|  |  |  |
| --- | --- | --- |
| WHStore:watsonhall:groups:owasp:OWASP_Image_Toolbox:owasp_logo_122106.eps  **Cornucopia**  **Ecommerce Website Edition v1.30-EN**  OWASP Cornucopia is a mechanism to assist software development teams identify security requirements in Agile, conventional and formal development processes.  Author  Colin Watson  Project Leaders  Colin Watson and Grant Ongers  Reviewers  Tom Brennan, Johanna Curiel, Darío De Filippis and Timo Goosen  Acknowledgments  Microsoft SDL Team for the Elevation of Privilege Threat Modelling Game, published under a Creative Commons Attribution license, as the inspiration for Cornucopia and from which many ideas, especially the game theory, were copied.  Keith Turpin and contributors to the “OWASP Secure Coding Practices - Quick Reference Guide”, originally donated to OWASP by Boeing, which is used as the primary source of security requirements information to formulate the content of the cards.  Contributors, supporters, sponsors and volunteers to the OWASP ASVS, AppSensor and Web Framework Security Matrix projects, Mitre’s Common Attack Pattern Enumeration and Classification (CAPEC), and SAFECode’s “Practical Security Stories and Security Tasks for Agile Development Environments” which are all used in the cross-references provided.  Playgen for providing an illuminating afternoon seminar on task gamification, and tartanmaker.com for the online tool to help create the card back pattern.  Blackfoot UK Limited for creating and donating print-ready design files, Tom Brennan and the OWASP Foundation for instigating the creation of an OWASP-branded box and leaflet, and OWASP employees, especially Kate Hartmann, for managing the ordering, stocking and despatch of printed card decks. Oana Cornea and other participants at the AppSec EU 2015 project summit for their help in creating the demonstration video. Colin Watson as author and co-project leader with Grant Ongers, along with other OWASP volunteers who have helped in many ways.  OWASP does not endorse or recommend commercial products or services © 2012-2024 OWASP Foundation This document is licensed under the Creative Commons Attribution-ShareAlike 3.0 license | |  |
| Introduction  The idea behind Cornucopia is to help development teams, especially those using Agile methodologies, to identify application security requirements and develop security-based user stories. 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. EoP is a great concept and game strategy, and was published under a Creative Commons Attribution License.  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)  Instead of EoP’s STRIDE suits (sets of cards with matching designs), Cornucopia suits are based on the structure of the OWASP Secure Coding Practices - Quick Reference Guide (SCP), but with additional consideration of sections in the OWASP Application Security Verification Standard, the OWASP Testing Guide and David Rook’s Principles of Secure Development. 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 v4.0 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. |
| Provide feedback  If you have ideas or feedback on the use of OWASP Cornucopia, please share them. Even better if you create alternative versions of the cards, or produce professional print-ready versions, please share that with the volunteers who created this edition and with the wider application development and application security community.  The best place to use to discuss or contribute is the mailing list for the OWASP project:   * Mailing list <https://lists.owasp.org/mailman/listinfo/owasp_cornucopia> * Project home page <https://www.owasp.org/index.php/OWASP_Cornucopia>   All OWASP documents and tools are free to download and use. OWASP Cornucopia is licensed under the Creative Commons Attribution-ShareAlike 3.0 license. |  | 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 An example is:  *William has control over the generation of session identifiers.*  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.  