Precisely's VDP – Introduction/Overview:

Welcome to our Vulnerability Disclosure Program. We value the security of our products and services and appreciate the efforts of researchers in helping us maintain a safe environment for our users.
This Program allows security researchers to responsibly test and report potential vulnerabilities within the defined scope. Please review the following sections carefully:

Safe Harbor Statement: Explanation of our Safe Harbor for researchers taking part in our VDP.
Initial setup prior to testing: How you should configure your testing tools and machine prior to participating in the VDP.
Rules of Engagement: Guidelines on how to conduct testing safely and legally.
Scope: Details of which systems, applications, and assets are in-scope (and out-of-scope).
Reporting Process: How to submit findings and what information to include.
Identifying vulnerabilities in a safe manner: Our rules for safe testing during identification of vulnerabilities.

By following these rules, you help us strengthen security while ensuring compliance and protecting all parties involved.

Safe Harbor statement for Bug Hunters participating in Precisely's VDP:

We value the contributions of security researchers in helping us maintain the security and integrity of our systems. This Safe Harbor provision ensures that individuals who engage in good-faith security research under this policy are protected from legal action by Precisely.

We recognise that mistakes can happen, even when acting in good faith. To encourage responsible disclosure and maintain trust:

- Researchers who follow our rules and act ethically will be granted safe harbor protection.
- If accidental damage occurs during testing within the defined guidelines, we will not pursue legal or punitive action.
- This protection applies only when researchers adhere to the scope and permitted actions outlined in this policy.

Safe harbour fosters a collaborative environment where researchers can confidently contribute to improving security without fear of disproportionate consequences.

By defining clear rules and offering safe harbour, we aim to create a transparent, fair, and secure process for vulnerability testing. These measures protect our systems, safeguard user data, and empower researchers to help us build stronger, more resilient products.

Scope of Protection:

- Activities conducted within the defined scope of this bug bounty program and in compliance with its rules will be considered authorised under applicable laws, including:
  - Computer Fraud and Abuse Act (CFAA)
  - Computer Misuse Act (CMA)
  - Digital Millennium Copyright Act (DMCA)
  - Relevant state and local computer crime laws
- Precisely will not pursue civil or criminal action, nor will we refer the matter to law enforcement, for accidental or good-faith violations of this policy.

Conditions for Safe Harbor:

To qualify for Safe Harbor protection, researchers must:

1. **Act in good faith** to identify and report security vulnerabilities.
2. **Follow the program rules**, including:
   - Test only in-scope systems.
   - Avoid privacy violations, service disruption, or data destruction.
   - Do not access, modify, or exfiltrate personal or sensitive data.
3. **Report findings promptly** through our official channel: **vulnerability.disclosure@precisely.com**
4. **Allow reasonable time for remediation** before public disclosure.
5. **Cease testing immediately** if you encounter sensitive data or are unsure whether an action is permitted.

Legal Protections:

- We waive any potential DMCA claims for circumvention of technological measures within the program's scope.
- We consider authorised research under this policy as permitted conduct under applicable computer use laws.
- This Safe Harbor does not extend to third-party systems outside our control. If your research involves third-party assets, they may pursue legal action independently.

Important Notes:

- If in doubt about whether an activity is covered, contact us first at **vulnerability.disclosure@precisely.com**
- We reserve the right to determine whether a violation was accidental or in good faith, but proactive communication will weigh heavily in your favour.

Disclaimer:

Precisely will not initiate legal action against security researchers who act in good faith, comply with this policy, and report vulnerabilities responsibly. This protection includes activities authorised under applicable laws such as the CFAA, DMCA, and similar statutes.

However, this Safe Harbor is conditional and applies only when:

- All testing is conducted within the defined scope of the VDP.
- Researchers adhere strictly to the Rules of Engagement and avoid any activity that could cause harm, service disruption, or compromise sensitive data.
- Reports are submitted promptly and through the official channel, allowing Precisely reasonable time for remediation before any public disclosure.

