This additional document contains the descriptions and examples of the newly proposed 11 common privacy weaknesses that have not covered in the existing CWEs (see Section 5 in the Common Privacy Weaknesses and Vulnerabilities in Software Applications manuscript for more details). Their detailed information is presented below.

Table 1: An example of unauthorised personal data modification weakness.

Subcategory: Insecurity

Class: Allowing unauthorised actors to modify personal data

Name: Unauthorised personal data modification

Description: The software lacks mechanisms to protect personal data from actors who are not authorised to modify personal data.

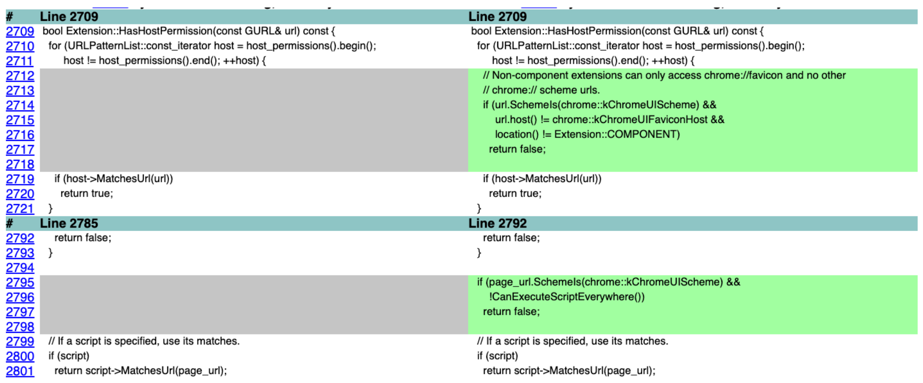
Mode of introduction: Phase: Implementation. Description: This weakness is caused by missing the implementation of personal data protection mechanisms to restrict actors who can modify personal data.

Common consequence: Scope: Integrity. Impact: Unauthorised modification. Note: This weakness violates user privacy as the personal data is modified without authorisation and user awareness. This may also affect the processing that uses this personal data in the software.

Detection method: Method: Manual analysis. Description: Evaluate parts in software that involve with modifying personal data and check whether the protection mechanisms are applied.

Potential mitigation: Phase: Implementation. Strategy: Implement an authorisation mechanism. Description: The software development team should implement a proper personal data protection mechanism to allow only authorised actors to modify personal data.

Demonstrative example: Issue 83010 in Chrome reports that the system allows the attackers to access and modify personal data via a malicious extension. The affected data includes saved passwords, preferences and permissions. References: `https://bugs.chromium.org/p/chromium/issues/detail?id=83010` and `https://src.chromium.org/viewvc/chrome/branches/742/src/chrome/common/extensions/extension.cc?r1=86315\&r2=86314\&pathrev=86315`.



```
#  Line 2709                                                      Line 2709
bool Extension::HasHostPermission(const GURL& url) const {         bool Extension::HasHostPermission(const GURL& url) const {
  for (URLPatternList::const_iterator host = host_permissions().begin();    for (URLPatternList::const_iterator host = host_permissions().begin();
       host != host_permissions().end(); ++host) {                      host != host_permissions().end(); ++host) {
                                                                    // Non-component extensions can only access chrome://favicon and no other
                                                                    // chrome:// scheme urls.
                                                                    if (url.SchemeIs(chrome::kChromeUIScheme) &&
                                                                        url.host() != chrome::kChromeUIFaviconHost &&
                                                                        location() != Extension::COMPONENT)
                                                                      return false;

    if (host->MatchesUrl(url))                                      if (host->MatchesUrl(url))
      return true;                                                    return true;
  }                                                               }
#  Line 2785                                                      Line 2792
    return false;                                                   return false;
  }                                                               }

                                                                  if (page_url.SchemeIs(chrome::kChromeUIScheme) &&
                                                                      !CanExecuteScriptEverywhere())
                                                                    return false;

  // If a script is specified, use its matches.                   // If a script is specified, use its matches.
  if (script)                                                     if (script)
    return script->MatchesUrl(page_url);                            return script->MatchesUrl(page_url);
```

Figure 1: An example of unauthorised personal data modification in Chrome (Issue 83010).

Table 2: An example of missing user privacy preference settings weakness.

Subcategory: Exclusion

Class: Not showing options asking for user privacy preferences

Name: Missing user privacy preference settings

Description: The software does not show options asking the users to set their privacy preferences (e.g. audience who can view their profiles). The users should be able to select how they prefer to manage their user privacy preference settings.

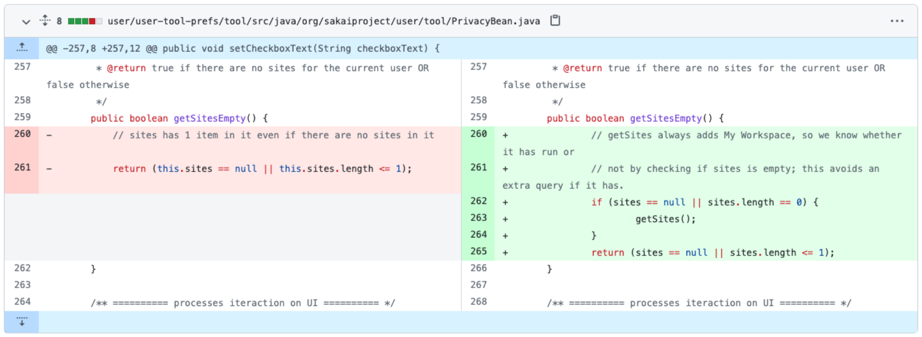
Mode of introduction: Phase: Architecture and Design. Description: This weakness is caused by not considering user privacy preference settings in the architecture and design phase.

Common consequence: Scope: Privacy. Impact: Privacy violation. Note: Users cannot determine their user privacy preferences in the software.