Further guidance about each card is available in the online Wiki Deck at  <https://wiki.owasp.org/index.php/Cornucopia_-_Ecommerce_Website_Edition_-_Wiki_Deck>  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 (ref: [OWASP SCP Quick Reference Guide v2.1](https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/assets/docs/OWASP_SCP_Quick_Reference_Guide_v21.pdf) ) * Verification IDs in “Application Security Verification Standard (ASVS) for Web Applications” (ref: [ASVS v3 and v4 downloads](https://github.com/OWASP/ASVS/) ) * Attack detection points IDs in “AppSensor”, OWASP, August 2010-2015 (ref: [AppSensor DetectionPoints](https://www.owasp.org/index.php/AppSensor_DetectionPoints)) * IDs in “Common Attack Pattern Enumeration and Classification (CAPEC)”, v2.8, Mitre Corporation, November 2015 (ref: [capec (31. July 2018)](https://capec.mitre.org/data/definitions/2000.html)) * Security-focused stories in 'Practical Security Stories and Security Tasks for Agile Development Environments', SAFECode, July 2012 (ref: [SAFECode Agile Dev Security](https://safecode.org/publication/SAFECode_Agile_Dev_Security0712.pdf))   A look-up means the attack is included within the referenced item, but does not necessarily encompass the whole of its intent. 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. There are no lookups on the six Aces and two Jokers. Instead these cards have some general tips in italicized text.  It is possible to play Cornucopia in many different ways. Here is one way, demonstrated online in a video at (refL [Cornucopia scoresheet](https://www.owasp.org/index.php/File:Cornucopia-scoresheet.pdf) ) which uses the new (May 2015) score/record sheet at |
| A - Preparations  A1. Obtain a deck, or print your own deck of Cornucopia cards (see page 2 of this document) and separate/cut out the cards A2. Identify an application or application process to review; this might be a concept, design or an actual implementation A3. Create a data flow diagram, user stories, or other artefacts to help the review A4. Identify and invite a group of 3-6 architects, developers, testers and other business stakeholders together and sit around a table (try to include someone fairly familiar with application security) A5. Have some prizes to hand (gold stars, chocolate, pizza, beer or flowers depending upon your office culture)  B - Play  One suit - Cornucopia - acts as trumps. Aces are high (i.e. they beat Kings). It helps if there is a non-player to document the issues and scores.  B1. Remove the Jokers and a few low-score (2, 3, 4) cards from Cornucopia suit to ensure each player will have the same number of cards B2. Shuffle the deck and deal all the cards B3. To begin, choose a player randomly who will play the first card - they can play any card from their hand except from the trump suit - Cornucopia B4. To play a card, each player must read it out aloud, and explain (see the online Wiki Deck for tips) how the threat could apply (the player gets a point for attacks that might work which the group thinks is an actionable bug) - do not try to think of mitigations at this stage, and do not exclude a threat just because of a belief that it is already mitigated - someone note the card and record the issues raised B5. Play clockwise, each person must play a card in the same way; if you have any card of the matching lead suit you must play one of those, otherwise they can play a card from any other suit. Only a higher card of the same suit, or the highest card in the trump suit Cornucopia, wins the hand.  B6. The person who wins the round, leads the next round (i.e. they play first), and thus defines the next lead suit B7. Repeat until all the cards are played |  | C - Scoring  The objective is to identify applicable threats, and win hands (rounds):  C1. Score +1 for each card you can identify as a valid threat to the application under consideration C2. Score +1 if you win a round C3. Once all cards have been played, whoever has the most points wins  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. |
