https://www.owasp.org/index.php/AppSensor\_DetectionPoints  • IDs in “Common Attack Pattern Enumeration and Classification (CAPEC)”, v2.8, Mitre Corporation, November 2015  http://capec.mitre.org/data/archive/capec\_v2.8.zip  <https://www.owasp.org/index.php/Cornucopia_-_Ecommerce_Website_Edition_-_Wiki_Deck>  • Security-focused stories in 'Practical Security Stories and Security Tasks for Agile Development Environments', SAFECode, July 2012   * http://www.safecode.org/publications/SAFECode\_Agile\_Dev\_Security0712.pdf<https://www.owasp.org/index.php/File:OWASP_SCP_Quick_Reference_Guide_v2.pdf> * A look-up means the attack is included within the referenced item, but does not necessarily encompass the whole of its intent.<https://www.owasp.org/images/3/33/OWASP_Application_Security_Verification_Standard_3.0.1.pdf> * For structured data like CAPEC, the most specific reference is provided but sometimes a cross-reference is provided that also has more specific (child) examples.<https://www.owasp.org/index.php/AppSensor_DetectionPoints> * There are no lookups on the six Aces and two Jokers.   <http://capec.mitre.org/data/archive/capec_v2.8.zip>   * Instead these cards have some general tips in italicized text.   <http://www.safecode.org/publications/SAFECode_Agile_Dev_Security0712.pdf>  It is possible to play Cornucopia in many different ways.  Here is one way, demonstrated online in a video at https://youtu.be/i5Y0akWj31k , which uses the new (May 2015) score/record sheet at https://www.owasp.org/index.php/File:Cornucopia-scoresheet.pdf  ${Common\_T01270}  ${Common\_T01280}  ${Common\_T01290}  A - Preparations  <https://youtu.be/i5Y0akWj31k><https://www.owasp.org/index.php/File:Cornucopia-scoresheet.pdf> |  | ${Common\_01400}  ${Common\_01410}  ${Common\_01420}  ${Common\_01430}  ${Common\_01440}  ${Common\_01450}  ${Common\_01500}  ${Common\_01510}  ${Common\_01520}  ${Common\_01530}  ${Common\_01540}  ${Common\_01550}  ${Common\_01560}  ${Common\_01570}  ${Common\_01580}  ${Common\_01590}  ${Common\_01600}  ${Common\_01610}  D - Closure  D1. Review all the applicable threats and the matching security requirements  D2. Create user stories, specifications and test cases as required for your development methodology.  ${Common\_T01730}  ${Common\_T01740}  Alternative game rules  If you are new to the game, remove the Aces and two Joker cards to begin with.  Add the Joker cards back in once people become more familiar with the process. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Consider allowing extra points for especially good contributions.  You can even play by yourself.  ${Common\_T01920}  Involving more people will be beneficial though.  In Microsoft's EoP guidance, they recommend cheating as a good game strategy.  ${Common\_T01950}  ${Common\_T01960}  ${Common\_T01970}  ${Common\_T01980}  ${Common\_T01990}  Development framework-specific modified card decks  At the end of 2012, the OWASP Framework Security Matrix was published which documents built in security controls in some commonly used languages and frameworks for web and mobile application development.  With certain provisos it is useful to consider how using these controls can simplify the identification of additional requirements – provided of course the controls are included, enabled and configured correctly.  Involving more people will be beneficial though.  In Microsoft's EoP guidance, they recommend cheating as a good game strategy.  Internal coding standards and libraries  Add your own list of excluded cards based on your organisation’s coding standards (provided they are confirmed by appropriate verification steps in the development lifecycle).  Your coding standards and libraries  Data validation and encoding  [your list] |  | [your list]  Cryptography   |  |  |  | | --- | --- | --- | | ${Common\_T02220} | | | | ${Common\_T02230}  *${Common\_T02240}*  ${Common\_T02250}  *${Common\_T02260}* | ${Common\_T02270}  *${Common\_T02280}*  ${Common\_T02290}  *${Common\_T02300}* | ${Common\_T02310}  *${Common\_T02320}*  ${Common\_T02330}  *${Common\_T02340}* |   [compliance list]  Cryptography   |  |  |  | | --- | --- | --- | | ${Common\_T02420} | | | | ${Common\_T02430}  *${Common\_T02440}*  ${Common\_T02450}  *${Common\_T02460}* | ${Common\_T02470}  *${Common\_T02480}*  ${Common\_T02490}  *${Common\_T02500}* | ${Common\_T02510}  *${Common\_T02520}*  ${Common\_T02530}  *${Common\_T02540}* | |

|  |  |  |
| --- | --- | --- |
| *${Common\_02600}*  *${Common\_02610}*  *${Common\_02620}*  *${Common\_02630}*  *In v1.20, the name on VE-10 changed to reflect the project’s new co-leader - this card is also the only one with two names in the attack.*  *4. Why aren’t there any images on the card faces?*  *There is quite a lot of text on the cards, and the cross-referencing takes up space too.*  *But it would be great to have additional design elements included.*  *Any volunteer*  *5. Are the attacks ranked by the number on the card?*  *Only approximately.*  *The risk will be application and organisation dependent, due to varying security and compliance requirements, so your own severity rating may place the cards in some other order than the numbers on the cards.*  *6. How long does it take to play a round of cards using the full deck?*  *This depends upon the amount of discussion and how familiar the players are with application security concepts.*  *But perhaps allow 1.5 to 2.0 hours for 4-6 people.*  *7. What sort of people should play the game?*  *Always try to have a mix of roles who can contribute alternative perspectives.