Detection method: Method: Manual analysis. Description: Evaluate the functions related to personal data processing.

Potential mitigation: Phase: Architecture and Design. Strategy: Consider user privacy preferences. Description: The software development team should consider which functions in the system need to provide options for users to determine their user privacy preferences.

Demonstrative example: Issue SAK-21599 in Sakai repository (e-learning platform) reports that the software does not show options for users to set privacy status preferences in selected courses. References: `https://jira.sakaiproject.org/browse/SAK-21599` and `https://github.com/sakaiproject/sakai/commit/c1ae444`.



```
8 user/user-tool-prefs/tool/src/java/org/sakaiproject/user/tool/PrivacyBean.java
@@ -257,8 +257,12 @@ public void setCheckboxText(String checkboxText) {
```

```
         * @return true if there are no sites for the current user OR false otherwise
         */
        public boolean getSitesEmpty() {
-               // sites has 1 item in it even if there are no sites in it
-               return (this.sites == null || this.sites.length <= 1);
        }

        /** ========== processes iteraction on UI ========== */
```

```
         * @return true if there are no sites for the current user OR false otherwise
         */
        public boolean getSitesEmpty() {
+               // getSites always adds My Workspace, so we know whether it has run or
+               // not by checking if sites is empty; this avoids an extra query if it has.
+               if (sites == null || sites.length == 0) {
+                       getSites();
+               }
+               return (sites == null || sites.length <= 1);
        }

        /** ========== processes iteraction on UI ========== */
```

Figure 2: An example of missing privacy preference settings in Sakai repository (Issue SAK-21599).

Table 3: An example of unsuccessful privacy preferences modification weakness.

Subcategory: Exclusion

Class: Not allowing a user to modify his/her privacy preferences

Name: Unsuccessful privacy preferences modification

Description: The software does not allow users to modify their privacy preferences.

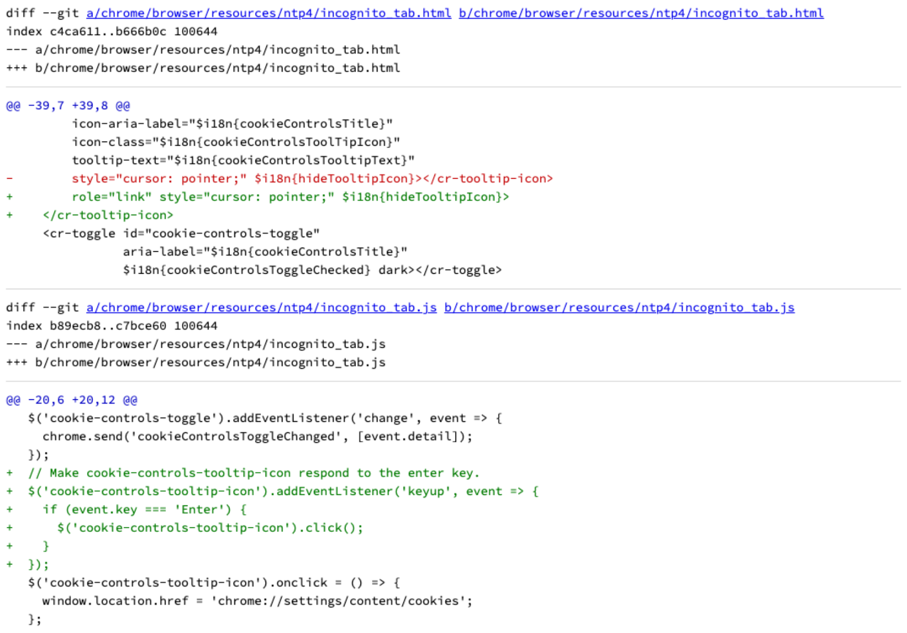
Mode of introduction: Phase: Implementation. Description: This weakness is caused when the users cannot change their privacy preferences in the software.

Common consequence: Scope: Privacy. Impact: Privacy violation. Note: User privacy is violated since the users cannot modify their privacy preferences.

Detection method: Method: Manual analysis. Description: Investigate the functions that involve with privacy preference settings.

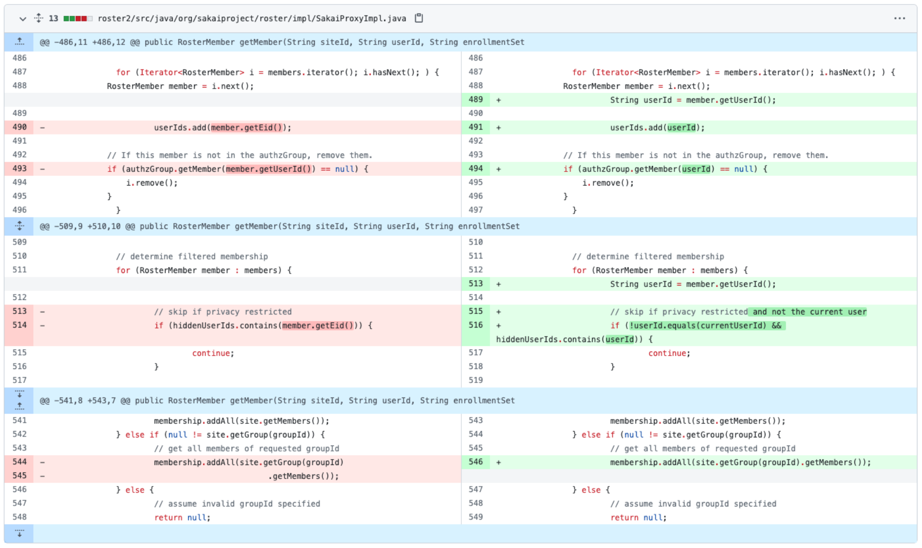
Potential mitigation: Phase: Implementation. Strategy: Allow to modify privacy preference settings. Description: The software development team should provide a function letting users to modify their privacy preferences.