| 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. Apart from the “trumps card game” rules described above which are very similar to the EoP, the deck can also be played as the “twenty-one card game” (also known as “pontoon” or “blackjack”) which normally reduces the number of cards played in each round.  Practice on an imaginary application, or even a future planned application, rather than trying to find fault with existing applications until the participants are happy with the usefulness of the game.  Consider just playing with one suit to make a shorter session – but try to cover all the suits for every project. Or even better just play one hand with some pre-selected cards, and score only on the ability to identify security requirements. Perhaps have one game of each suit each day for a week or so, if the participants cannot spare long enough for a full deck.  Some teams have preferred to play a full hand of cards, and then discuss what is on the cards after each round (instead of after each person plays a card).  Another suggestion is that if a player fails to identify the card is relevant, allow other players to suggest ideas, and potentially let them gain the point for the card. Consider allowing extra points for especially good contributions.  You can even play by yourself. Just use the cards to act as thought-provokers. Involving more people will be beneficial though.  In Microsoft's EoP guidance, they recommend cheating as a good game strategy.  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.  Consider removing the following cards from the decks if you are confidence they are addressed by the way you are using the language/framework. Items in parentheses are “maybes”. |  | 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]*  Authentication  *[your list]* | Session management  *[your list]*  Authorization  *[your list]* | Cryptography  *[your list]*  Cornucopia  *[your list]* | |  |  |  |   Compliance requirement decks  Create a smaller deck by only including cards for a particular compliance requirement.   |  |  |  | | --- | --- | --- | | Compliance requirement | | | | Data validation and encoding  *[compliance list]*  Authentication  *[compliance list]* | Session management  *[compliance list]*  Authorization  *[compliance list]* | Cryptography  *[compliance list]*  Cornucopia  *[compliance list]* | |
| ***Frequently asked questions***  *1. Can I copy or edit the game?*  *Yes of course. All OWASP materials are free to do with as you like provided you comply with the Creative Commons Attribution-ShareAlike 3.0 license. Perhaps if you create a new version, you might donate it to the OWASP Cornucopia Project?*  *2. How can I get involved?*  *Please send ideas or offers of help to the project’s mailing list.*  *3. How were the attackers’ names chosen?*  *EoP begins every description with words like 'An attacker can...'. These have to be phrased as an attack but I was not keen on the anonymous terminology, wanting something more engaging, and therefore used personal names. These can be thought of as external or internal people or aliases for computer systems. But instead of just random names, I thought how they might reflect the OWASP community aspect. Therefore, apart from 'Alice and Bob' I use the given (first) names of current and recent OWASP employees and Board members (assigned in no order), and then randomly selected the remaining 50 or so names from the current list of paying individual OWASP members. No name was used more than once, and where people had provided two personal names, I dropped one part to try to ensure no-one can be easily identified. Names were not deliberately allocated to any particular attack, defence or requirement. The cultural and gender mix simply reflects theses sources of names, and is not meant to be world-representative. 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/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* |  | |  | | --- | | OWASP SCP  69, 107-109, 136-137, 153, 156, 158, 162 | | OWASP ASVS  1.6.4, 2.10.4, 4.3.2, 7.1.1, 10.2.3, 14.1.1, 14.2.2, 14.3.3 | | OWASP APPSENSOR  HT1-3 | | CAPEC  54, 541 | | SAFECODE  4, 23 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  - | | OWASP ASVS  1.5.3, 5.1.1-5.1.4, 13.2.1, 14.1.2, 14.4.1 | | OWASP APPSENSOR  RE7-8, AE4-7, IE2-3, CIE1, CIE3-4, HT1-3 | | CAPEC  28, 48, 126, 165, 213, 220-221, 261-262, 271-272 | | SAFECODE  3, 16, 24, 35 | | $Common\_Title\_full} | |  |
|  | **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 |  |