*  *But include someone who has a reasonable knowledge of application vulnerability terminology.*  *Otherwise try to include a mix of architects, developers, testers and a relevant project manager or business owner.*  *8. Who should take notes and record scores?*  *It is better if that someone else, not playing the game, takes notes about the requirements identified and issues discussed.*  *This could be used as training for a more junior developer, or performed by the project manager.*  *Some organisations have made a recording to review afterwards when the requirements are written up more formally.*  *9. Should we always use the full deck of cards?*  *No.*  *A smaller deck is quicker to play.*  *Start your first game with only enough cards for two or three rounds.*  *Always consider removing cards that are not appropriate at all of the target application or function being reviewed.*  *For the first few times people play the game it is also usually better to remove the Aces and the two Jokers.*  *It is also usual to play the game without any trumps suit until people are more familiar with the idea.*  *10. What should players do when they have an Ace card that says “invented a new X attack”?*  *The player can make up any attack they think is valid, but must match the suit of the card e.g. data validation and encoding).*  *With players new to the game, it can be better to remove these to begin with (see also FAQ 9).*  *11. I don’t understand what the attack means on each card - is there more detailed information?*  *Yes, the online Wiki Deck at was created to help players understand the attacks.*  *See*  *https://www.owasp.org/index.php/Cornucopia\_-\_Ecommerce\_Website\_Edition\_-\_Wiki\_Deck*  *12. My company wants to print its own version of OWASP Cornucopia - what license do we need to refer to?*  *Please refer to the full answer to this question on the project’s web pages at*  *https://www.owasp.org/index.php/OWASP\_Cornucopia - tab=FAQs*  *https://www.owasp.org/index.php/Cornucopia\_-\_Ecommerce\_Website\_Edition\_-\_Wiki\_Deck*  *${Common\_T03010}*  *${Common\_T03020}*  *${Common\_T03030}*  *${Common\_T03040}*  <https://www.owasp.org/index.php/Cornucopia_-_Ecommerce_Website_Edition_-_Wiki_Deck>  *${Common\_T03050}*  *${Common\_T03060}*  *${Common\_T03070}*  [https://www.owasp.org/index.php/OWASP\_Cornucopia - tab=FAQs](https://www.owasp.org/index.php/OWASP_Cornucopia#tab=FAQs) |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Data validation & encoding** | **A** | **Data validation & encoding** |  | **Data validation & encoding** | **2** | **Data validation & encoding** | **3** |
| You have invented a new attack against Data Validation and Encoding | (No Card) | Brian can gather information about the underlying configurations, schemas, logic, code, software, services and infrastructure due to the content of error messages, or poor configuration, or the presence of default installation files or old, test, backup or copies of resources, or exposure of source code | Robert can input malicious data because the allowed protocol format is not being checked, or duplicates are accepted, or the structure is not being verified, or the individual data elements are not being validated for format, type, range, length and a whitelist of allowed characters or formats |
| *Read more about this topic in OWASP's free Cheat Sheets on Input Validation, XSS Prevention, DOM-based XSS Prevention, SQL Injection Prevention, and Query Parameterization* |  | |  | | --- | | ${VE\_VE2\_SCP}  ${Common\_T0020} | | ${Common\_T0030}  ${Common\_T0040} | | ${Common\_T0050}  ${Common\_T0060} | | ${Common\_T0070}  ${Common\_T0080} | | ${Common\_T0090}  ${Common\_T0100} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T00110}  ${Common\_T00120} | | ${Common\_T00130}  ${Common\_T00140} | | ${Common\_T00150}  ${Common\_T00160} | | ${Common\_T00170}  ${Common\_T00180} | | ${Common\_T00190}  ${Common\_T00200} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |
| **Data validation & encoding** | **4** | **Data validation & encoding** | **5** | **Data validation & encoding** | **6** | **Data validation & encoding** | **7** |
| Dave can input malicious field names or data because it is not being checked within the context of the current user and process | Jee can bypass the centralized encoding routines since they are not being used everywhere, or the wrong encodings are being used | Jason can bypass the centralized validation routines since they are not being used on all inputs | Jan can craft special payloads to foil input validation because the character set is not specified/enforced, or the data is encoded multiple times, or the data is not fully converted into the same format the application uses (e.g. canonicalization) before being validated, or variables are not strongly typed |
| |  | | --- | | ${Common\_T00210}  ${Common\_T00220} | | ${Common\_T00230}  ${Common\_T00240} | | ${Common\_T00250}  ${Common\_T00260} | | ${Common\_T00270}  ${Common\_T00280} | | ${Common\_T00290}  ${Common\_T00300} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T00310}  ${Common\_T00320} | | ${Common\_T00330}  ${Common\_T00340} | | ${Common\_T00350}  ${Common\_T00360} | | ${Common\_T00370}  ${Common\_T00380} | | ${Common\_T00390}  ${Common\_T00400} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T00410}  ${Common\_T00420} | | ${Common\_T00430}  ${Common\_T00440} | | ${Common\_T00450}  ${Common\_T00460} | | ${Common\_T00470}  ${Common\_T00480} | | ${Common\_T00490}  ${Common\_T00500} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T00510}  ${Common\_T00520} | | ${Common\_T00530}  ${Common\_T00540} | | ${Common\_T00550}  ${Common\_T00560} | | ${Common\_T00570}  ${Common\_T00580} | | ${Common\_T00590}  ${Common\_T00600} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Data validation & encoding** | **8** | **Data validation & encoding** | **9** | **Data validation & encoding** | **10** | **Data validation & encoding** | **J** |