Limitations of Protection:

- This Safe Harbor does not extend to actions involving third-party systems or assets outside Precisely's control. Any such activity may expose researchers to independent legal action by those parties.
- Precisely reserves the right to determine whether a violation was accidental or in good faith. Proactive communication and transparency will weigh heavily in favour of the researcher when making this determination.
- Safe Harbor does not cover malicious intent, fraudulent behaviour, or any activity that exceeds the boundaries of authorised testing.

Important Note:
If you are uncertain whether a specific action is permitted, you must contact us before proceeding. Failure to do so may result in loss of Safe Harbor protection.

Initial setup prior to testing:

Before you begin with your testing, please configure your testing tools in the manner described below, and additionally select a 4 to 6 digit unique numerical identifier that will be tied to your testing, then use that unique identifier in the situations outlined below. This is so we can easily determine that you're someone hunting on the VDP rather than a malicious hacker, and also so that we are able to determine which hunter you are based upon the UUID being used.

This will be useful in situations where two people make the same report, as we can then review our logs and see which UUID was associated with the original finding and which one found it afterwards, to help us determine which report to mark as a duplicate (although it should be noted that "duplicate" status will generally be assigned to whoever submitted the report after the original report was submitted, so if someone writes and submits their report before you do, then yours will be assigned as duplicate. If by sheer coincidence, both reports are submitted at the same time, then in that situation we can check our logs and look to see which UUID was used first, and then decide which report to mark as duplicate based upon that).

HTTP Header to use for testing

Configure your tool(s) so that this header is sent with all HTTP requests:

- *Configure burp suite or ZAP or whatever other tools you are using to send a specific HTTP header alongside all of your traffic, specifically the header that you send should be in the following format:*

X-Precisely-VDP: YourUsername-UUID

*With **"YourUsername"** being the alias/pseudonym that you are using for testing), and **"UUID"** being a 4 to 6 digit unique identifier chosen by yourself and used during testing.*

*The reasoning for this is firstly so that we know whoever is carrying out this testing is likely someone taking part in our VDP, hence the presence of the **X-Precisely-VDP** header inside HTTP requests. Secondly, the **YourUsername-UUID** header value indicates which specific bug hunter you are. This is important in case two different bug hunters come across the same issue and both report it – we can use the value of this header to determine which hunter found the bug originally, by looking at the HTTP requests associated with each report and checking the username contained in the header value of the HTTP request that was made first, so we know which bug hunter found the issue first, allowing us to triage the bug report made by the user who made that particular HTTP request first, whilst marking the second report as "duplicate"*

Email to use for testing

Registration of testing email accounts:

> ***YourUsername+VDP1337@mail-provider.com*** *with **"1337"** being the unique identifier that is specific to your account (You can let Precisely's security team know which unique identifier you have decided to choose so that we know who is testing what).*

*The reasoning for this is so that we know email functionality being tested that contains **+VDP<UUID>** in the address is being performed by someone taking part in the VDP, rather than it potentially being a cybercriminal testing email-related functionality in an attempt to hack us. Email addresses are formatted in a manner where anything after "+" gets ignored, so **user@mail.com** and **user+1@mail.com** are technically both the same address, and mail sent to **user+1@mail.com** will arrive in the inbox of **user@mail.com**.*

Name/Address/Phone to use for testing

Address Information:

Name: **YourUsername**

Address: **<UUID> Vuln Street, Bounty Town, United States of Bugland** (*any fake address*)*

Phone Number: **123-1234-5678** *(any fake number)*

*(With the house number for the address matching the UUID that you are using in your HTTP header and for your mail, so for example if your UUID is "**1337**" then you could input "**1337 fake street**" as the first line of the address)