|  | |  | | --- | | OWASP SCP  8, 10, 183 | | OWASP ASVS  4.2.1, 5.1.1, 5.1.2, 11.1.1, 11.1.2 | | OWASP APPSENSOR  RE3-6, AE8-11, SE1, SE3-6, IE2-4, HT1-3 | | CAPEC  28, 31, 48, 126, 162, 165, 213, 220-221, 261 | | SAFECODE  24, 35 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  3, 15, 18-22, 168 | | OWASP ASVS  1.1.6, 5.3.3, 5.2.1, 5.2.2, 5.2.5 | | OWASP APPSENSOR  - | | CAPEC  28, 31, 152, 160, 468 | | SAFECODE  2, 17 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  3, 168 | | OWASP ASVS  1.1.6, 1.5.3, 5.1.3, 13.2.2, 13.2.5 | | OWASP APPSENSOR  IE2-3 | | CAPEC  28 | | SAFECODE  3, 16, 24 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  4-5, 7, 150 | | OWASP ASVS  1.5.3, 13.2.2, 13.2.5 | | OWASP APPSENSOR  IE2-3, EE1-2 | | CAPEC  28, 153, 165 | | SAFECODE  3, 16, 24 | | $Common\_Title\_full} | |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **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 |
| |  | | --- | | OWASP SCP  15, 169 | | OWASP ASVS  1.1.6, 5.2.2, 5.2.5 | | OWASP APPSENSOR  - | | CAPEC  28, 31, 152, 160, 468 | | SAFECODE  2, 17 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  6, 21-22, 168 | | OWASP ASVS  7.1.3 | | OWASP APPSENSOR  IE2-3 | | CAPEC  28 | | SAFECODE  3, 16, 24 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  2, 19, 92, 95, 180 | | OWASP ASVS  1.12.2, 5.1.3, 9.2.3, 12.2.1, 12.3.1-12.3.3, 12.4.2, 12.5.2, 14.5.3 | | OWASP APPSENSOR  IE4, IE5 | | CAPEC  12, 51, 57, 90, 111, 145, 194-195, 202, 218, 463 | | SAFECODE  14 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  1, 17 | | OWASP ASVS  1.5.3 | | OWASP APPSENSOR  RE3, RE4 | | CAPEC  87, 207, 554 | | SAFECODE  2, 17 | | $Common\_Title\_full} | |
| **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) |
| |  | | --- | | OWASP SCP  10, 15-16, 19-20 | | OWASP ASVS  5.2.1, 5.2.5, 5.3.3, 5.5.4 | | OWASP APPSENSOR  IE1, RP3 | | CAPEC  28, 31, 152, 160, 468 | | SAFECODE  2, 17 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  15, 19-22, 167, 180, 204, 211-212 | | OWASP ASVS  5.2.1, 5.2.2, 5.3.4, 5.3.7-5.3.10 | | OWASP APPSENSOR  CIE1, CIE2 | | CAPEC  23, 28, 76, 152, 160, 261 | | SAFECODE  2, 19-20 | | $Common\_Title\_full} | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **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* |  | |  | | --- | | OWASP SCP  47, 52 | | OWASP ASVS  2.5.2, 7.1.2, 7.1.4, 7.2.1, 8.2.1-8.2.3, 8.3.6 | | OWASP APPSENSOR  UT1 | | CAPEC  - | | SAFECODE  28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  36-37, 40, 43, 48, 51, 119, 139-140, 146 | | OWASP ASVS  2.5.2, 2.5.3 | | OWASP APPSENSOR  - | | CAPEC  37, 546 | | SAFECODE  28 | | $Common\_Title\_full} | |
| **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 |
| |  | | --- | | OWASP SCP  33, 53 | | OWASP ASVS  2.2.1, 4.1.5 | | OWASP APPSENSOR  AE1 | | CAPEC  383 | | SAFECODE  28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  54, 175, 178 | | OWASP ASVS  4.1.5 | | OWASP APPSENSOR  AE12, HT3 | | CAPEC  70 | | SAFECODE  28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  37, 45-46, 178 | | OWASP ASVS  2.5.6 | | OWASP APPSENSOR  - | | CAPEC  50 | | SAFECODE  28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  33, 38-39, 41, 50, 53 | | OWASP ASVS  2.1.2, 2.1.7, 2.1.10, 2.2.1, 2.2.1 | | OWASP APPSENSOR  AE2, AE3 | | CAPEC  2, 16 | | SAFECODE  27 | | 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 |
| |  | | --- | | OWASP SCP  28 | | OWASP ASVS  4.1.5 | | OWASP APPSENSOR  - | | CAPEC  115 | | SAFECODE  28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  55-56 | | OWASP ASVS  1.4.3, 1.4.5, 2.1.6, 2.2.4, 4.3.3 | | OWASP APPSENSOR  - | | CAPEC  21 | | SAFECODE  14, 28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  25-27 | | OWASP ASVS  1.1.6, 1.4.4 | | OWASP APPSENSOR  - | | CAPEC  90, 115 | | SAFECODE  14, 28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  23, 32, 34 | | OWASP ASVS  1.4.3, 1.4.5 | | OWASP APPSENSOR  - | | CAPEC  115 | | SAFECODE  14, 28 | | $Common\_Title\_full} | |
| **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) |