| Sarah can bypass the centralized sanitization routines since they are not being used comprehensively | Shamun can bypass input validation or output validation checks because validation failures are not rejected and/or sanitized | Darío can exploit the trust the application places in a source of data (e.g. user-definable data, manipulation of locally stored data, alteration to state data on a client device, lack of verification of identity during data validation such as Darío can pretend to be Colin) | Dennis has control over input validation, output validation or output encoding code or routines so they can be bypassed |
| |  | | --- | | ${Common\_T00610}  ${Common\_T00620} | | ${Common\_T00630}  ${Common\_T00640} | | ${Common\_T00650}  ${Common\_T00660} | | ${Common\_T00670}  ${Common\_T00680} | | ${Common\_T00690}  ${Common\_T00700} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T00710}  ${Common\_T00720} | | ${Common\_T00730}  ${Common\_T00740} | | ${Common\_T00750}  ${Common\_T00760} | | ${Common\_T00770}  ${Common\_T00780} | | ${Common\_T00790}  ${Common\_T00800} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T00810}  ${Common\_T00820} | | ${Common\_T00830}  ${Common\_T00840} | | ${Common\_T00850}  ${Common\_T00860} | | ${Common\_T00870}  ${Common\_T00880} | | ${Common\_T00890}  ${Common\_T00900} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T00910}  ${Common\_T00920} | | ${Common\_T00930}  ${Common\_T00940} | | ${Common\_T00950}  ${Common\_T00960} | | ${Common\_T00970}  ${Common\_T00980} | | ${Common\_T00990}  ${Common\_T01000} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |
| **Data validation & encoding** | **Q** | **Data validation & encoding** | **K** |  |  |  |  |
| Geoff can inject data into a client or device side interpreter because a parameterised interface is not being used, or has not been implemented correctly, or the data has not been encoded correctly for the context, or there is no restrictive policy on code or data includes | Gabe can inject data into an server-side interpreter (e.g. SQL, OS commands, Xpath, Server JavaScript, SMTP) because a strongly typed parameterised interface is not being used or has not been implemented correctly | (No Card) | (No Card) |
| |  | | --- | | ${Common\_T01010}  ${Common\_T01020} | | ${Common\_T01030}  ${Common\_T01040} | | ${Common\_T01050}  ${Common\_T01060} | | ${Common\_T01070}  ${Common\_T01080} | | ${Common\_T01090}  ${Common\_T01100} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T01110}  ${Common\_T01120} | | ${Common\_T01130}  ${Common\_T01140} | | ${Common\_T01150}  ${Common\_T01160} | | ${Common\_T01170}  ${Common\_T01180} | | ${Common\_T01190}  ${Common\_T01200} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Authentication** | **A** | **Authentication** |  | **Authentication** | **2** | **Authentication** | **3** |
| You have invented a new attack against Authentication | (No Card) | James can undertake authentication functions without the real user ever being aware this has occurred (e.g. attempt to log in, log in with stolen credentials, reset the password) | Muhammad can obtain a user's password or other secrets such as security questions, by observation during entry, or from a local cache, or from memory, or in transit, or by reading it from some unprotected location, or because it is widely known, or because it never expires, or because the user cannot change her own password |
| *Read more about this topic in OWASP's free Authentication Cheat Sheet* |  | |  | | --- | | ${Common\_T01210}  ${Common\_T01220} | | ${Common\_T01230}  ${Common\_T01240} | | ${Common\_T01250}  ${Common\_T01260} | | ${Common\_T01270}  ${Common\_T01280} | | ${Common\_T01290}  ${Common\_T01300} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T01310}  ${Common\_T01320} | | ${Common\_T01330}  ${Common\_T01340} | | ${Common\_T01350}  ${Common\_T01360} | | ${Common\_T01370}  ${Common\_T01380} | | ${Common\_T01390}  ${Common\_T01400} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |
| **Authentication** | **4** | **Authentication** | **5** | **Authentication** | **6** | **Authentication** | **7** |
| Sebastien can easily identify user names or can enumerate them | Javier can use default, test or easily guessable credentials to authenticate, or can use an old account or an account not necessary for the application | Sven can reuse a temporary password because the user does not have to change it on first use, or it has too long or no expiry, or it does not use an out-of-band delivery method (e.g. post, mobile app, SMS) | Cecilia can use brute force and dictionary attacks against one or many accounts without limit, or these attacks are simplified due to insufficient complexity, length, expiration and re-use requirements for passwords |