*For some forms of this nature, there will be input validation on the "Name" section, preventing you from inputting your username and only allowing a real name instead. In this situation, just use any real name (for example "John Smith") – Additionally, some forms verify that you're inputting a valid address and will generate a drop-down list of addresses as you begin to type your address, allowing you to select which one is yours as it predicts addresses based on the address you're starting to type. If this is the case, then simply use a real address and instead of setting your UUID as the house number, set it as part of your phone number instead, for example: **123-<UUID>-4567***

Rules of Engagement:

PLEASE ENSURE THAT YOU READ THIS SECONDARY DOCUMENT BEFORE PARTICIPATING IN OUR VDP:

This section will include the rules of engagement for taking part in our VDP. Please read this section very carefully and ensure that you are not violating any of these rules. First we will cover some general rules of engagement that should be applied to the VDP as a whole, and then we will cover the specific rules for certain attack vectors (for example which actions you are permitted to perform when it comes to attempting to confirm whether or not a security issue exists. We will list the names of some attack vectors and then explain the manner in which they should be tested in order to ensure that they aren't tested excessively or in a manner where it could result in a significant amount of PII or local system information being disclosed).

Not following these rules could potentially get you banned from participating in the VDP, and if you outright ignore the rules (for example rather than reading the specific files we've specified to confirm whether or not an LFI exists, you decide to read every single server-side script and local system file on the target environment) then this would be considered an exception to our safe harbour for bug hunters and could potentially result in legal action. So, please read this next section very carefully and use your common sense whilst testing.

Our goal is to foster a collaborative environment where security researchers can help us improve resilience without fear of disproportionate consequences. To achieve this, we provide clear guidelines on what is permitted, what is prohibited, and how to report findings.

Purpose of These Rules

The Rules of Engagement exist to:

- Protect the integrity and availability of our systems.
- Safeguard sensitive and personal data.
- Ensure testing does not disrupt services or harm users.
- Provide clarity on acceptable behaviour and scope.

Key Principles

- **Good Faith Testing**: All research must be conducted in good faith, with the intent to identify and report vulnerabilities responsibly.
- **Scope Compliance**: Testing must be limited to assets explicitly listed as in-scope. Activities outside this scope are not authorised and may result in loss of Safe Harbor protection.
- **Non-Disruptive Behaviour**: Avoid any action that could degrade performance, interrupt services, or compromise data integrity.
- **Respect for Privacy**: Do not access, modify, or exfiltrate personal or sensitive information. If encountered accidentally, cease testing immediately and report the finding.
- **Transparency and Communication**: If you are unsure whether an action is permitted, contact us before proceeding. Proactive communication will weigh heavily in your favour.

Prohibited Activities

To maintain a safe and controlled testing environment, the following are strictly prohibited:

- Exploiting vulnerabilities beyond what is necessary to demonstrate impact.
- Using automated vulnerability scanners or tools that generate excessive traffic.
- Performing denial-of-service (DoS) attacks or any activity that impacts availability.
- Social engineering, phishing, or physical attacks.
- Testing third-party systems or services outside our control.

Reporting Expectations

- Report vulnerabilities promptly through the official channel.
- Provide sufficient detail for us to reproduce and validate the issue.
- Do not publicly disclose findings without explicit permission and only after remediation.

Consequences of Non-Compliance

Failure to follow these rules may result in removal from the programme. Deliberate violations—such as accessing unauthorised files or ignoring scope limitations—will be considered outside Safe Harbor protections and may lead to legal action.

Do:

- Act in *good faith* and *follow all program rules*.
- Report vulnerabilities *immediately* via the official channels.
- *Stop testing if you encounter sensitive data* or are unsure about an action.
- Use *non-intrusive methods* to confirm vulnerabilities.
- *Communicate proactively* if you need clarification.

Do NOT:

- Exploit vulnerabilities *beyond proof-of-concept*.
- Perform *DoS attacks* or *disrupt services*.
- Access, modify, or exfiltrate personal or sensitive data.
- Test *third-party systems outside of our control*.
- Use *automated vulnerability scanners* (recon automation is allowed).
- Engage in *phishing*, *social engineering*, or *physical attack*s.