| |  | | --- | | OWASP SCP  23, 29, 42, 49 | | OWASP ASVS  1.4.3, 1.4.5, 2.5.6, 2.5.7 | | OWASP APPSENSOR  - | | CAPEC  36, 50, 115, 121, 179 | | SAFECODE  14, 28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  24 | | OWASP ASVS  4.1.1, 10.2.3-10.2.6 | | OWASP APPSENSOR  - | | CAPEC  115, 207, 554 | | SAFECODE  14, 28 | | $Common\_Title\_full} | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **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* |  | |  | | --- | | OWASP SCP  58-59 | | OWASP ASVS  3.7.1 | | OWASP APPSENSOR  SE2 | | CAPEC  31, 60-61 | | SAFECODE  28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  68 | | OWASP ASVS  3.3.3, 3.3.4 | | OWASP APPSENSOR  - | | CAPEC  - | | SAFECODE  28 | | $Common\_Title\_full} | |
| **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 |
| |  | | --- | | OWASP SCP  59, 61 | | OWASP ASVS  3.4.1-3.4.5 | | OWASP APPSENSOR  SE2 | | CAPEC  31, 61 | | SAFECODE  28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  60, 62, 66-67, 71-72 | | OWASP ASVS  3.2.1, 3.2.2, 3.2.4, 3.3.1 | | OWASP APPSENSOR  SE4-6 | | CAPEC  31 | | SAFECODE  28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  64-65 | | OWASP ASVS  3.3.2-3.3.4 | | OWASP APPSENSOR  SE5, SE6 | | CAPEC  21 | | SAFECODE  28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  62-63 | | OWASP ASVS  3.3.1, 3.3.4 | | OWASP APPSENSOR  - | | CAPEC  21 | | SAFECODE  28 | | $Common\_Title\_full} | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **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 |
| |  | | --- | | OWASP SCP  96 | | OWASP ASVS  3.6.1, 3.3.2 | | OWASP APPSENSOR  - | | CAPEC  21 | | SAFECODE  28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  69, 75-76, 119, 138 | | OWASP ASVS  1.9.1, 3.1.1, 7.1.1, 7.1.2, 7.2.1, 9.1.3, 9.2.2 | | OWASP APPSENSOR  SE4-6 | | CAPEC  31, 60 | | SAFECODE  28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  73-74 | | OWASP ASVS  4.2.2 | | OWASP APPSENSOR  IE4 | | CAPEC  62, 111 | | SAFECODE  18 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  - | | OWASP ASVS  11.1.1-11.1.3 | | OWASP APPSENSOR  IE5 | | CAPEC  60 | | SAFECODE  12, 14 | | 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) |
| |  | | --- | | OWASP SCP  58 | | OWASP ASVS  1.1.6, 3.7.1 | | OWASP APPSENSOR  - | | CAPEC  21 | | SAFECODE  14, 28 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  58, 60 | | OWASP ASVS  1.1.6 | | OWASP APPSENSOR  - | | CAPEC  21 | | SAFECODE  14, 28 | | $Common\_Title\_full} | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **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 he 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 through other information leakage |
| *Read more about this topic in OWASP's Development and Testing Guides* |  | |  | | --- | | OWASP SCP  44 | | OWASP ASVS  4.1.3, 4.2.1, 5.1.5 | | OWASP APPSENSOR  - | | CAPEC  153 | | SAFECODE  8, 10-11 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  51, 100, 135, 139-141, 150 | | OWASP ASVS  1.12.1, 4.1.3, 4.1.5, 8.1.2, 8.2.1, 8.3.1, 8.3.4, 8.3.6, 8.3.8, 12.4.1 | | OWASP APPSENSOR  - | | CAPEC  69, 213 | | SAFECODE  8, 10-11 | | $Common\_Title\_full} | |
| **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  4.1.5 | | OWASP APPSENSOR  - | | CAPEC  122 | | SAFECODE  8, 10-11 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  70, 81, 83-4, 87-9, 99, 117, 131-2, 142, 154, 170, 179 | | OWASP ASVS  1.2.2, 4.1.1, 4.1.3, 4.2.1 | | OWASP APPSENSOR  ACE1, ACE2, ACE3, ACE4, HT2 | | CAPEC  75, 87, 95, 126, 149, 155, 203, 213, 264-265 | | SAFECODE  8, 10-11, 13 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  81, 88, 131 | | OWASP ASVS  4.1.3, 4.2.1 | | OWASP APPSENSOR  ACE1-4 | | CAPEC  122 | | SAFECODE  8, 10-11 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  81, 85-86, 131 | | OWASP ASVS  4.1.3, 4.2.1 | | OWASP APPSENSOR  ACE1-4 | | CAPEC  122 | | SAFECODE  8, 10-11 | | $Common\_Title\_full} | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **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  4.1.2, 4.2.1, 4.3.3, 7.3.4, 11.1.1, 11.1.2 | | OWASP APPSENSOR  ACE3 | | CAPEC  25, 39, 74, 162, 166, 207 | | SAFECODE  8, 10-12 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  94 | | OWASP ASVS  11.1.3, 11.1.4 | | OWASP APPSENSOR  AE3, FIO1-2, UT2-4, STE1-3 | | CAPEC  26, 29, 119, 261 | | SAFECODE  1, 35 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  78, 91 | | OWASP ASVS  1.1.6, 4.1.1 | | OWASP APPSENSOR  ACE1-4 | | CAPEC  36, 95, 121, 179 | | SAFECODE  8, 10-11 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  89-90 | | OWASP ASVS  4.1.2, 10.2.3, 10.2.3-10.2.6 | | OWASP APPSENSOR  - | | CAPEC  75, 133, 203 | | SAFECODE  8, 10-11 | | $Common\_Title\_full} | |