| |  | | --- | | ${Common\_T01410}  ${Common\_T01420} | | ${Common\_T01430}  ${Common\_T01440} | | ${Common\_T01450}  ${Common\_T01460} | | ${Common\_T01470}  ${Common\_T01480} | | ${Common\_T01490}  ${Common\_T01500} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T01510}  ${Common\_T01520} | | ${Common\_T01530}  ${Common\_T01540} | | ${Common\_T01550}  ${Common\_T01560} | | ${Common\_T01570}  ${Common\_T01580} | | ${Common\_T01590}  ${Common\_T01600} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T01510}  ${Common\_T01520} | | ${Common\_T01530}  ${Common\_T01540} | | ${Common\_T01550}  ${Common\_T01560} | | ${Common\_T01570}  ${Common\_T01580} | | ${Common\_T01590}  ${Common\_T01600} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T01610}  ${Common\_T01620} | | ${Common\_T01630}  ${Common\_T01640} | | ${Common\_T01650}  ${Common\_T01660} | | ${Common\_T01670}  ${Common\_T01680} | | ${Common\_T01690}  ${Common\_T01700} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Authentication** | **8** | **Authentication** | **9** | **Authentication** | **10** | **Authentication** | **J** |
| Kate can bypass authentication because it does not fail secure (i.e. it defaults to allowing unauthenticated access) | Claudia can undertake more critical functions because authentication requirements are too weak (e.g. do not use strong authentication such as two factor), or there is no requirement to re-authenticate for these | Pravin can bypass authentication controls because a centralized standard, tested, proven and approved authentication module/framework/service, separate to the resource being requested, is not being used | Mark can access resources or services because there is no authentication requirement, or it was mistakenly assumed authentication would be undertaken by some other system or performed in some previous action |
| |  | | --- | | ${Common\_T01710}  ${Common\_T01720} | | ${Common\_T01730}  ${Common\_T01740} | | ${Common\_T01750}  ${Common\_T01760} | | ${Common\_T01770}  ${Common\_T01780} | | ${Common\_T01790}  ${Common\_T01800} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T01810}  ${Common\_T01820} | | ${Common\_T01830}  ${Common\_T01840} | | ${Common\_T01850}  ${Common\_T01860} | | ${Common\_T01870}  ${Common\_T01880} | | ${Common\_T01890}  ${Common\_T01900} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T01910}  ${Common\_T01920} | | ${Common\_T01930}  ${Common\_T01940} | | ${Common\_T01950}  ${Common\_T01960} | | ${Common\_T01970}  ${Common\_T01980} | | ${Common\_T01990}  ${Common\_T02000} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T02010}  ${Common\_T02020} | | ${Common\_T02030}  ${Common\_T02040} | | ${Common\_T02050}  ${Common\_T02060} | | ${Common\_T02070}  ${Common\_T02080} | | ${Common\_T02090}  ${Common\_T02100} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |
| **Authentication** | **Q** | **Authentication** | **K** |  |  |  |  |
| Jaime can bypass authentication because it is not enforced with equal rigor for all types of authentication functionality (e.g. register, password change, password recovery, log out, administration) or across all versions/channels (e.g. mobile website, mobile app, full website, API, call centre) | Olga can influence or alter authentication code/routines so they can be bypassed | (No Card) | (No Card) |
| |  | | --- | | ${Common\_T02110}  ${Common\_T02120} | | ${Common\_T02130}  ${Common\_T02140} | | ${Common\_T02150}  ${Common\_T02160} | | ${Common\_T02170}  ${Common\_T02180} | | ${Common\_T02190}  ${Common\_T02200} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T02210}  ${Common\_T02220} | | ${Common\_T02230}  ${Common\_T02240} | | ${Common\_T02250}  ${Common\_T02260} | | ${Common\_T02270}  ${Common\_T02280} | | ${Common\_T02290}  ${Common\_T02300} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Session management** | **A** | **Session management** |  | **Session management** | **2** | **Session management** | **3** |
| You have invented a new attack against Session Management | (No Card) | William has control over the generation of session identifiers | Ryan can use a single account in parallel since concurrent sessions are allowed |
| *Read more about this topic in OWASP's free Cheat Sheets on Session Management, and Cross Site Request Forgery (CSRF) Prevention* |  | |  | | --- | | ${Common\_T02310}  ${Common\_T02320} | | ${Common\_T02330}  ${Common\_T02340} | | ${Common\_T02350}  ${Common\_T02360} | | ${Common\_T02370}  ${Common\_T02380} | | ${Common\_T02390}  ${Common\_T02400} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T02410}  ${Common\_T02420} | | ${Common\_T02430}  ${Common\_T02440} | | ${Common\_T02450}  ${Common\_T02460} | | ${Common\_T02470}  ${Common\_T02480} | | ${Common\_T02490}  ${Common\_T02500} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |
| **Session management** | **4** | **Session management** | **5** | **Session management** | **6** | **Session management** | **7** |
| Alison can set session identification cookies on another web application because the domain and path are not restricted sufficiently | John can predict or guess session identifiers because they are not changed when the user's role alters (e.g. pre and post authentication) and when switching between non-encrypted and encrypted communications, or are not sufficiently long and random, or are not changed periodically | Gary can take over a user's session because there is a long or no inactivity timeout, or a long or no overall session time limit, or the same session can be used from more than one device/location | Casey can utilize Adam's session after he has finished, because there is no log out function, or he cannot easily log out, or log out does not properly terminate the session |