Safe Harbour

- Applies only if:
    - Testing stays within scope.
    - Rules are followed.
    - Findings are reported responsibly.
- Malicious or reckless behaviour voids protection.

General rules for participation in the VDP

- As soon as any vulnerability has been identified, report it immediately. Do not attempt to escalate it. If you report it and you think that it might be possible to increase the severity, then still report it immediately and mention within your report that you think it can potentially be escalated in terms of severity, then (if given permission) you can attempt to increase the severity whilst the bug report is still open.
- You must be of legal age to carry out testing against our VDP.
- You must not publicly disclose any of your findings before we have addressed and fixed the issue. Once the issue is fixed, you may request permission to publicly disclose what you found, but unless you are given explicit permission to allow for disclosure, then doing so will be considered a violation of our program rules.
- Ensure that anything you are testing adheres to our in-scope targets. Look at the requests being made whilst you're testing, in case some requests are being made to a third-party service that isn't under the control of Precisely. If you see requests being made to an external third-party that has nothing to do with Precisely, then you are NOT permitted to test this. You need to ensure that any requests you are sending are requests that are being made to our in-scope assets and not an external service that is being interacted with via one of our in-scope assets (for example if "target.com" is in scope, but whilst testing "target.com" you see some HTTP requests being made to a third-party CDN, then you are not permitted to test that third-party CDN for bugs. Your testing needs to be strictly limited solely to our in-scope assets).
- Usage of automated vulnerability scanners is strictly prohibited. You may automate your reconnaissance and information gathering process, but under no circumstances are you allowed to automate the vulnerability identification process.
- Ideally, prior to carrying out your testing, email us with your email message body containing the IP address that you are going to be testing from. When sending this email, do so using the format that is specified above (containing your UUID).

Scope:

Below, you can find a list of our products and websites that are in scope for the VDP. If something is not listed as "in-scope" then you should automatically assume that it is out of scope. Any assets hosted by third-party providers are out of scope, and any assets hosted by acquisitions that were acquired less than 6 months ago are out of scope even in instances where their domains are part of a wildcard scope that would signify them as being "in scope".

When we refer to **"**in scope**"** assets within our Vulnerability Disclosure Program (VDP), we are talking about *systems, applications, services, and infrastructure that Precisely explicitly authorizes researchers to test*. These are the digital properties we own or control and are prepared to monitor, investigate, and remediate findings for. Any testing performed should remain strictly limited to these defined in-scope assets.

Conversely, **"**out of scope**"** assets are *systems that researchers are not permitted to test*. These may include third-party services, infrastructure not owned or controlled by Precisely, environments where testing could cause harm, or systems excluded for legal, operational, or safety reasons. Attempting to assess these assets could impact systems we do not manage and may fall outside our Safe Harbor protections.

In short:

- **In scope** = You *may* test these targets safely and report vulnerabilities found.
- **Out of scope** = You *must not* test these targets under any circumstances.

This distinction ensures that testing is conducted safely, responsibly, and within environments we can properly support and remediate.

Assets that are in scope

Any Precisely product hosted within Precisely-owned infrastructure.

Assets that are out of scope

Any Precisely product hosted within AWS, third-party, or other non-Precisely infrastructure.

Vulnerabilities/Attack Vectors that are out of scope

If an attack vector is not listed here, then you should generally assume that it is in scope and will be considered a valid vulnerability that can be reported. There might be the occasional exception where we have forgotten to list a specific attack vector here, if that is the case, then you should automatically assume that it is in scope if it is something that is considered medium or above within the CIA Triad, *unless it is only considered medium or above due to the "A" part of the CIA triad (i.e. it is considered an issue because it affects the availability of the target. If that is the case, then it is out of scope as we would consider it to fall under the category of DoS-Related bugs which we have listed as out of scope).* If you are unsure as to whether or not a certain attack scenario is in scope, simply contact us via email and we will clarify whether it is in scope or not.