| **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  5.3.8 | | OWASP APPSENSOR  - | | CAPEC  17, 30, 69, 234 | | SAFECODE  8, 10-11 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  77, 89, 91 | | OWASP ASVS  4.1.1, 4.1.2, 10.2.3-10.2.6 | | OWASP APPSENSOR  - | | CAPEC  207, 554 | | SAFECODE  8, 10-11 | | $Common\_Title\_full} | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **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  6.2.2 | | OWASP APPSENSOR  - | | CAPEC  - | | SAFECODE  21, 29 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  92, 205, 212 | | OWASP ASVS  14.1.1, 14.1.4, 14.1.5, 10.2.3-10.2.6, 10.3.1, 10.3.2 | | OWASP APPSENSOR  SE1, IE4 | | CAPEC  31, 39, 68, 75, 133, 145, 162, 203, 438-439, 442 | | SAFECODE  12, 14 | | $Common\_Title\_full} | |
| **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  6.1.1, 8.3.4, 9.1.1 | | OWASP APPSENSOR  - | | CAPEC  185-187 | | SAFECODE  14, 29-30 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  103, 145 | | OWASP ASVS  1.9.1, 6.2.1, 9.1.3, 9.2.2 | | OWASP APPSENSOR  - | | CAPEC  - | | SAFECODE  21, 29 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  36-37, 143, 146-147 | | OWASP ASVS  1.9.1, 2.2.5, 2.5.1, 8.3.4, 8.3.6, 9.1.3, 9.2.2 | | OWASP APPSENSOR  - | | CAPEC  31, 57, 102, 157-158, 384, 466, 546 | | SAFECODE  29 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  75, 144-145, 148 | | OWASP ASVS  1.9.2, 6.2.7, 9.1.1, 9.2.1, 9.2.4, 14.4.5 | | OWASP APPSENSOR  IE4 | | CAPEC  31, 216 | | SAFECODE  14, 29-30 | | $Common\_Title\_full} | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **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  2.4.1, 6.2.2, 6.2.3, 8.3.4 | | OWASP APPSENSOR  - | | CAPEC  31, 37, 55 | | SAFECODE  21, 29, 31 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  60, 104-105 | | OWASP ASVS  6.2.2, 6.2.3, 6.3.1, 6.3.3 | | OWASP APPSENSOR  - | | CAPEC  97 | | SAFECODE  14, 21, 29, 32-33 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  104-105 | | OWASP ASVS  6.3.3 | | OWASP APPSENSOR  - | | CAPEC  97, 463 | | SAFECODE  14, 21, 29, 31-33 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  35, 90, 171-172 | | OWASP ASVS  1.6.1, 1.6.2, 1.6.4, 2.10.4, 6.4.1, 6.4.2 | | OWASP APPSENSOR  - | | CAPEC  116 | | SAFECODE  21, 29 | | $Common\_Title\_full} | |
| **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  1.6.1, 1.6.2, 1.6.3, 6.2.3, 8.3.6 | | OWASP APPSENSOR  - | | CAPEC  116-117 | | SAFECODE  21, 29 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  31, 101 | | OWASP ASVS  1.6.2, 6.2.5-6.2.8 | | OWASP APPSENSOR  - | | CAPEC  207, 554 | | SAFECODE  14, 21, 29 | | $Common\_Title\_full} | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **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  14.1.2 | | OWASP APPSENSOR  - | | CAPEC  25-26, 29, 96, 123-124, 128-129, 264-265 | | SAFECODE  3, 5-7, 9, 22, 25-26, 34 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  134 | | OWASP ASVS  14.1.1 | | OWASP APPSENSOR  - | | CAPEC  189, 207 | | SAFECODE  - | | $Common\_Title\_full} | |
| **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  7.2.1, 7.2.2 | | OWASP APPSENSOR  - | | CAPEC  - | | SAFECODE  - | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  - | | OWASP ASVS  1.9.2, 9.1.1, 5.1.5, 9.2.1, 9.2.4 | | OWASP APPSENSOR  - | | CAPEC  89, 103, 181, 459 | | SAFECODE  - | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  109-112, 155 | | OWASP ASVS  4.1.5, 7.1.4 | | OWASP APPSENSOR  - | | CAPEC  54, 98, 164 | | SAFECODE  4, 11, 23 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  113-115, 117-118, 121-130 | | OWASP ASVS  7.1.2, 7.1.4, 7.2.1, 7.2.2, 7.3.1-7.3.3, 8.3.5, 9.2.5 | | OWASP APPSENSOR  - | | CAPEC  93 | | SAFECODE  4 | | $Common\_Title\_full} | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **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  1.4.5, 10.3.1, 