| |  | | --- | | ${Common\_T02510}  ${Common\_T02520} | | ${Common\_T02530}  ${Common\_T02540} | | ${Common\_T02550}  ${Common\_T02560} | | ${Common\_T02570}  ${Common\_T02580} | | ${Common\_T02590}  ${Common\_T02600} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T02610}  ${Common\_T02620} | | ${Common\_T02630}  ${Common\_T02640} | | ${Common\_T02650}  ${Common\_T02660} | | ${Common\_T02670}  ${Common\_T02680} | | ${Common\_T02690}  ${Common\_T02700} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T02710}  ${Common\_T02720} | | ${Common\_T02730}  ${Common\_T02740} | | ${Common\_T02750}  ${Common\_T02760} | | ${Common\_T02770}  ${Common\_T02780} | | ${Common\_T02790}  ${Common\_T02800} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T02810}  ${Common\_T02820} | | ${Common\_T02830}  ${Common\_T02840} | | ${Common\_T02850}  ${Common\_T02860} | | ${Common\_T02870}  ${Common\_T02880} | | ${Common\_T02890}  ${Common\_T02900} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Session management** | **8** | **Session management** | **9** | **Session management** | **10** | **Session management** | **J** |
| Matt can abuse long sessions because the application does not require periodic re-authentication to check if privileges have changed | Ivan can steal session identifiers because they are sent over insecure channels, or are logged, or are revealed in error messages, or are included in URLs, or are accessible un-necessarily by code which the attacker can influence or alter | Marce can forge requests because per-session, or per-request for more critical actions, strong random tokens (i.e. anti-CSRF tokens) or similar are not being used for actions that change state | Jeff can resend an identical repeat interaction (e.g. HTTP request, signal, button press) and it is accepted, not rejected |
| |  | | --- | | ${Common\_T02910}  ${Common\_T02920} | | ${Common\_T02930}  ${Common\_T02940} | | ${Common\_T02950}  ${Common\_T02960} | | ${Common\_T02970}  ${Common\_T02980} | | ${Common\_T02990}  ${Common\_T03000} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T03010}  ${Common\_T03020} | | ${Common\_T03030}  ${Common\_T03040} | | ${Common\_T03050}  ${Common\_T03060} | | ${Common\_T03070}  ${Common\_T03080} | | ${Common\_T03090}  ${Common\_T03100} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T03110}  ${Common\_T03120} | | ${Common\_T03130}  ${Common\_T03140} | | ${Common\_T03150}  ${Common\_T03160} | | ${Common\_T03170}  ${Common\_T03180} | | ${Common\_T03190}  ${Common\_T03200} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T03210}  ${Common\_T03220} | | ${Common\_T03230}  ${Common\_T03240} | | ${Common\_T03250}  ${Common\_T03260} | | ${Common\_T03270}  ${Common\_T03280} | | ${Common\_T03290}  ${Common\_T03300} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |
| **Session management** | **Q** | **Session management** | **K** |  |  |  |  |
| Salim can bypass session management because it is not applied comprehensively and consistently across the application | Peter can bypass the session management controls because they have been self-built and/or are weak, instead of using a standard framework or approved tested module | (No Card) | (No Card) |
| |  | | --- | | ${Common\_T03310}  ${Common\_T03320} | | ${Common\_T03330}  ${Common\_T03340} | | ${Common\_T03350}  ${Common\_T03360} | | ${Common\_T03370}  ${Common\_T03380} | | ${Common\_T03390}  ${Common\_T03400} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | ${Common\_T03410}  ${Common\_T03420} | | ${Common\_T03430}  ${Common\_T03340} | | ${Common\_T03350}  ${Common\_T03360} | | ${Common\_T03370}  ${Common\_T03380} | | ${Common\_T03390}  ${Common\_T03400} | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Authorization** | **A** | **Authorization** |  | **Authorization** | **2** | **Authorization** | **3** |
| You have invented a new attack against Authorization | (No Card) | Tim can influence where data is sent or forwarded to | Christian can access information, which they should not have permission to, through another mechanism that does have permission (e.g. search indexer, logger, reporting), or because it is cached, or kept for longer than necessary, or other information leakage |
| *Read more about this topic in OWASP's Development and Testing Guides* |  | |  | | --- | | OWASP SCP  ${AZ\_AZ2\_OWASP\_SCP} | | OWASP ASVS  4.1, 4.16, 16.1 | | OWASP AppSensor  ${AZ\_AZ2\_OWASP\_AppSensor} | | CAPEC  153 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  51, 100, 135, 139, 140, 141, 150 | | OWASP ASVS  4.1, 8.2, 9.1-9.6, 9.11, 16.6, 16.7 | | OWASP AppSensor  - | | CAPEC  69, 213 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |
| **Authorization** | **4** | **Authorization** | **5** | **Authorization** | **6** | **Authorization** | **7** |
| Kelly can bypass authorization controls because they do not fail securely (i.e. they default to allowing access) | Chad can access resources (including services, processes, AJAX, Flash, video, images, documents, temporary files, session data, system properties, configuration data, registry settings, logs) he should not be able to due to missing authorization, or due to excessive privileges (e.g. not using the principle of least privilege) | Eduardo can access data he does not have permission to, even though he has permission to the form/page/URL/entry point | Yuanjing can access application functions, objects, or properties he is not authorized to access |