Demonstrative examples: Issue 1048121 in Chrome reports that a user could not modify their *block third parties cookies* setting, which can be considered as a privacy preference. The browser showed the icon for the user to change his privacy preference setting, however nothing happened after he clicked that icon. References: `https://bugs.chromium.org/p/chromium/issues/detail?id=1048121` and `https://chromium.googlesource.com/chromium/src.git/+/e942d725842c0572e99321232541a5d16c6319a9\%5E\%21/\#F0`. Issue SAK-29299 in Sakai repository reports that a user failed to modify his privacy preference in the system. The user had modified his privacy status to hide himself in the system, however the system still showed his account. References: `https://jira.sakaiproject.org/browse/SAK-29299` and `https://github.com/sakaiproject/sakai/commit/84b7332`.



```
diff --git a/chrome/browser/resources/ntp4/incognito_tab.html b/chrome/browser/resources/ntp4/incognito_tab.html
index c4ca611..b666b0c 100644
--- a/chrome/browser/resources/ntp4/incognito_tab.html
+++ b/chrome/browser/resources/ntp4/incognito_tab.html

@@ -39,7 +39,8 @@
         icon-aria-label="$i18n{cookieControlsTitle}"
         icon-class="$i18n{cookieControlsToolTipIcon}"
         tooltip-text="$i18n{cookieControlsTooltipText}"
-        style="cursor: pointer;" $i18n{hideTooltipIcon}></cr-tooltip-icon>
+        role="link" style="cursor: pointer;" $i18n{hideTooltipIcon}>
+    </cr-tooltip-icon>
     <cr-toggle id="cookie-controls-toggle"
            aria-label="$i18n{cookieControlsTitle}"
            $i18n{cookieControlsToggleChecked} dark></cr-toggle>

diff --git a/chrome/browser/resources/ntp4/incognito_tab.js b/chrome/browser/resources/ntp4/incognito_tab.js
index b89ecb8..c7bce60 100644
--- a/chrome/browser/resources/ntp4/incognito_tab.js
+++ b/chrome/browser/resources/ntp4/incognito_tab.js

@@ -20,6 +20,12 @@
   $('cookie-controls-toggle').addEventListener('change', event => {
     chrome.send('cookieControlsToggleChanged', [event.detail]);
   });
+  // Make cookie-controls-tooltip-icon respond to the enter key.
+  $('cookie-controls-tooltip-icon').addEventListener('keyup', event => {
+    if (event.key === 'Enter') {
+      $('cookie-controls-tooltip-icon').click();
+    }
+  });
   $('cookie-controls-tooltip-icon').onclick = () => {
     window.location.href = 'chrome://settings/content/cookies';
   };
```

Figure 3: An example of unsuccessful privacy preferences modification in Chrome (Issue 1048121).



```
13  roster2/src/java/org/sakaiproject/roster/impl/SakaiProxyImpl.java

@@ -486,11 +486,12 @@ public RosterMember getMember(String siteId, String userId, String enrollmentSet
```

```
              for (Iterator<RosterMember> i = members.iterator(); i.hasNext(); ) {
            RosterMember member = i.next();

-                   userIds.add(member.getEid());

            // If this member is not in the authzGroup, remove them.
-           if (authzGroup.getMember(member.getUserId()) == null) {
                i.remove();
            }
              }
```

```
              for (Iterator<RosterMember> i = members.iterator(); i.hasNext(); ) {
            RosterMember member = i.next();
+                   String userId = member.getUserId();

+                   userIds.add(userId);

            // If this member is not in the authzGroup, remove them.
+           if (authzGroup.getMember(userId) == null) {
                i.remove();
            }
              }
```

```
@@ -509,9 +510,10 @@ public RosterMember getMember(String siteId, String userId, String enrollmentSet
```

```
              // determine filtered membership
              for (RosterMember member : members) {

-                   // skip if privacy restricted
-                   if (hiddenUserIds.contains(member.getEid())) {
                            continue;
                    }

```

```
              // determine filtered membership
              for (RosterMember member : members) {
+                   String userId = member.getUserId();

+                   // skip if privacy restricted and not the current user
+                   if (!userId.equals(currentUserId) &&
hiddenUserIds.contains(userId)) {
                            continue;
                    }

```

```
@@ -541,8 +543,7 @@ public RosterMember getMember(String siteId, String userId, String enrollmentSet
```

```
                    membership.addAll(site.getMembers());
              } else if (null != site.getGroup(groupId)) {
                    // get all members of requested groupId
-                   membership.addAll(site.getGroup(groupId)
-                                               .getMembers());
              } else {
                    // assume invalid groupId specified
                    return null;
```

```
                    membership.addAll(site.getMembers());
              } else if (null != site.getGroup(groupId)) {
                    // get all members of requested groupId
+                   membership.addAll(site.getGroup(groupId).getMembers());
              } else {
                    // assume invalid groupId specified
                    return null;
```

Figure 4: An example of unsuccessful privacy preferences modification in Sakai repository (Issue SAK-29299).

Table 4: An example of missing a notice to notify changes in consent weakness.

Subcategory: Exclusion

Class: Not notifying an update notice of changes of consent

Name: Missing a notice to notify changes in consent.

Description: The software does not provide a function to notify users about any changes in consent. This makes users unaware of changes related to their personal data processing that they have given consent to.

