Apps Script Code:

```
function URLinDocument() {
  const label = GmailApp.getUserLabelByName("processLinks");

  if (!label) {
    Logger.log("Label not found: processLinks.");
    return;
  }

  // Get all threads in the label
  const threads = label.getThreads();
  if (threads.length === 0) {
    Logger.log("No emails found in the label.");
    return;
  }


  // Create a new document (Google Docs) with the current date and time as
its name
  const dateTime = new Date();
  const documentName = Utilities.formatDate(dateTime,
Session.getScriptTimeZone(), 'yyyy-MM-dd_HH-mm-ss');
  const newDocument = DocumentApp.create(documentName);  // Create a new
Google Doc
  const body = newDocument.getBody();


  // Add a title to the document
  body.appendParagraph("Extracted URLs containing
&url=").setHeading(DocumentApp.ParagraphHeading.HEADING1);


  // Initialize an array to collect all URLs containing &url=
  let urlList = [];


  // Process each thread and extract URLs from the body of each message
  threads.forEach(thread => {
    const messages = thread.getMessages();
    messages.forEach(message => {
      const bodyText = message.getPlainBody();


      // Extract URLs containing &url= using the helper function
```

```
      const urlsInBody = extractUrlsContainingUrlParameter(bodyText);
      urlsInBody.forEach(url => {
        const actualUrl = extractRealUrl(url);  // Get the real URL from the
Google redirect
        if (actualUrl) {
          urlList.push(actualUrl);  // Add the real URL to the list
        }
      });


   // Optionally mark the email as read after processing
      message.markRead();
    });


    // Remove the thread from the label after processing
    label.removeFromThread(thread);
  });


  // Capture the logs (for debugging purposes, it will show processed URLs)
  const logOutput = Logger.getLog();  // Capture the logs after running the
script
  Logger.log("Log Output:\n" + logOutput);


  // Write the collected URLs to the document if there are any
  if (urlList.length > 0) {
    const uniqueUrls = Array.from(new Set(urlList));  // Remove duplicates
    uniqueUrls.forEach(url => {
      body.appendParagraph(url);  // Append each URL as a new paragraph
    });

    Logger.log(`Total unique URLs containing "&url=": ${uniqueUrls.length}`);
    Logger.log("URLs with &url=:\n" + uniqueUrls.join("\n"));
  } else {
    Logger.log("No URLs containing '&url=' found.");
  }
}


// Helper function to extract URLs containing &url=
function extractUrlsContainingUrlParameter(body) {
  const regex = /https?:\/\/[^\s]+[?&]url=[^\s]+/g; // Match URLs containing
?url= or &url=
  const matches = body.match(regex);
  return matches || [];  // Return the matched URLs or an empty array if none
found
```

```
}


// Helper function to extract the real URL from a Google redirect URL
function extractRealUrl(redirectUrl) {
  const regex = /[?&]url=([^&]+)/;  // Regex to match the `url=` parameter
  const match = redirectUrl.match(regex);

  if (match && match[1]) {
    return decodeURIComponent(match[1]);  // Decode the URL and return the
actual link
  } else {
    return null;  // Return null if no `url` parameter is found
  }}
```