| |  | | --- | | OWASP SCP  79, 80 | | OWASP ASVS  4.8 | | OWASP AppSensor  - | | CAPEC  122 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP 70,81,83-4,87-9, 99,117,131-2,142,154,170,179 | | OWASP ASVS  4.1, 4.4, 4.9,, 19.3 | | OWASP AppSensor  ACE1-4, HT2 | | CAPEC  75, 87, 95, 126, 149, 155, 203, 213, 264-5 | | SAFECode  8, 10, 11, 13 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  81, 88, 131 | | OWASP ASVS  4.1, 4.4 | | OWASP AppSensor  ACE1-4 | | CAPEC  122 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  81, 85, 86, 131 | | OWASP ASVS  4.1, 4.4 | | OWASP AppSensor  ACE1-4 | | CAPEC  122 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Authorization** | **8** | **Authorization** | **9** | **Authorization** | **10** | **Authorization** | **J** |
| Tom can bypass business rules by altering the usual process sequence or flow, or by undertaking the process in the incorrect order, or by manipulating date and time values used by the application, or by using valid features for unintended purposes, or by otherwise manipulating control data | Mike can misuse an application by using a valid feature too fast, or too frequently, or other way that is not intended, or consumes the application's resources, or causes race conditions, or over-utilizes a feature | Richard can bypass the centralized authorization controls since they are not being used comprehensively on all interactions | Dinis can access security configuration information, or access control lists |
| |  | | --- | | OWASP SCP  10, 32, 93, 94, 189 | | OWASP ASVS  4.10, 4.15, 4.16, 8.13, 15.1 | | OWASP AppSensor  ACE3 | | CAPEC  25, 39, 74, 162, 166, 207 | | SAFECode  8, 10, 11, 12 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  94 | | OWASP ASVS  4.14, 15.2 | | OWASP AppSensor  AE3, FIO1-2, UT2-4, STE1-3 | | CAPEC  26, 29, 119, 261 | | SAFECode  1, 35 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  78, 91 | | OWASP ASVS  1.7, 4.11 | | OWASP AppSensor  ACE1-4 | | CAPEC  36, 95, 121, 179 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  89, 90 | | OWASP ASVS  4.10, 13.2 | | OWASP AppSensor  - | | CAPEC  75, 133, 203 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |
| **Authorization** | **Q** | **Authorization** | **K** |  |  |  |  |
| Christopher can inject a command that the application will run at a higher privilege level | Ryan can influence or alter authorization controls and permissions, and can therefore bypass them | (No Card) | (No Card) |
| |  | | --- | | OWASP SCP  209 | | OWASP ASVS  5.12 | | OWASP AppSensor  - | | CAPEC  17, 30, 69, 234 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  77, 89, 91 | | OWASP ASVS  4.9, 4.10, 13.2 | | OWASP AppSensor  - | | CAPEC  207, 554 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Cryptography** | **A** | **Cryptography** |  | **Cryptography** | **2** | **Cryptography** | **3** |
| You have invented a new attack against Cryptography | (No Card) | Kyun can access data because it has been obfuscated rather than using an approved cryptographic function | Axel can modify transient or permanent data (stored or in transit), or source code, or updates/patches, or configuration data, because it is not subject to integrity checking |
| *Read more about this topic in OWASP's free Cheat Sheets on Cryptographic Storage, and Transport Layer Protection* |  | |  | | --- | | OWASP SCP  105, 133, 135 | | OWASP ASVS  - | | OWASP AppSensor  - | | CAPEC  - | | SAFECode  21, 29 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  92, 205, 212 | | OWASP ASVS  8.11, 11.7, 13.2, 19.5, 19.6, 19.7, 19.8 | | OWASP AppSensor  SE1, IE4 | | CAPEC  31, 39, 68, 75, 133, 145, 162, 203,438-9,442 | | SAFECode  12, 14 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |
| **Cryptography** | **4** | **Cryptography** | **5** | **Cryptography** | **6** | **Cryptography** | **7** |
| Paulo can access data in transit that is not encrypted, even though the channel is encrypted | Kyle can bypass cryptographic controls because they do not fail securely (i.e. they default to unprotected) | Romain can read and modify unencrypted data in memory or in transit (e.g. cryptographic secrets, credentials, session identifiers, personal and commercially-sensitive data), in use or in communications within the application, or between the application and users, or between the application and external systems | Gunter can intercept or modify encrypted data in transit because the protocol is poorly deployed, or weakly configured, or certificates are invalid, or certificates are not trusted, or the connection can be degraded to a weaker or un-encrypted communication |
| |  | | --- | | OWASP SCP  37, 88, 143, 214 | | OWASP ASVS  7.12, 9.2 | | OWASP AppSensor  - | | CAPEC  185, 186, 187 | | SAFECode  14, 29, 30 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  103, 145 | | OWASP ASVS  7.2, 10.3 | | OWASP AppSensor  - | | CAPEC  - | | SAFECode  21, 29 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  36, 37, 143, 146, 147 | | OWASP ASVS  2.16, 9.2, 9.11, 10.3, 19.2 | | OWASP AppSensor  - | | CAPEC  31, 57, 102, 157, 158, 384, 466, 546 | | SAFECode  29 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  75, 144, 145, 148 | | OWASP ASVS  10.1, 10.5, 10.10, 10.11, 10.12, 10.13, 10.14 | | OWASP AppSensor  IE4 | | CAPEC  31, 216 | | SAFECode  14, 29, 30 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Cryptography** | **8** | **Cryptography** | **9** | **Cryptography** | **10** | **Cryptography** | **J** |