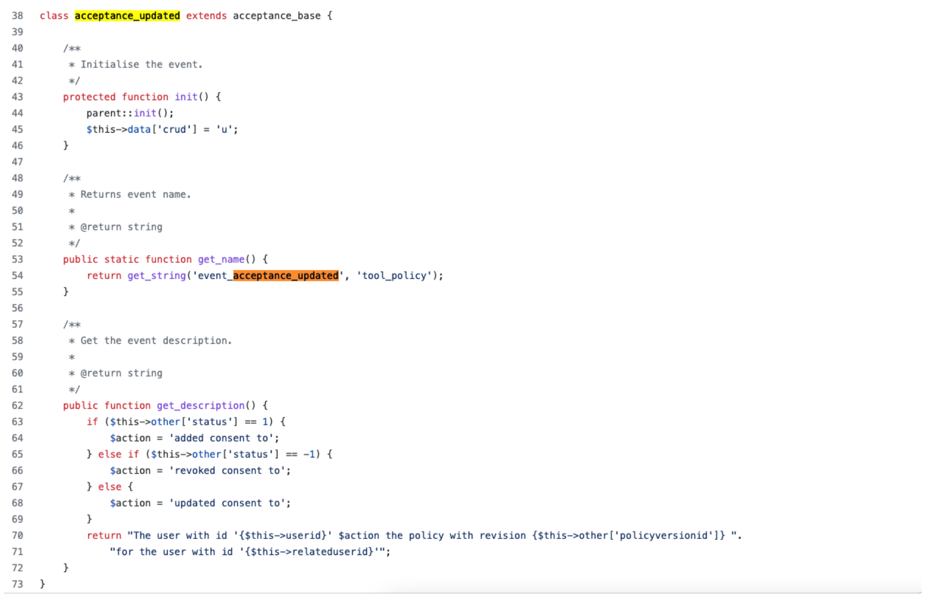
Mode of introduction: Phase: Implementation. Description: This weakness is caused when the consent related to their personal data processing is changed, but the users are not notified.

Common consequence: Scope: Privacy. Impact: Privacy violation. Note: Personal data may be used in the processing that is not satisfied based on user consent.

Detection method: Method: Manual analysis. Description: Investigate the functions involve with user consent.

Potential mitigation: Phase: Implementation. Strategy: Notify users. Description: The software development team should provide a function to send out an update notice to users when the conditions in the consent are changed.

Demonstrative example: Issue MDL-61273 in Moodle reports several tasks related to the implementation of a consent page. One of the tasks is to notify users when the consent is changed. The users need to accept the new version of consent before accessing the site. References: `https://tracker.moodle.org/browse/MDL-61273` and `https://github.com/moodle/moodle/blob/511a87f`.



```php
class acceptance_updated extends acceptance_base {

    /**
     * Initialise the event.
     */
    protected function init() {
        parent::init();
        $this->data['crud'] = 'u';
    }

    /**
     * Returns event name.
     *
     * @return string
     */
    public static function get_name() {
        return get_string('event_acceptance_updated', 'tool_policy');
    }

    /**
     * Get the event description.
     *
     * @return string
     */
    public function get_description() {
        if ($this->other['status'] == 1) {
            $action = 'added consent to';
        } else if ($this->other['status'] == -1) {
            $action = 'revoked consent to';
        } else {
            $action = 'updated consent to';
        }
        return "The user with id '{$this->userid}' $action the policy with revision {$this->other['policyversionid']} ".
            "for the user with id '{$this->relateduserid}'";
    }
}
```

Figure 5: An example of accepting the new version of consent in Moodle (Issue MDL-61273).

Table 5: An example of unsuccessful consent modification weakness.

Subcategory: Exclusion

Class: Not allowing a user to modify his/her consent

Name: Unsuccessful consent modification

Description: The software does not allow users to modify their consent.

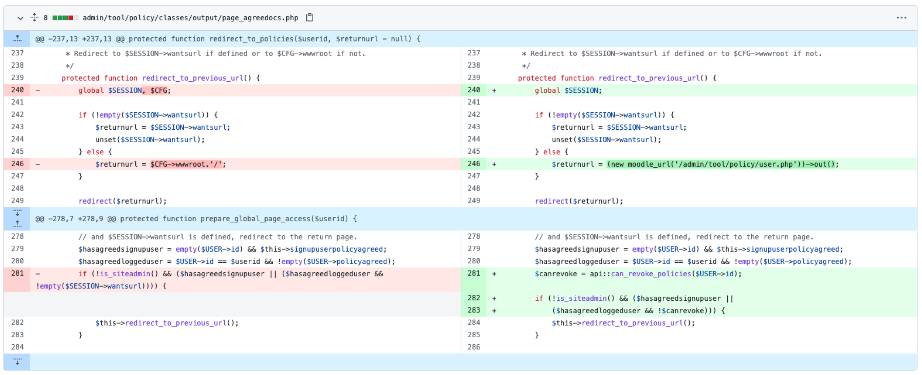
Mode of introduction: Phase: Implementation. Description: This weakness is caused when the users cannot modify their consent in the software.

Common consequence: Scope: Privacy. Impact: Privacy violation. Note: User privacy is violated since the users cannot make changes to their consent.

Detection method: Method: Manual analysis. Description: Investigate the functions involve with user consent.

Potential mitigation: Phase: Implementation. Strategy: Allow to modify consent. Description: The software development team should provide a function for users to successfully modify their consent.

Demonstrative example: Issue MDL-62062 in Moodle reports that a user cannot find a way to change her consent agreements once the consent agreements has been accepted. References: `https://tracker.moodle.org/browse/MDL-62062` and `https://github.com/moodle/moodle/commit/3d34aa5`. Issue MDL-63109 in Moodle reports that an administrator cannot successfully modify the consent status on behalf of the selected users. References: `https://tracker.moodle.org/browse/MDL-63109` and `https://github.com/moodle/moodle/commit/558126a`.



```
8 admin/tool/policy/classes/output/page_agreedocs.php
@@ -237,13 +237,13 @@ protected function redirect_to_policies($userid, $returnurl = null) {
```

```
     * Redirect to $SESSION->wantsurl if defined or to $CFG->wwwroot if not.
     */
    protected function redirect_to_previous_url() {
-       global $SESSION, $CFG;

        if (!empty($SESSION->wantsurl)) {
            $returnurl = $SESSION->wantsurl;
            unset($SESSION->wantsurl);
        } else {
-           $returnurl = $CFG->wwwroot.'/';
        }

        redirect($returnurl);
```