When we describe certain vulnerabilities or attack vectors as **"**out of scope**"**, we mean that, even if they can be demonstrated, they *will not be accepted as valid submissions within the VDP*. These exclusions typically cover issues that pose minimal security risk, rely on unrealistic attack conditions, create unnecessary operational risk, or cannot be safely tested without disrupting services.

Out-of-scope vulnerabilities often include low-impact findings, expected or intentional system behaviour, theoretical issues without practical exploitation, and anything that could harm system availability or user experience. They may also include categories of testing that fall outside safe, responsible research practices (e.g., denial-of-service, social engineering, or attacks requiring physical access).

In short:

- Out of scope vulnerabilities = Issues that should *not* be tested or reported, even if you discover them during normal testing.
- These exclusions help ensure research remains safe, meaningful, and focused on findings with real security impact.

LIST OF OUT OF SCOPE VULNERABILITIES:

-
  -
    - **Any theoretical/hypothetical vulnerabilities without demonstrable impact.**
    - **Logout CSRF.**
    - **CSRF on forms that are accessible/available to anonymous unauthenticated users.**
    - **Full Path Disclosure/Stack Traces/Error Messages (*unless they lead to worthwhile information disclosure – Also note that these are still fine to make a note of and use them as part of an exploit chain. Vulnerabilities which involve abusing a Full Path Disclosure or information found in a stack trace in order to escalate it to something that is in scope, then this can still be reported as a valid issue. It is just FPD's or Stack Trace disclosure that can't be reported as a standalone issue, but if it's used as part of a chain to cause a vulnerability with a higher impact, then that is fine*).**
    - **Clickjacking / UI Redressing (*unless a significant impact can be demonstrated*).**
    - **DoS-Related bugs.**
    - **Missing HTTP Security Headers.**
    - **Lack of HttpOnly or Secure flag on cookies.**

- **Missing SPF/DMARC/etc for email.**
- **Zero-day exploits / newly-released exploit PoC's (*if they have been public for less than 30 days, then they are out of scope. If they have been public for more than 30 days then they are in scope*).**
- **Self-XSS.**
- **Exposed banner or service version information.**
- **Bruteforce *(unless a viable rate limit bypass technique is used in order to make the bruteforce attack possible*).**
- **Disclosure of known public files (*such as robots.txt*).**
- **Lack of HTTP 403 on directories (*unless the directory discloses excessive PII*).**
- **Anything that requires an excessive or unrealistic level of user interaction from the victim in order for exploitation to be viable.**
- **XSS that can only be triggered using a data: URI scheme as the payload (*since this is executing in a null context – to determine whether or not it is a null context, try outputting document.domain and if it displays the domain, then it's a valid report. If it displays an empty alert box then it is executing under a null context and is therefore invalid*).**
- **XSF (*Cross-Site Flashing*).**
- **Any attacks that involve phishing or social engineering.**
- **Text-based Injection (e.g. a reflection point without the ability to inject HTML or JavaScript).**
- **Vulnerabilities that only work in outdated browsers (*such as Internet Explorer*).**
- **Vulnerabilities that only work in situations where the victim needs to have a specific browser plugin or addon installed.**
- **Attacks that involve flooding someone's email (*for example by repeatedly sending "Forgot password?" requests to someone else's email address in order to flood their inbox with hundreds of emails*).**
- **Rate limit bypasses that don't have any viable security impact.**
- **Anything that requires local network access or physical access to Precisely's internal network or office machines.**
- **UI/UX bugs without security impact, or spelling mistakes.**
- **CSS Injection (*unless it allows for the possibility of dangling markup injection via @import)***
- **Missing CSP directives / CSP directives that can be bypassed *(only in scope if the insecure CSP is abused to***

trigger an XSS, HTML Injection, or Dangling Markup Injection attack)