| Eoin can access stored business data (e.g. passwords, session identifiers, PII, cardholder data) because it is not securely encrypted or securely hashed | Andy can bypass random number generation, random GUID generation, hashing and encryption functions because they have been self-built and/or are weak | Susanna can break the cryptography in use because it is not strong enough for the degree of protection required, or it is not strong enough for the amount of effort the attacker is willing to make | Justin can read credentials for accessing internal or external resources, services and others systems because they are stored in an unencrypted format, or saved in the source code |
| |  | | --- | | OWASP SCP  30, 31, 70, 133, 135 | | OWASP ASVS  2.13, 7.7, 7.8, 9.2 | | OWASP AppSensor  - | | CAPEC  31, 37, 55 | | SAFECode  21, 29, 31 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  60, 104, 105 | | OWASP ASVS  7.6, 7.7, 7.8, 7.15 | | OWASP AppSensor  - | | CAPEC  97 | | SAFECode  14, 21, 29, 32, 33 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  104, 105 | | OWASP ASVS  - | | OWASP AppSensor  - | | CAPEC  97, 463 | | SAFECode  14, 21, 29, 31, 32, 33 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  35, 90, 171, 172 | | OWASP ASVS  2.29 | | OWASP AppSensor  - | | CAPEC  116 | | SAFECode  21, 29 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |
| **Cryptography** | **Q** | **Cryptography** | **K** |  |  |  |  |
| Randolph can access or predict the master cryptographic secrets | Dan can influence or alter cryptography code/routines (encryption, hashing, digital signatures, random number and GUID generation) and can therefore bypass them | (No Card) | (No Card) |
| |  | | --- | | OWASP SCP  35, 102 | | OWASP ASVS  7.8, 7.9, 7.11, 7.13, 7.14 | | OWASP AppSensor  - | | CAPEC  116, 117 | | SAFECode  21, 29 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  31, 101 | | OWASP ASVS  7.11 | | OWASP AppSensor  - | | CAPEC  207, 554 | | SAFECode  14, 21, 29 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Cornucopia** | **A** | **Cornucopia** |  | **Cornucopia** | **2** | **Cornucopia** | **3** |
| You have invented a new attack of any type | (No Card) | Lee can bypass application controls because dangerous/risky programming language functions have been used instead of safer alternatives, or there are type conversion errors, or because the application is unreliable when an external resource is unavailable, or there are race conditions, or there are resource initialization or allocation issues, or overflows can occur | Andrew can access source code, or decompile, or otherwise access business logic to understand how the application works and any secrets contained |
| *Read more about application security in OWASP's free Guides on Requirements, Development, Code Review and Testing, the Cheat Sheet series, and the Open Software Assurance Maturity Model* |  | |  | | --- | | OWASP SCP  194-202, 205-209 | | OWASP ASVS  5.1 | | OWASP AppSensor  - | | CAPEC  25, 26, 29, 96, 123-4, 128-9, 264-5 | | SAFECode  3, 5-7, 9, 22, 25-26, 34 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  134 | | OWASP ASVS  19.5 | | OWASP AppSensor  - | | CAPEC  189, 207 | | SAFECode  - | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |
| **Cornucopia** | **4** | **Cornucopia** | **5** | **Cornucopia** | **6** | **Cornucopia** | **7** |
| Keith can perform an action and it is not possible to attribute it to him | Larry can influence the trust other parties including users have in the application, or abuse that trust elsewhere (e.g. in another application) | Aaron can bypass controls because error/exception handling is missing, or is implemented inconsistently or partially, or does not deny access by default (i.e. errors should terminate access/execution), or relies on handling by some other service or system | Mwengu's actions cannot be investigated because there is not an adequate accurately time-stamped record of security events, or there is not a full audit trail, or these can be altered or deleted by Mwengu, or there is no centralized logging service |
| |  | | --- | | OWASP SCP  23, 32, 34, 42, 51, 181 | | OWASP ASVS  8.10 | | OWASP AppSensor  - | | CAPEC  - | | SAFECode  - | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  - | | OWASP ASVS  - | | OWASP AppSensor  - | | CAPEC  89, 103, 181, 459 | | SAFECode  - | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  109, 110, 111, 112, 155 | | OWASP ASVS  8.2, 8.4 | | OWASP AppSensor  - | | CAPEC  54, 98, 164 | | SAFECode  4, 11, 23 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  113-115, 117, 118, 121-130 | | OWASP ASVS  2.12, 8.3-8.12, 9.10, 10.4 | | OWASP AppSensor  - | | CAPEC  93 | | SAFECode  4 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Cornucopia** | **8** | **Cornucopia** | **9** | **Cornucopia** | **10** | **Cornucopia** | **J** |
| David can bypass the application to gain access to data because the network and host infrastructure, and supporting services/applications, have not been securely configured, the configuration rechecked periodically and security patches applied, or the data is stored locally, or the data is not physically protected | Michael can bypass the application to gain access to data because administrative tools or administrative interfaces are not secured adequately | Xavier can circumvent the application's controls because code frameworks, libraries and components contain malicious code or vulnerabilities (e.g. in-house, commercial off the shelf, outsourced, open source, externally-located) | Roman can exploit the application because it was compiled using out-of-date tools, or its configuration is not secure by default, or security information was not documented and passed on to operational teams |