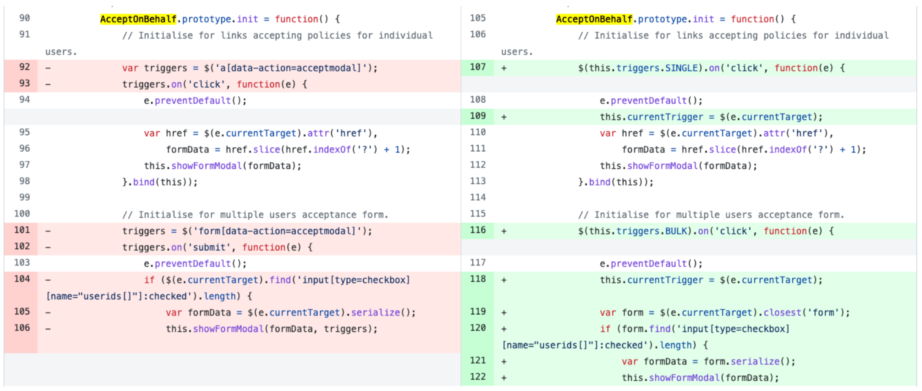
```
     * Redirect to $SESSION->wantsurl if defined or to $CFG->wwwroot if not.
     */
    protected function redirect_to_previous_url() {
+       global $SESSION;

        if (!empty($SESSION->wantsurl)) {
            $returnurl = $SESSION->wantsurl;
            unset($SESSION->wantsurl);
        } else {
+           $returnurl = (new moodle_url('/admin/tool/policy/user.php'))->out();
        }

        redirect($returnurl);
```

```
@@ -278,7 +278,9 @@ protected function prepare_global_page_access($userid) {
```

```
        // and $SESSION->wantsurl is defined, redirect to the return page.
        $hasagreedsignupuser = empty($USER->id) && $this->signupuserpolicyagreed;
        $hasagreedloggeduser = $USER->id == $userid && !empty($USER->policyagreed);
-       if (!is_siteadmin() && ($hasagreedsignupuser || ($hasagreedloggeduser &&
!empty($SESSION->wantsurl)))) {
            $this->redirect_to_previous_url();
        }

```

```
        // and $SESSION->wantsurl is defined, redirect to the return page.
        $hasagreedsignupuser = empty($USER->id) && $this->signupuserpolicyagreed;
        $hasagreedloggeduser = $USER->id == $userid && !empty($USER->policyagreed);
+       $canrevoke = api::can_revoke_policies($USER->id);

+       if (!is_siteadmin() && ($hasagreedsignupuser ||
+           ($hasagreedloggeduser && !$canrevoke))) {
            $this->redirect_to_previous_url();
        }

```

Figure 6: An example of unsuccessful consent modification in Moodle (Issue MDL-62062).



```
        AcceptOnBehalf.prototype.init = function() {
            // Initialise for links accepting policies for individual
users.
-           var triggers = $('a[data-action=acceptmodal]');
-           triggers.on('click', function(e) {
                e.preventDefault();

                var href = $(e.currentTarget).attr('href'),
                    formData = href.slice(href.indexOf('?') + 1);
                this.showFormModal(formData);
            }.bind(this));

            // Initialise for multiple users acceptance form.
-           triggers = $('form[data-action=acceptmodal]');
-           triggers.on('submit', function(e) {
                e.preventDefault();
-               if ($(e.currentTarget).find('input[type=checkbox]
[name="userids[]"]:checked').length) {
-                   var formData = $(e.currentTarget).serialize();
-                   this.showFormModal(formData, triggers);
```

```
        AcceptOnBehalf.prototype.init = function() {
            // Initialise for links accepting policies for individual
users.
+           $(this.triggers.SINGLE).on('click', function(e) {

                e.preventDefault();
+               this.currentTrigger = $(e.currentTarget);
                var href = $(e.currentTarget).attr('href'),
                    formData = href.slice(href.indexOf('?') + 1);
                this.showFormModal(formData);
            }.bind(this));

            // Initialise for multiple users acceptance form.
+           $(this.triggers.BULK).on('click', function(e) {

                e.preventDefault();
+               this.currentTrigger = $(e.currentTarget);
+               var form = $(e.currentTarget).closest('form');
+               if (form.find('input[type=checkbox]
[name="userids[]"]:checked').length) {
+                   var formData = form.serialize();
+                   this.showFormModal(formData);
```

Figure 7: An example of unsuccessful consent modification in Moodle (Issue MDL-63109).

Table 6: An example of missing consent withdrawal weakness.

Subcategory: Exclusion

Class: Not allowing a user to withdraw his/her consent

Name: Missing consent withdrawal

Description: The software forces users to give consent before providing its services. These services may include personal data processing. However, the software does not a provide a function for users to withdraw their consent. The users can only accept consent, but they cannot withdraw their consent when they wish to. This vulnerability seriously violates user privacy.

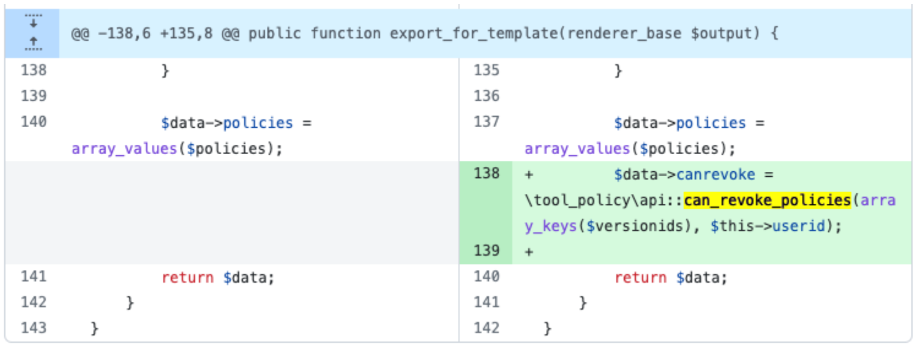
Mode of introduction: Phase: Architecture and Design. This weakness is caused by a missing privacy consideration about consent management and its related processes, which leads to the missing consent withdrawal function.