10.3.2, 14.1.4, 14.1.5, 14.2.1, 14.2.2 | | OWASP APPSENSOR  RE1, RE2 | | CAPEC  37, 220, 310, 436, 536 | | SAFECODE  - | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  23, 29, 56, 81-82, 84-90 | | OWASP ASVS  1.4.3, 1.4.5, 4.3.1 | | OWASP APPSENSOR  - | | CAPEC  122, 233 | | SAFECODE  - | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  57, 151-152, 204-205, 213-214 | | OWASP ASVS  1.14..3, 10.1.1, 10.2.3-10.2.6, 14.2.1 | | OWASP APPSENSOR  - | | CAPEC  68, 438-439, 442, 524, 538 | | SAFECODE  15 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  90, 137, 148, 151-154, 175-179, 186, 192 | | OWASP ASVS  1.14.3, 14.1.1-14.1.5, 14.2.1 | | OWASP APPSENSOR  - | | CAPEC  - | | SAFECODE  4 | | $Common\_Title\_full} | |
| **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  8.1.4, 11.1.1-11.1.4 | | OWASP APPSENSOR  (All) | | CAPEC  - | | SAFECODE  1, 27 | | $Common\_Title\_full} | | |  | | --- | | OWASP SCP  41, 55 | | OWASP ASVS  2.2.1, 11.1.3, 11.1.4 | | OWASP APPSENSOR  UT1-4, STE3 | | CAPEC  2, 25, 119, 125 | | SAFECODE  1 | | $Common\_Title\_full} | | *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 | Play rules updated based on feedback during workshops. Added reference to PCI SSC Information Supplement: PCI DSS E-commerce Guidelines. Descriptive text extended and updated. Added contributors section, page numbering, FAQs and change log. |
| 1 | 25 Feb 2013 | Release. |
| 1.01 | 03 Jun 2013 | Framework-specific card deck discussion added Additional FAQs created. Descriptive text updated. New cover image, and previous cover image moved to back. Cut lines added. FAQs 5 and 6 added. Attack descriptions on cards with tinted backgrounds changed to black (from dark grey). Project contributors added. |
| 1.02 | 14 Aug 2013 | Warning about time to print added. Additional alternative game rules added (twenty-one, play a deck over a week, play full hand and then discuss). Compliance deck concept added. FAQs 5 and 6 added. Attack descriptions on cards with tinted backgrounds changed to black (from dark grey). Project contributors added. |
| 1.03 | 18 Sep 2013 | Minor attack wording changes on two cards. OWASP SCP and ASVS cross-references checked and updated. Code letters added for suits. 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 | Updates to alternative game rules. Additional FAQs created. Contributors updated. Podcast and video links added. |
| 1.1 | 04 Mar 2015 | Change log date corrected for v1.05. Cross-references updated for 2014 version of ASVS. Contributors updated. Minor text changes to cards to improve readability. |
| 1.2 | 29 Jun 2016 | Video mentioned/linked Separate score sheet mentioned/linked. Previous embedded score sheet pages deleted Correction (identified by Tom Brennan) and addition to text on card 8 Authentication. Oana Cornea and other participants at the AppSec EU 2015 project summit added to list of contributors. Darío De Filippis added as project co-leader. Wiki Deck link added Cross-references updated for ASVS v3.0.1 and CAPEC v2.8. Minor text changes to a small number of cards. Added “-EN” to version number in preparation for “-ES” version. Susana Romaniz added as a contributor to the Spanish translation. Minor text changes to instructions and FAQs. |
| 1.3 | 01 Jan 2023 | Cross-references updated from ASVS v3.0.1 to ASVS v4.0 by Johan Sydseter. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Project contributors  All OWASP projects rely on the voluntary efforts of people in the software development and information security sectors.  They have contributed their time and energy to make suggestions, provide feedback, write, review and edit documentation, give encouragement, trial the game, and promote the concept.  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 * Johan Sydseter |  * 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> * Podcast interview, OWASP 24/7 Podcast channel, 21st March 2014 <http://trustedsoftwarealliance.com/2014/03/21/the-owasp-cornucopia-project-with-colin-watson/> * Video of presentation, OWASP EU Tour 2013 London, 3rd June 2013 <https://www.youtube.com/watch?v=Q_LE-8xNXVk>   See the project website for further information and presentation materials. |  |