| |  | | --- | | OWASP SCP  151, 152, 156, 160, 161, 173-177 | | OWASP ASVS  19.1, 19.4, 19.6, 19.7, 19.8 | | OWASP AppSensor  RE1, RE2 | | CAPEC  37, 220, 310, 436, 536 | | SAFECode  - | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  23, 29, 56, 81, 82, 84-90 | | OWASP ASVS  2.1, 2.32 | | OWASP AppSensor  - | | CAPEC  122, 233 | | SAFECode  - | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  57, 151, 152, 204, 205, 213, 214 | | OWASP ASVS  1.11- | | OWASP AppSensor  - | | CAPEC  68, 438, 439, 442, 524, 538 | | SAFECode  15 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  90, 137, 148, 151-154, 175-179, 186, 192 | | OWASP ASVS  19.5, 19.9 | | OWASP AppSensor  - | | CAPEC  - | | SAFECode  4 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | |
| **Cornucopia** | **Q** | **Cornucopia** | **K** | **Wild Card** | **Joker** | **Wild Card** | **Joker** |
| Jim can undertake malicious, non-normal, actions without real-time detection and response by the application | Gareth can utilize the application to deny service to some or all of its users | Alice can utilize the application to attack users' systems and data | Bob can influence, alter or affect the application so that it no longer complies with legal, regulatory, contractual or other organizational mandates |
| |  | | --- | | OWASP SCP  - | | OWASP ASVS  4.14, 9.8, 15.1, 15.2 | | OWASP AppSensor  (All) | | CAPEC  - | | SAFECode  1, 27 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | |  | | --- | | OWASP SCP  41, 55 | | OWASP ASVS  - | | OWASP AppSensor  UT1-4, STE3 | | CAPEC  2, 25, 119, 125 | | SAFECode  1 | | OWASP Cornucopia Ecommerce Website Edition v1.20-EN | | *Have you thought about becoming an individual OWASP member? All tools, guidance and local meetings are free for everyone, but individual membership helps support OWASP's work* | *Examine vulnerabilities and discover how they can be fixed using training applications in the free OWASP Broken Web Applications VM, or using the online challenges in the free Hacking Lab* |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cut here |  |  |  |  |  |  |  |  |
|  | **${ }** |  |  |  |  |  | **J** |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |



Change Log

| Version / Date | | Comments |
| --- | --- | --- |
| 0.1 | 30 Jul 2012 | Original Draft |
| 0.2 | 10 Aug 2012 | Draft reviewed and updated |
| 0.3 | 15 Aug 2012 | Draft announced OWASP SCP mailing list for comment. |
| 0.4 | 25 Feb 2013 | ${Common\_T03200} ${Common\_T03210} ${Common\_T03220} Added contributors section, page numbering, FAQs and change log. |
| 1 | 25 Feb 2013 | Release. |
| 1.01 | 03 Jun 2013 | ${Common\_T03270} ${Common\_T03280} ${Common\_T03290} ${Common\_T03300} ${Common\_T03310} ${Common\_T03320} ${Common\_T03330} Project contributors added. |
| 1.02 | 14 Aug 2013 | ${Common\_T03360} ${Common\_T03370} ${Common\_T03380} ${Common\_T03390} ${Common\_T03400} Project contributors added. |
| 1.03 | 18 Sep 2013 | ${Common\_T03430} ${Common\_T03440} ${Common\_T03450} All remaining attack descriptions on cards changed to black (from dark grey) and background colours amended to provide more contrast and increase readability. |
| 1.04 | 01 Feb 2014 | Text “password change, password change,” corrected to “password change, password recovery,” on Queen of Authentication card. |
| 1.05 | 21 Mar 2014 | ${Common\_T03500} ${Common\_T03510} ${Common\_T03520} Podcast and video links added. |
| 1.1 | 04 Mar 2015 | ${Common\_T03550} ${Common\_T03560} Minor text changes to cards to improve readability. |
| 1.2 | 29 Jun 2016 | ${Common\_T03590} ${Common\_T03600} ${Common\_T03610} ${Common\_T03620} ${Common\_T03630} ${Common\_T03640} ${Common\_T03650} ${Common\_T03660} ${Common\_T03670} ${Common\_T03680} ${Common\_T03690} Minor text changes to instructions and FAQs. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ${Common\_T3800}  ${Common\_T3810}  ${Common\_T2820}  Without all their efforts, the project would not have progressed to this point.  Please contact the mailing list or project leaders directly, if anyone is missing from the below lists.   |  |  |  | | --- | --- | --- | | * Simon Bennetts * Tom Brennan * Fabio Cerullo * Oana Cornea * Johanna Curiel * Todd Dahl * Luis Enriquez * Ken Ferris * Darío De Filippis | * Sebastien Gioria * Tobias Gondrom * Timo Goosen * Anthony Harrison * John Herrlin * Jerry Hoff * Marios Kourtesis * Antonis Manaras * Jim Manico | * Mark Miller * Cam Morris * Susana Romaniz * Ravishankar Sahadevan * Tao Sauvage * Stephen de Vries * Colin Watson |  * •OWASP’s hard-working employees, especially Kate Hartmann * •Attendees at OWASP London, OWASP Manchester, OWASP Netherlands and OWASP Scotland chapter meetings, and the London Gamification meetup, who made helpful suggestions and asked challenging questions * •Blackfoot UK Limited for gifting print-ready design files and hundreds of professionally printed card decks for distribution by post and at OWASP chapter meetings * •OWASP NYC for creating an OWASP box design and distributing packs at AppSec USA 2014.   Podcasts and videos  The following supporting OWASP Cornucopia resources are available online:   * • Video - Using the cards, created during AppSec EU 2015 project summit, 20th May 2015<https://www.youtube.com/watch?v=i5Y0akWj31k> * https://www.youtube.com/watch?v=i5Y0akWj31k<http://trustedsoftwarealliance.com/2014/03/21/the-owasp-cornucopia-project-with-colin-watson/> * • Podcast interview, OWASP 24/7 Podcast channel, 21st March 2014<https://www.youtube.com/watch?v=Q_LE-8xNXVk>   http://trustedsoftwarealliance.com/2014/03/21/the-owasp-cornucopia-project-with-colin-watson/ |  |