Common consequence: The software violates user privacy by not allowing users to express their agreement on the use of their personal data.

Detection method: Method: Manual analysis. Description: A consent management page or window in a software does not show an icon or option to withdraw consent.

Potential mitigation: Phase: Architecture and Design. Strategy: Consider a user consent withdrawal. Description: The software development team should consider which points in the software that should provide a user an ability to withdraw consent.

Demonstrative example: Issue MDL-62309 in Moodle reports that the users cannot withdraw consent and cannot enter the site without giving consent. This issue violates user privacy as the users should be able to freely withdraw consent. Reference: `https://tracker.moodle.org/browse/MDL-62309`.



```
@@ -138,6 +135,8 @@ public function export_for_template(renderer_base $output) {
            }                                          }

            $data->policies =                          $data->policies =
array_values($policies);                   array_values($policies);
                                           +          $data->canrevoke =
                                           \tool_policy\api::can_revoke_policies(arra
                                           y_keys($versionids), $this->userid);
                                           +
            return $data;                              return $data;
        }                                          }
    }                                          }
```

Figure 8: An example of missing consent withdrawal function in Moodle (Issue MDL-62309).

Table 7: An example of collecting personal data without user consent/permissions weakness.

Subcategory: Surveillance

Class: Collecting personal data without user consent/permissions

Name: Missing a consent check before collecting personal data

Description: The software does not check for a user consent prior to personal data collection. This makes the software collect personal data that users have not given consent to (e.g., location and speech).

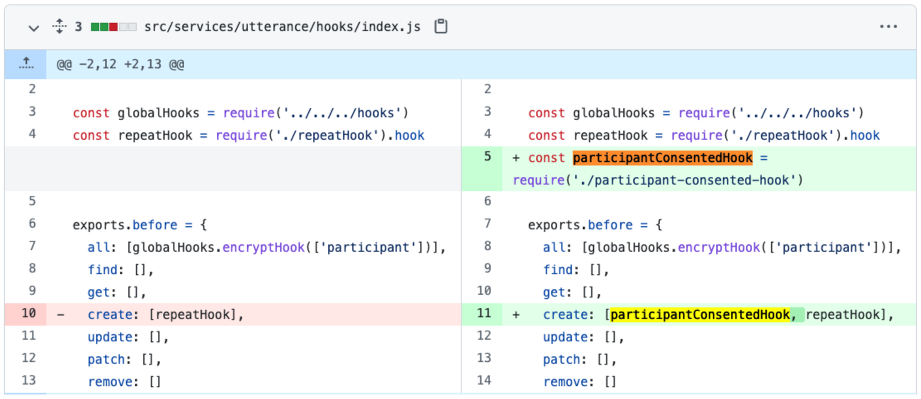
Mode of introduction: Phase: Implementation. This weakness is caused by missing a consent check before collecting personal data.

Common consequence: The software violates user privacy since users have not given consent/permissions to collect their personal data.

Detection method: Method: Manual analysis. Description: Perform a code check at points of personal data collection.

Potential mitigation: Phase: Implementation. Strategy: Check for user consent before collecting data. Description: The software development team should perform a consent check at every point that collects personal data in the software.

Demonstrative example: A commit 0b09df0 in HumanDynamics/rhythm-server repository collects user speech without consent check. Reference: `https://github.com/HumanDynamics/rhythm-server/commit/0b09df0`.



```
src/services/utterance/hooks/index.js
@@ -2,12 +2,12 @@
```

```
   const globalHooks = require('../../../hooks')
   const repeatHook = require('./repeatHook').hook

   exports.before = {
     all: [globalHooks.encryptHook(['participant'])],
     find: [],
     get: [],
-    create: [repeatHook],
     update: [],
     patch: [],
     remove: []
```

```
   const globalHooks = require('../../../hooks')
   const repeatHook = require('./repeatHook').hook
+  const participantConsentedHook = require('./participant-consented-hook')

   exports.before = {
     all: [globalHooks.encryptHook(['participant'])],
     find: [],
     get: [],
+    create: [participantConsentedHook, repeatHook],
     update: [],
     patch: [],
     remove: []
```

Figure 9: An example of missing a consent check before collecting personal data in HumanDynamics/rhythm-server repository (Commit 0b09df0).

Table 8: An example of using personal data for unspecified purposes weakness.

Subcategory: Secondary use

Class: Using personal data for unspecified purposes

Name: Using personal data for unspecified purposes

Description: The software uses personal data for any purpose other than the original purposes specified to the users. The users are not aware or have not given consent to the additional purposes.