- **MIME Sniffing (*unless an impact can be demonstrated as a result of it, e.g. XSS or MIME Confusion to bypass file extension limitations and trigger malicious code*).**
- **Attacks limited to outdated versions of SSL/TLS (*such as BEAST*).**
- **SSL forward secrecy not enabled.**
- **SSL Insecure Cipher Suites.**
- **Arbitrary File Deletion (*although this is a high-impact vulnerability that would be very useful for us to be aware of, we are listing this as out of scope as it's not possible to test for the existence of this vulnerability without causing damage to our systems by deleting webpages or scripts which could potentially break the intended functionality of the application being targeted. There is an exception, being that if the vulnerability exists within the webroot directory and can be confirmed by deleting the robots.txt file, then testing for it is permitted but only in this specific situation. This exception also applies in situations where a directory traversal is used as part of the arbitrary file deletion bug, but only if you have a Full Path Disclosure and can therefore traverse to the webroot directoy in order to delete that specific file*).**
- **Arbitrary File Overwrite (*same reasoning as above, with the one exception being that if the vulnerability exists in the webroot directory it can be tested for by attempting to overwrite the contents of robots.txt. It is not permitted in any other situations*).**
- **Presence of out of date software or libraries (*unless a valid vulnerability can be demonstrated as a result of the out of date software or library being used*).**
- **Supply chain attacks specifically crafted to target packages that we are known to use (*for example launching a supply chain attack against an NPM package used by one of our web applications and then using the results of that supply chain attack to target our application*).**

Submitting your report:

If you have identified something which you believe to be a valid security issue that isn't out of scope, then when it comes to writing and submitting your report, please do your best to follow the steps outlined below whilst writing and submitting your report. This will allow us to triage and assess your finding at a faster rate, resulting in us being able to respond to your report quicker, It will also reduce the chances of us having to send follow-up emails to you requesting for additional information (which again, saves time for both us and for you). In this section of the page, we will outline exactly what information should be included in your report, how your report should be formatted and structured, and the correct process for reaching out to our VDP triage team in the intended manner. All of this information can be found in the sections below:

What you should include in the report

When you have identified a potential vulnerability and you are ready to submit your report, you should attempt to include as much of the following information as possible:

-
  -
    - **The URL to the webpage that is vulnerable.**
    - **If this is known to you, include the specific version of the affected product,**
    - **A copy of the HTTP request and HTTP response that was sent and received when the vulnerability was executed.**
    - **Screenshots showing the vulnerability taking place.**
    - **A video recording showing the vulnerability taking place.**
    - **A clear and concise explanation of what the vulnerability entails, whether there are any required prerequisites, whether user interaction is necessary, and what sort of impact is caused as a result of the vulnerability.**
    - **A clear and concise list of steps to reproduce.**
    - **The name of the specific attack vector (and if possible, the specific CWE ID related to that particular attack vector).**
    - **The CVSS rating of the identified vulnerability (calculated via either NIST, first.org, or Mitre with the link to the calculated CVSS rating included within your report).**
    - **An automated PoC if applicable (The PoC should just confirm that the vulnerability exists by demonstrating it in the least dangerous or intrusive way possible, under no circumstances should the PoC be weaponized in any way).**

- **Your report should be submitted only once. If there is a delay in getting a response from our team, do NOT submit another report. You can leave a comment on your original report asking for an update, but it is not permitted for you to submit the same report multiple times.**

Properly formatting your report

Now that we've covered what should be included within the contents of a report, we will explain the manner in which a report should be formatted to include this information prior to submitting it:

- **The title of your report submission should be brief and straight to the point. It should include the name of the attack vector, the affected section of the product, and the resulting impact. No additional information should be included in the title. An example title could be "*IDOR affecing user profile functionality in Product X, resulting in leakage of PII*"**
- **In the main body for the report, the very first line should be a link to the calculator that you have used to determine the CVSS rating for this bug (with the CVSS rating itself included after the link). If applicable, you should also include the CWE ID for the specific attack vector after the CVSS rating, For example: "*http://url/to/CVSS/calculator/results-for-this-vuln (CVSS Rating: 8.2) – (CWE-88)*"**
- **After this, you should include a brief (no more than a few short paragraphs) technical explanation of the security issue, explaining a brief overview of the impact and the scale of who or what is affected, whether any prerequisites are required for exploitation such as access to a user account with elevated privileges or the requirement to trick someone into clicking on a link or whether any other form of user interaction is required, an explanation of which specific functionality is affected and in which manner it is affected, what you believe the maximum security impact of your report to be, with a brief explanation of what the worst-case scenario could look like in terms of impact when it comes to a realistic attack scenario, and finally a sentence or two explaining whether you personally agree with the CVSS rating, and if not, then what would your personal rating be (*either using the CVSS Rating itself, a "Low, Medium High, or Critical" rating, or the "P1 to P5" rating style*). If you disagree with the CVSS rating, please explain why so that we can take your reasoning into consideration,**

- **After this section, include a link to the vulnerable URL and vulnerable parameter name.**
- **Next, include your steps to reproduce, These should be clear and concise, so that even someone without any technical background could follow these steps and see the same end result as what you are getting.**
- **After your steps to reproduce, include a copy of the HTTP request that triggered the vulnerability (*just the final request that actually triggered it, not ALL of the HTTP requests sent throughout the entire steps to reproduce*). Also include the HTTP response that was returned, so that we can see which exact response was sent when you submitted the request.**
- **If you made one, include your (non-weaponized) automated PoC within your report. If it is a program or script (f*or example written in something such as Python or C*++) then include it as an email attachment within your submission, If it is something simple like a bash one-liner or a cURL command, just include it in the submitted report after your steps to reproduce and HTTP request/response. If you are attaching a program/script as your automated PoC, then attach the source code as a .txt file – for example if it is a C++ PoC then you must attach it as a plaintext document, not as a compiled executable.**
- **Finally, if you have any screenshots demonstrating the vulnerability or its impact, either paste the images into the bottom of your report, or alternatively attach them to the report email. If you are including a video demo of the vulnerability, attach it to the email in .mp4 or .avi format – Do NOT upload it to a video sharing site such as YouTube.**

Once all of this information has been included (or as much of it as possible) in the specific order outlined in the steps above, your report is now ready to be submitted. Please note that under no circumstances should you ever submit the same report more than once. At times, we may have a backlog of reports, so please be patient and we will do our best to get back to you as fast as possible. Submitting the same report multiple times will just end up creating a larger backlog.

Instructions for sending your report

Once you are sure that your report has been written in the correct format and contains all of the necessary details to convey the vulnerability that you are making a report of, in order to make the report itself, you can simply email it to us at **vulnerability.disclosure@precisely.com**

- Within your email title, alongside the brief description of the vulnerability as outlined in the "Properly formatting your report" section above, you should include your testing username and UUID in square brackets directly after the description of the issue. For example, if your report description is "*XSS affecting comment functionality in Product X*" then your email title should look like: "*XSS affecting comment functionality in Product X (YourUsername 1234)"* with "1234" being the UUID that you selected and have been using during your testing.

- When you are sending your email, use the same email that you used for testing, with the "+VDP" identifier that you use whilst testing email-related functionality.
- If your email bounces back or there is some other issue, then check our **security.txt** page for additional information on making your report. Information on our **security.txt** page can be found below:

What is security.txt and why are we using it?

A **security.txt** file is a simple, standardized way for organizations to publish how security researchers should contact them. Defined in RFC 9116, it acts like a digital signpost that tells researchers where to report potential vulnerabilities and what process to follow.

We link to our security.txt page so that anyone assessing our VDP has an easily discoverable, authoritative source for our official security contact details, disclosure policy, and any supporting information (e.g., encryption keys). This reduces confusion, ensures reports go to the right place, and supports responsible disclosure.

Including a security.txt file is now a common best practice across the industry—it increases transparency, improves the experience for external researchers, and strengthens the overall maturity of our vulnerability disclosure process.

https://precisely.com/.well-known/security.txt