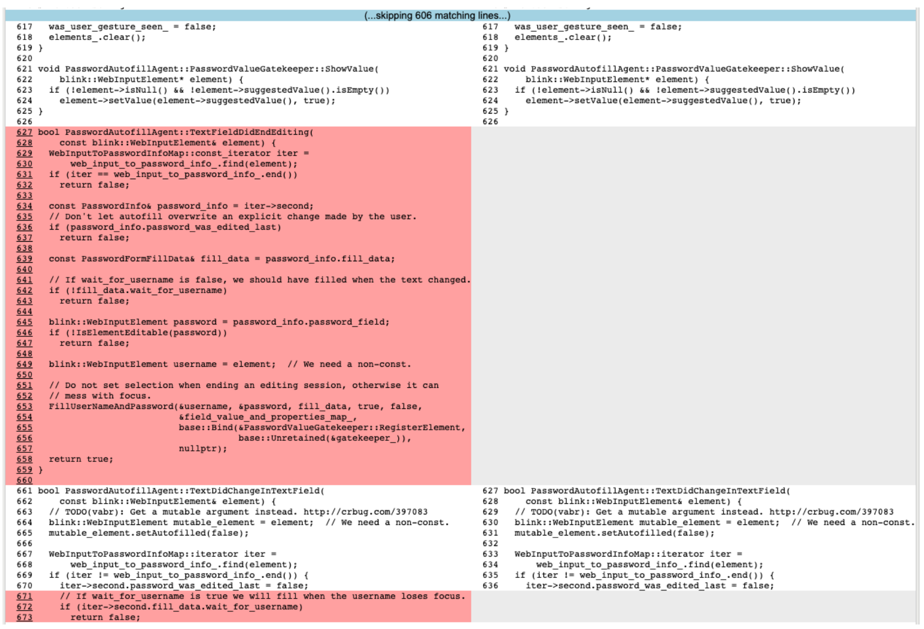
Mode of introduction: Phase: Implementation. Description: This weakness is caused when the software uses personal data for any purposes other than the original purposes.

Common consequence: Scope: Privacy. Impact: Privacy violation. Note: User privacy is violated since the users are not aware or have not given consent to use their personal data for unspecified purposes.

Detection method: Method: Manual analysis. Description: Perform a check if the software uses personal data for other purposes that are not informed to the users.

Potential mitigation: Phase: Implementation. Strategy: Check for a purpose before use. Description: The software development team should implement a function to check for the personal data that can be used for a specific purpose.

Demonstrative example: Issue 636461 in Chrome reports that the autocomplete password fields in incognito mode automatically fill the saved passwords. The user has accepted to the purpose of passwords autofill in normal mode. However, the browser applied this purpose in incognito mode which was not the user's intention. In incognito mode, the browser should preserve user privacy by not logging any traceable activities or identifiable personal data. References: `https://bugs.chromium.org/p/chromium/issues/detail?id=636461` and `https://codereview.chromium.org/2236413002/diff/1/components/autofill/content/renderer/password_autofill_agent.cc`.



(...skipping 606 matching lines...)

```
  was_user_gesture_seen_ = false;
  elements_.clear();
}

void PasswordAutofillAgent::PasswordValueGatekeeper::ShowValue(
    blink::WebInputElement* element) {
  if (!element->isNull() && !element->suggestedValue().isEmpty())
    element->setValue(element->suggestedValue(), true);
}

bool PasswordAutofillAgent::TextFieldDidEndEditing(
    const blink::WebInputElement& element) {
  WebInputToPasswordInfoMap::const_iterator iter =
      web_input_to_password_info_.find(element);
  if (iter == web_input_to_password_info_.end())
    return false;

  const PasswordInfo& password_info = iter->second;
  // Don't let autofill overwrite an explicit change made by the user.
  if (password_info.password_was_edited_last)
    return false;

  const PasswordFormFillData& fill_data = password_info.fill_data;

  // If wait_for_username is false, we should have filled when the text changed.
  if (!fill_data.wait_for_username)
    return false;

  blink::WebInputElement password = password_info.password_field;
  if (!IsElementEditable(password))
    return false;

  blink::WebInputElement username = element;  // We need a non-const.

  // Do not set selection when ending an editing session, otherwise it can
  // mess with focus.
  FillUserNameAndPassword(&username, &password, fill_data, true, false,
                          &field_value_and_properties_map_,
                          base::Bind(&PasswordValueGatekeeper::RegisterElement,
                                     base::Unretained(&gatekeeper_)),
                          nullptr);
  return true;
}

bool PasswordAutofillAgent::TextDidChangeInTextField(
    const blink::WebInputElement& element) {
  // TODO(vabr): Get a mutable argument instead. http://crbug.com/397083
  blink::WebInputElement mutable_element = element;  // We need a non-const.
  mutable_element.setAutofilled(false);

  WebInputToPasswordInfoMap::iterator iter =
      web_input_to_password_info_.find(element);
  if (iter != web_input_to_password_info_.end()) {
    iter->second.password_was_edited_last = false;
    // If wait_for_username is true we will fill when the username loses focus.
    if (iter->second.fill_data.wait_for_username)
      return false;
```

```
  was_user_gesture_seen_ = false;
  elements_.clear();
}

void PasswordAutofillAgent::PasswordValueGatekeeper::ShowValue(
    blink::WebInputElement* element) {
  if (!element->isNull() && !element->suggestedValue().isEmpty())
    element->setValue(element->suggestedValue(), true);
}

bool PasswordAutofillAgent::TextDidChangeInTextField(
    const blink::WebInputElement& element) {
  // TODO(vabr): Get a mutable argument instead. http://crbug.com/397083
  blink::WebInputElement mutable_element = element;  // We need a non-const.
  mutable_element.setAutofilled(false);

  WebInputToPasswordInfoMap::iterator iter =
      web_input_to_password_info_.find(element);
  if (iter != web_input_to_password_info_.end()) {
    iter->second.password_was_edited_last = false;
```

Figure 10: An example of using personal data for unspecified purposes in Chrome (Issue 636461).

Table 9: An example of missing a permission check before using personal data weakness.

Subcategory: Secondary use

Class: Using personal data without user permissions

Name: Missing a permission check before using personal data.

Description: The software does not check for user permissions before using their personal data in any personal data processing. Different users may allow the software to use their personal data for different purposes.

Mode of introduction: Phase: Implementation. Description: This weakness is caused by missing a check for user permissions before using personal data.

Common consequence: Scope: Privacy. Impact: Privacy violation. Note: Personal data is used without user permissions. This leads to user privacy violation.

Detection method: Method: Manual analysis. Description: Perform a code check at the functions that use personal data.

Potential mitigation: Phase: Implementation. Strategy: Check for user permissions before using personal data. Description: The software development team should implement a function to check for user permissions before using personal data in any personal data processing.

Demonstrative example: Issue SAK-26744 in Sakai repository reports that the requested friends function in the system does not check for user permissions before linking their profiles. Some users might not allow to show or link their profile publicly to preserve their privacy. The system therefore needs to check their privacy permissions before using their personal data. This issue was already fixed, but we do not have an access to the code of this issue. Reference: `https://jira.sakaiproject.org/browse/SAK-26744`.

Table 10: An example of missing a permission check before transferring personal data weakness.

Subcategory: Secondary use

Class: Transferring personal data without user permissions

Name: Missing a permission check before transferring personal data

Description: The software does not check if users allow their personal data to be transferred.

Mode of introduction: Phase: Implementation. Description: This weakness is caused by missing a check for user permissions before transferring personal data.

Common consequence: Scope: Privacy. Impact: Privacy violation. Note: Personal data is transferred without user permissions. This leads to user privacy violation.

Detection method: Method: Manual analysis. Description: Perform a code check at the functions that transfer personal data.

Potential mitigation: Phase: Implementation. Strategy: Check for user permissions before using personal data. Description: The software development team should implement a function to check for user permissions before transferring personal data.

Demonstrative example: Issue MDL-62549 requests the system to warn users before transferring user data to YouTube or other external platforms. This function reminds the users about the data transfer. In addition, it also checks for user permissions before tranferring data. This issue is still in progress, thus the code changes are not captured. Reference: `https://tracker.moodle.org/browse/MDL-62549`.

Table 11: An example of improper personal data protection at third parties weakness.

Subcategory: Insecurity

Class: Applying lower levels of personal data protection at third parties

Name: Improper personal data protection at third parties

Description: The software transfers personal data to third parties for processing. However, the third parties apply lower levels of personal data protection than the sources do.

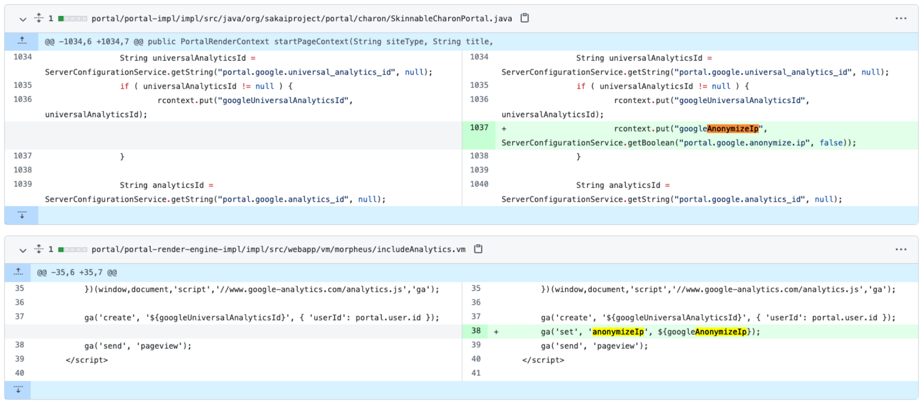
Mode of introduction: Phase: Implementation. Description: This weakness is caused when third parties apply lower levels of personal data protection than the sources do.

Common consequence: Scope: Privacy. Impact: Privacy leak. Note: This weakness may cause a privacy leak since personal data is not protected using the same or higher levels of protection.

Detection method: Method: Manual analysis. Description: Perform a check at the functions that process personal data at third parties site.

Potential mitigation: Phase: Implementation. Strategy: Implement equivalent personal data protection mechanisms. Description: The software development team has to inform third parties about the protection mechanisms that the source uses to protect personal data. The third parties then implement equivalent personal data protection mechanisms.

Demonstrative example: Issue SAK-41039 in Sakai repository reports that the IP addresses are not anonymised when using a third party plugin (i.e., Google Analytics). The users would like to anonymise their IP addresses when using the Google Analytics. References: `https://jira.sakaiproject.org/browse/SAK-41039` and `https://github.com/sakaiproject/sakai/commit/b1cedcd`.



```
1 portal/portal-impl/impl/src/java/org/sakaiproject/portal/charon/SkinnableCharonPortal.java
@@ -1034,6 +1034,7 @@ public PortalRenderContext startPageContext(String siteType, String title,
```

```
		String universalAnalyticsId =
ServerConfigurationService.getString("portal.google.universal_analytics_id", null);
		if ( universalAnalyticsId != null ) {
			rcontext.put("googleUniversalAnalyticsId",
universalAnalyticsId);
		}

		String analyticsId =
ServerConfigurationService.getString("portal.google.analytics_id", null);
```

```
		String universalAnalyticsId =
ServerConfigurationService.getString("portal.google.universal_analytics_id", null);
		if ( universalAnalyticsId != null ) {
			rcontext.put("googleUniversalAnalyticsId",
universalAnalyticsId);
+			rcontext.put("googleAnonymizeIp",
ServerConfigurationService.getBoolean("portal.google.anonymize.ip", false));
		}

		String analyticsId =
ServerConfigurationService.getString("portal.google.analytics_id", null);
```

```
1 portal/portal-render-engine-impl/impl/src/webapp/vm/morpheus/includeAnalytics.vm
@@ -35,6 +35,7 @@
```

```
        })(window,document,'script','//www.google-analytics.com/analytics.js','ga');

        ga('create', '${googleUniversalAnalyticsId}', { 'userId': portal.user.id });
        ga('send', 'pageview');
    </script>

```

```
        })(window,document,'script','//www.google-analytics.com/analytics.js','ga');

        ga('create', '${googleUniversalAnalyticsId}', { 'userId': portal.user.id });
+       ga('set', 'anonymizeIp', ${googleAnonymizeIp});
        ga('send', 'pageview');
    </script>

```

Figure 11: An example of improper personal data protection at third parties in Sakai repository (Issue SAK-41039).